МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

А. М. Кочкаров
Р.И. Селимсултанова

СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие для обучающихся на 2 курса направление подготовки 01.03.02 Прикладная математика и информатика (очной формы обучения)

г. Черкесск 2024 г

УДК 004.42
ББК 32.93.26
К 75

Рассмотрено на заседании кафедры «Математика»
Протокол № 1 от «31» августа 2023 г.
Рекомендовано к изданию редакционно-издательским советом СКГА.
Протокол № 26 от «29» сентября 2023 г.

Рецензенты:
Чех С.А.– Ст. преподаватель кафедры
Шапошникова О.И. – к. ф.-м. н., доцент

К 75 **Кочкаров, А.М.** Системы программирования: учебно-методическое пособие для обучающихся 2 курса направления подготовки 01.03.02 Прикладная математика и информатика (очной формы обучения) / А.М. Кочкаров, Р.И. Селимсултанова. – Черкесск: СКГА, 2024. – 76 с.

Учебно-методическое пособие предназначено для обучающихся направление подготовки 01.03.02 Прикладная математика, и информатика содержит краткий теоретический справочный материал по программированию на Python, варианты индивидуальных заданий к лабораторным работам.

УДК 004.42
ББК 32.93.26

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Python (Пайтон, Питон) – популярный современный язык программирования высокого уровня, реализующий принцип повышение производительности разработчика при максимальной простоте записи кода. Он отличается наличием как стандартных библиотек с большой объём необходимых функций, так и постоянно пополняющимся списком новых библиотек, направленных на решение прикладных задач.

По этой причине **Python** широко используется для компьютерного практикума по целому ряду дисциплин. Данное пособие предназначено для организации лабораторных занятий в компьютерном классе, входящих в учебный план дисциплины «Системы программирования» для обучающихся направления подготовки 01.03.02 Прикладная математика и информатика. По этой причине основной материал изложен конспективно и фактически представляет собой краткий справочник, необходимый для выполнения заданий, и сами задания. Задания делятся на фронтальные аудиторные, в том числе предназначенные для работы в мини-группах, и внеаудиторные, представленные в виде набора индивидуальных вариантов. Полученные на занятиях практические навыки предполагается в дальнейшем использовать для следующих дисциплин: «Системы и методы компьютерного моделирования», «Машинное обучение и анализ данных», «Компьютерная графика».

Учебно-методическое пособие предназначено для образовательного процесса, основанного на целенаправленной и контролируемой преподавателем самостоятельной работе студентов (как аудиторной, так и внеаудиторной), обеспечивающей развитие обще профессиональных и профессиональных компетенций. Актуальность представленных материалов, решенных в логике компетентностного подхода, обусловлена потребностями высшего образования и требованиями нового государственного стандарта по данному направлению.

Базовые элементы языка. Начало работы в IDLE

Для любой версии Python программный код на этом языке выполняется интерпретатором, который должен быть установлен на компьютере. В данном пособии используются два режима работы в Python: интерактивный режим и режим исполнения программного кода. Программа на языке Python в самой простой форме – это обычный текстовый файл с расширением *.py, содержащий инструкции (код). Например, файл с именем my_file.py, содержащий текст:

```
print('herzen university')
print(2**100)
```

является программой на языке Python (между прочим, второй оператор выведет 2100 = 1267650600228229401496703205376).

Работа в интерактивном режиме

Такой способ подходит для освоения элементов языка, проверки правильности инструкций и организации кратких расчетов. Вход в режим интерпретатора будем осуществлять через меню графического интерфейса **IDLE.** Python IDLE – интегрированная среда разработки на языке Python. (IDLE = **I**ntegrated **D**eve**L**opment **E**nvironment). Для этого следует запустить файл с именем IDLE (IDLE (Python 3.n GUI – 32(/64) bit)) в папке Python. В открывшемся окне **Python Shell** (оболочка Python), будет присутствовать приглашение для ввода инструкций (>>>). После ввода инструкции нажимаем **Enter.** В результате получим, например, следующую цепочку:

```
>>> 2*3
6
>>>
```

Среда **Python Shell** позволяет также открыть окно для написания кода программы и её запуска на выполнение. Для создания кода программы следует открыть меню **«файл»** пункт **«новый»** набрать код и запустить через **Run** (Запустить) → **Run Module** (Запустить модуль). Через меню **«файл»** также можно открыть и уже существующий файл. Другой способ – найти файл с расширением *.py и в контекстном меню такого файла выбрать пункт **Edit with IDLE** – откроется окно для редактирования кода программы и её запуска.

В рамках изучения курса возможности **Atom, Geany** и **PyCharm** не отличаются, поэтому выбор остается за Вами: **Atom** и **PyCharm** являются более современными и популярными IDE, в то время как **Geany** обладает необходимыми возможностями и поддерживает русский язык в интерфейсе. В курсе достаточно использовать одну из IDE. Мы будем рассматривать PyCharm. Он использует концепцию проектов, поэтому в первую очередь необходимо создать пустой проект, для этого переходим в меню File →New Project, и, введя его название, нажать кнопку OK

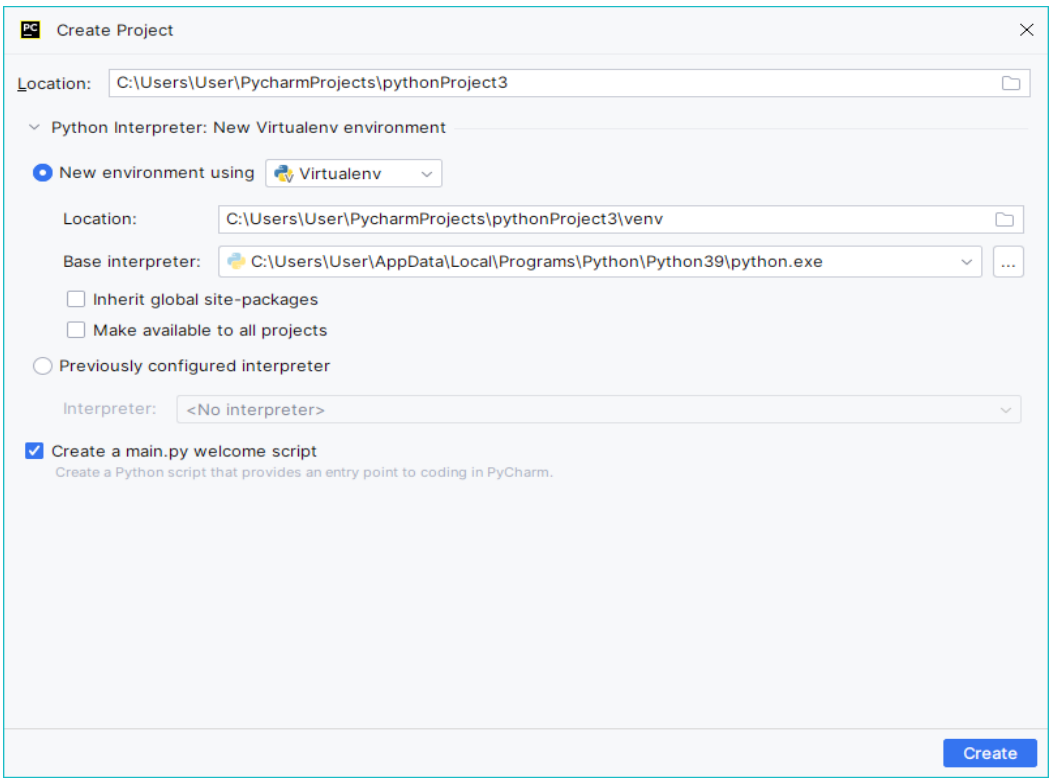

Рисунок 1 – Создание проекта IDE PyCharm

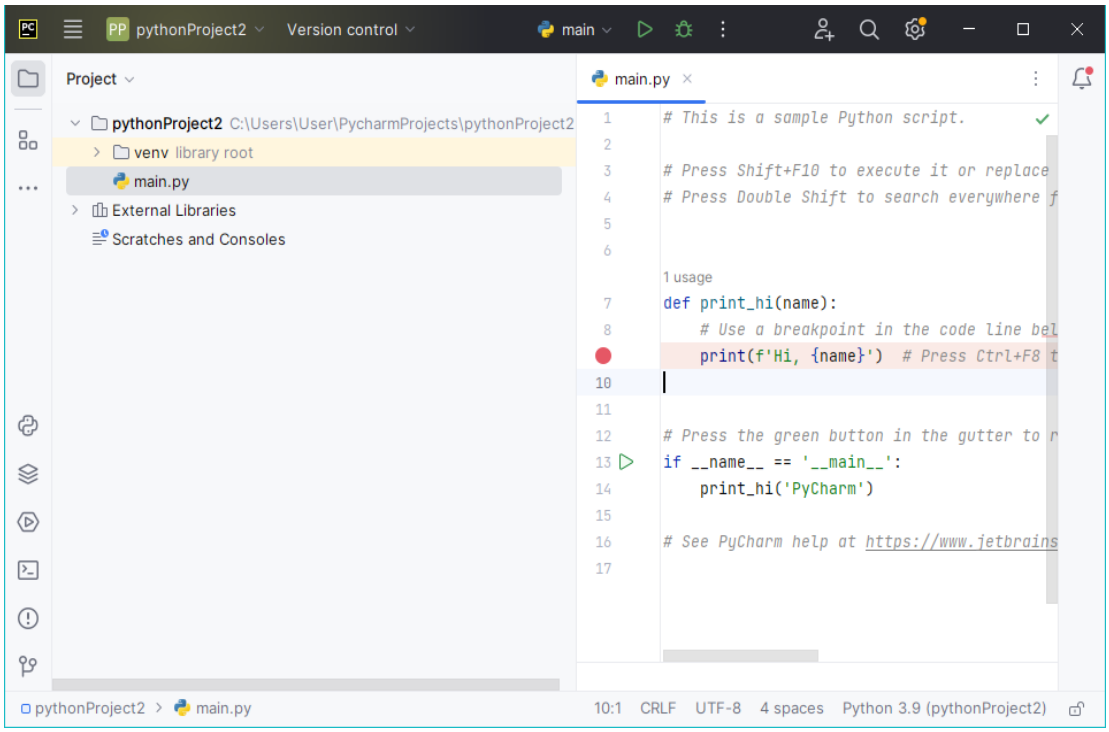

Рисунок 2. – Создание проекта IDE PyCharm

После открытия проекта необходимо выполнить нескольких шагов:

1. Для создания файла выберите меню File → New (Alt + Ins).

2. Выберите Python File и введите имя файла (например, 'main.py'), после чего подтвердите создание нажатием кнопки OK.

3. Наберите текст программы

4. Для запуска программы нажмите клавишу F5.

5. Протестируйте работу программы, запустив ее несколько раз, введя различные входные данные.

Инструкция (оператор) присваивания

Инструкция присваивания создает ссылку на объект. Переменные создаются при первом присваивании. Прежде чем переменную можно будет использовать, ей должно быть присвоено значение. Стандартная форма инструкции присваивания использует знак равенства: `x = 15`.

Важно: Python различает строчные и прописные буквы. В языке Python используется ***динамическая типизация*** (типы данных определяются автоматически и их не требуется объявлять в программном коде), но при этом он является языком со ***строгой типизацией*** (можно выполнять над объектом только те операции, которые применимы к его типу).

В **Python** предусмотрены расширенные варианты использования инструкции присваивания, которые приведены в таблице ниже:

Групповое присваивание	**Последовательное присваивание**	**Присваивание с перестановкой**
`a = b = 1`	`a, b = 1, 2`	`a, b = b, a`

`>> a=b=1`
`>> a,b`
`(1, 1)`
	`>>> a,b = 1,2`
	`>>> a,b`
	`(1, 2)`
		`>>> a,b = b,a`
		`>>> a,b`
		`(2, 1)`

Пример применения инструкции присваивания:

```
>>> X,Y = 4,8
>>> X,Y = Y, X*Y
>>> X,Y
(8, 32)
```

С помощью указателя звездочка (*) можно присваивать переменным значения элементов составных объектов. Сравните:

```
>>> a,*b='abcde'
>>> a
'a'
>>> b
['b','c','d','e']
```

А такой вариант выдаст ошибку: `a,b='abcde'`

Важно: количество переменных слева от оператора присваивания должно соответствовать набору элементов из списка присваивания справа от оператора присваивания.

Ввод-вывод в Python. Инструкции print() и input().

Для считывания строки со стандартного ввода используется функция `input()`, которая считывает строку с клавиатуры и возвращает значение

считанной строки, которое сразу же можно присвоить переменным: a=input(). Можно объединить считывание строк и преобразование типов, если вызывать функцию int для того значения, которое вернет функция input(): a = int(input('*текст*')).

Функции input() передает данные в программу, причем только форме строки. Например, инструкция x=input('*текст*'), предлагает ввести после сообщения '*текст*', некоторую строку, которая будет присвоена переменной x. Чтобы передать число, нужно преобразовать строку с помощью функций преобразования типов int(x) и float(x), например:

```
a = input('введите a')
b = int(a) # преобразует строку a в целое число
b = float(a) # преобразует a в вещественное число
```

Для вывода данных можно использовать функцию print(), формат записи которой имеет вид: print(*список вывода*). Например:

```
x=3
y= x**3
print(x,y). Выведет 3 27
```

Для разделения набора выводимых значений предусмотрены специальные атрибуты:

sep – набор символов, вставляемых между данными при выводе. Значение по умолчанию – пробел. Если sep = символ «пусто» (то есть '') – разделителя не будет.

end – набор символов, добавляемых в конец выводимого текста. По умолчанию это символ переноса строки \n. Если end = '', то следующий вывод через print продолжится без переноса строки. С учетом этих атрибутов функцию вывода записывают в виде

```
x=3; y=5
print(x,y,x*y, sep=' *** ', end='\n')
```

Выведет: 3 *** 5 *** 15

Рассмотрим программу с разными вариантами атрибутов вывода:

```
x = 'Program'; y = 123; z = 'python'
print(x,y,z)              # Выведет три объекта
print()                   # Выведет пустую строку
print(x,y,z, sep='')      #Разделителя не будет
print(x,y,z, sep=',')     #Разделитель запятая
```

При запуске программы получим следующие строчки:

```
Program 123 python

Program123python
Program,123,python
```

Арифметические выражения

Список основных наиболее употребляемых математических действий

`x + y` `x - y`	Сложение, Вычитание	`abs(x)`	Модуль числа
`x * y` `x / y`	Умножение, Деление	`divmod(x, y)`	Пара значений (`x // y,x % y`)
`x // y`	Целая часть от деления	`x ** y`	Возведение в степень x^y
`x % y`	Остаток от деления	`pow(x,y[,z])`	x^y (по модулю `z`, если модуль задан)

Обратите внимание: если записать `3.5 % 1` , то получим `0.5`.

Рассмотрим несколько простых примеров:

`> 7//2` `3`	`>>> 3*'abc'` `'abcabcabc'`	`>>> 2**8` `256`	`>>> pow(2,0.5)` `1.41421356237…`

Отметим, что строковые переменные типа `'abc'`можно записывать как в одинарных, так и в двойных кавычках. Как и в других языках, в Python имеется встроенный набор функций преобразования типов. Вот список некоторых из них:

`int(x)`	целая часть числа	`bin(x)`	перевод в двоичный код
`round(x)`	округление	`str(x)`	Перевод числа в строковое представление
`float(x)`	Перевод в число с плавающей точкой	`ord(x)`	Возвращает символа код
`int(x)`	целая часть числа	`bin(x)`	перевод в двоичный код

Важно отметить, что результат применения этих функций зависит от типа аргумента, например для `int` и `float` переменная `x` может быть строкой.

Лабораторная работа №1
Базовые элементы языка программирования Python

Цель работы: Создать программу для решения вычислительной задачи (например, по набору вводимых параметров найти по стандартным формулам заданную величину).Для ввода и вывода данных использовать инструкции print() и input()

Задачи к лабораторной работе № 1.1. Оператор присваивания. Математические операции.

1. Вычислите значение функции: $y = 3x + \sin(x + 2)$
2. Вычислите значение функции: $y = ax + \cos(2x + 1).$
3. Вычислите значение функции: $y = ax + b \cdot \sin(2x + 2).$
4. Вычислите значение функции: $y = ax^3 + cos(3x + 1).$
5. Вычислите значение функции: $y = \frac{x^2}{a} + \cos(2x - 1).$
6. Вычислите значение функции: $y = \frac{x}{a} + 2x$.
7. Вычислите значение функции: $y = 3x - 2x + 1.$
8. Вычислите значение функции: $y = 2\sqrt{x^3 + 1} - 2b.$
9. Вычислите значение функции: $y = \frac{1}{x^2+1} - a.$
10. Вычислите значение функции: $y = \frac{a}{x^2+1} - \cos(2x - 1).$
11. Вычислите значение функции: $y = x^3 - 2x + 4.$
12. Вычислите значение функции: $y = ax + bx^3 - 8.$
13. Вычислите значение функции: $y = a\sqrt{x + 4} - b.$
14. Вычислите значение функции: $y = \cos(2x - 1) + \sin x.$
15. Вычислите значение функции: $y = a\sqrt{x} + bx$
16. Вычислите значение функции: $y = abs(3e^x + 3 - 2\ln(x)) + 1).$
17. Вычислите значение функции: $y = tg(x) - abs(2\sin 2x + 7.8\cos x) + 10.$
18. Вычислите значение функции: $y = (x - 2\sin x)/abs(8x - 5\, arctg(3x + 1)).$
19. Вычислите значение функции: $y = 3x^3 + cos(x + 1).$
20. Вычислите значение функции: $y = \frac{x^3}{a} - \sin(2x + 1).$
21. Вычислите значение функции: $y = \frac{3-a}{x^2+1} + \sin(2x + 1).$
22. Вычислите значение функции: $y = -ax + bx + 4.$
23. Вычислите значение функции: $y = -\cos(3x + 1) + \sin 2x.$
24. Вычислите значение функции: $y = 2a\sqrt{x + 2} + b(x + 1).$
25. Вычислите значение функции: $y = \frac{x-1}{x^2+3} - ax.$

Задачи к лабораторной работе № 1.2. Оператор присваивания. Ввод-вывод информации.

1. Даны координаты трех вершин треугольника:(x_1, y_1), (x_2, y_2), (x_3, y_3) Найти его периметр и площадь, используя формулу для расстояния между

двумя точками на плоскости. Для нахождения площади треугольника со сторонами a, b, c использовать формулу Герона: $S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$, где $p = (a+b+c)/2$ – полупериметр.

2. Дано значение температуры T в градусах Фаренгейта. Определить значение этой же температуры в градусах Цельсия. Температура по Цельсию T_C и температура по Фаренгейту T_F связаны следующим соотношением: $T_C = (T_F - 32)\ 5/9$.

3. Найти решение системы линейных уравнений вида $\begin{cases} A_1 \cdot x + B_1 \cdot y = C_1 \\ A \cdot x + B \cdot y = C \end{cases}$, заданной своими коэффициентами A_1, B_1, C_1, A, B, C. Если известно, что данная система имеет единственное решение. Воспользоваться формулами:

$x = \frac{C_1 \cdot B_2 - C_2 \cdot B_1}{D}$, $y = \frac{A_1 \cdot C_2 - A_2 \cdot C_1}{D}$, где $D = A_1 \cdot B - A \cdot B_1$.

4. Дано значение угла α в градусах ($0 < \alpha < 360$). Определить значение этого же угла в радианах, учитывая, что 180 градусов равно π радианов.

5. Скорость первого автомобиля V_1 км/ч, второго — V_2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга. Данное расстояние равно сумме начального расстояния и общего пути, проделанного автомобилями; общий путь = время × суммарная скорость.

6. Даны три точки A, B, C на числовой оси. Найти длины отрезков AC и BC и их сумму

7. Даны два ненулевых числа. Найти сумму, разность, произведение и частное их квадратов.

8. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч ($U < V$). Время движения лодки по озеру T_1 ч, а по реке (против течения) — T_2 ч. Определить путь S, пройденный лодкой. Учесть, что при движении против течения скорость лодки уменьшается на величину скорости течения.

9. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^4 за две операции.

10. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^8 за три операции.

11. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^9 за четыре операции.

12. Вводится вещественное число a. Не пользуясь никакими арифметическими операциями, кроме умножения, получить a^{10} за четыре операции.

13. Найти сумму цифр введенного четырехзначного числа.

14. Введено трехзначное число. Вывести число в зеркальном отображении.
15. Определить сумму квадратов цифр введенного трехзначного числа.
16. Вводится четырехзначное число. Из его цифр получить два двузначных числа. Первое состоит из первой и третьей цифр исходного числа, второе — из второй и четвертой. Например, 3765 → 36 и 75.
17. Вводятся два трехзначных числа. Получить шестизначное число, состоящее из цифр исходных чисел. Например, 265 и 145 → 265145.
18. Вводится трехзначное число. Из его цифр получить два двузначных числа. Первое состоит из первой и третьей цифр исходного числа, второе — из второй и третьей. Например, 765 → 75 и 65.
19. Вводятся два числа: двузначное и трехзначное. Получить пятизначное число, состоящее из цифр исходных чисел. Например, 25 и 137 → 25137.
20. Составить программу, которая преобразует введенное с клавиатуры дробное число
21. в денежный формат. Например, число 45.7 должно быть преобразовано к виду 45 руб. 70 коп.
22. Составить программу для перевода суммы из долларов в рубли. Вводится текущий курс доллара и сумма в долларах. Результат должен выводиться в денежном формате, например, 345 руб. 50 коп.
23. Составить программу для вычисления величины дохода по вкладу. Вводятся величина вклада, процентная ставка (в процентах годовых) и время хранения (в днях).
24. Написать программу для вычисления стоимости поездки на дачу (туда и обратно).
25. Исходные данные: расстояние до дачи (в км), количество бензина, которое потребляет автомобиль на каждые 100 км, цена 1 л бензина.

Стандартные алгоритмические структуры
Операторы сравнения. Логические операторы. Условный оператор if (if-elif-else, if/else)

Операторы сравнения:

- < Меньше — условие верно, если первый операнд меньше второго.
- > Больше — условие верно, если первый операнд больше второго.
- <= Меньше или равно.
- >= Больше или равно.
- == Равенство. Условие верно, если два операнда равны.
- != Неравенство. Условие верно, если два операнда неравны.

Операторы сравнения в Python можно объединять в цепочки (в отличие от большинства других языков программирования, где для этого нужно использовать логические связки), например, `a == b == c` или `1<=x<= 10`

Операторы сравнения возвращают значения специального логического типа bool. Логические операторы: логическое И - and, логическое ИЛИ - or, логическое НЕ - not.

Например, необходимо проверить, что два данных целых числа *n* и *m* являются четными:

```
n % 2 == 0 and m % 2 == 0.
```

Проверим, что хотя бы одно из чисел *a* или *b* оканчивается на 0:

```
a % 10 == 0 or b % 10 == 0.
```

Проверим, что число *a* — положительное, а *b* — неотрицательное:

```
a > 0 and not (b < 0)
```

Операторы ветвления (***условные операторы***) предназначены для выбора к исполнению одного из возможных действий (операторов) в зависимости от некоторого условия (при этом одно из действий может быть пустым, т. е. отсутствовать). В качестве условий выбора используется значение логического выражения.

Синтаксис условного оператора:

```
if выражение_1:
    инструкции_1
elif выражение_2:
    инструкции_2
elif выражение_N:
    инструкции_N
else:
    инструкции
```

Общая форма: Синтаксически сначала записывается часть **if** с условным выражением, далее могут следовать одна или более необязательных частей `elif` («else if») с условными выражениями и, наконец, необязательная часть `else`. Условные выражения и часть `else` имеют привязанные к ним блоки исполняемых инструкций, с отступом относительно основной инструкции.

В условной инструкции могут отсутствовать части elif, else и последующий блок. Общий вид синтаксиса тернарного условного оператора: `true_result if инструкция_1 else false_result`.

В самой простой форме условный оператор (инструкция) **if** реализуется с помощью интуитивно понятного синтаксиса.

```
>>> if X>0.5: print(X)
```

Рассмотрим программу:

```
x=5; y=3
if y>0.5: #двоеточие после условия обязательно!
x=10
y=3*y
print(x,y,x*y, sep=' ** ', end='\n')
```

Результат работы программы имеет вид: `10 ** 9 ** 90`

Обратите внимание: ключевое слово **if** записываются маленькими буквами, обязательно присутствует двоеточие после условия и **тело инструкции** в примере выделяется с помощью отступа.

Допускается записывать одну инструкцию в нескольких строках, Для этого достаточно заключить часть инструкции в пару скобок – круглых ( ), квадратных [ ] или фигурных { }.

```
if (a == 1 and b == 2 and c == 3): print('x=', x)
```

Пример. Написать программу, определяющую четверть координатной плоскости, в которой находится точка с координатами (*x*, *y*)

```
x = int(input()
y = int(input())
if x>0 and y>0:
     print('Первая четверть')
elif x>0 and y<0:
    print('четвертая четверть')
elif y>0:
    print('Вторая четверть')
else:
    print('Третья четверть')
```

Сокращенные формы условного выражения `if/else` имеют вид

`A = X` **`if`** `<условие>` **`else`** `Y`

интерпретатор выполняет выражение X, если `<условие>`= «истина», иначе выполняется Y . Сравните эквивалентные инструкции:

```
if y>3: x=5
else: x=3
```

```
x=5 if y>3 else 3
```

Лабораторная работа №2
Условный оператор. Вычисление функции.

Цель занятия: получение теоретических знаний и практических навыков работы с условными операторами на Python.

Задачи к лабораторной работе № 2.1

1) Вычислите значение функции: $y = \begin{cases} ax + 1, & x > 0, \\ ax - 1, & x \le 0. \end{cases}$

2) Вычислите значение функции: $y = \begin{cases} x - 1, & x < 1, \\ ax + 1, & x \ge 1. \end{cases}$

3) Вычислите значение функции: $y = \begin{cases} 3a, & x < 0, \\ 4ax - 1, & x \ge 0. \end{cases}$

4) Вычислите значение функции: $y = \begin{cases} 2a, & x > 4, \\ 3x - 1, & x \le 4. \end{cases}$

5) Вычислите значение функции: $y = \begin{cases} 2ax - 2, & x > 2, \\ 3a - 2x, & x \le 2. \end{cases}$

6) Вычислите значение функции: $y = \begin{cases} 2ax - 1, & x > 1, \\ x, & x \le 1. \end{cases}$

7) Вычислите значение функции: $y = \begin{cases} x, & x > 2, \\ 2a - 1, & x \le 2. \end{cases}$

8) Вычислите значение функции: $y = \begin{cases} \cos(2x - 1), & x > 2, \\ \sin(3x + 1), & x \le 2. \end{cases}$

9) Вычислите значение функции: $y = \begin{cases} 2x^3 - 2x - 1, & x > 2, \\ 3x - 2x + 1, & x \le 2. \end{cases}$

10) Вычислите значение функции: $y = \begin{cases} 2ax - 1, & x > 1, \\ 1/a, & x \le 1. \end{cases}$

11) Вычислите значение функции: $y = \begin{cases} a\sqrt{x} + 1, & x \ge 0, \\ ax - 1, & x < 0. \end{cases}$

12) Вычислите значение функции: $y = \begin{cases} \sqrt{x} + a, & x \ge 0, \\ a/x - 1, & x < 0. \end{cases}$

13) Вычислите значение функции: $y = \begin{cases} 1/x + a, & x > 0, \\ x - 1, & x \le 0. \end{cases}$

14) Вычислите значение функции: $y = \begin{cases} \cos(x), & x > \pi/2, \\ \sin(x), & x \le \pi/2. \end{cases}$

15) Вычислите значение функции: $y = \begin{cases} \sqrt{x - 2}, & x > 2, \\ (x - 2) + 1, & x \le 2. \end{cases}$

16) Вычислите значение функции: $y = \begin{cases} ax - 3, & x > 2, \\ ax - 3, & x \le 2. \end{cases}$

17) Вычислите значение функции: $y = \begin{cases} x - 2x + 1, & x < 3, \\ ax + 3, & x \ge 3. \end{cases}$

18) Вычислите значение функции: $y = \begin{cases} 3xa + 1, & x < 0, \\ 2x - a, & x \ge 0. \end{cases}$

19) Вычислите значение функции: $y = \begin{cases} 2a + x, & x > 2, \\ 4x - 3, & x \le 2. \end{cases}$

20) Вычислите значение функции: $y = \begin{cases} 3ax - 1, & x > 0, \\ 2a - 3x, & x \le 0. \end{cases}$

21) Вычислите значение функции: $y = \begin{cases} 2ax - 3x + 1, & x > 1, \\ 3x - 4, & x \le 1. \end{cases}$

22) Вычислите значение функции: $y = \begin{cases} 2x + 1, & x > 2, \\ 2ax - 1, & x \le 2. \end{cases}$

23) Вычислите значение функции: $y = \begin{cases} \cos(2x + 3), & x > 3, \\ \sin(2x - 1), & x \le 3. \end{cases}$

24) Вычислите значение функции: $y = \begin{cases} 2x^3 - 2x - 2, & x > 2, \\ 3x + 2x - 1, & x \le 2. \end{cases}$

25) Вычислите значение функции: $y = \begin{cases} 2ax + 1, & x > 0, \\ 2/ax, & x \le 0. \end{cases}$

26) Вычислите значение функции: $y = \begin{cases} 2a\sqrt{x} + 3, & x \ge 0, \\ 3ax - 1, & x < 0. \end{cases}$

27) Вычислите значение функции: $y = \begin{cases} 2\sqrt{x} + 3a, & x \ge 0, \\ \frac{5a}{x} + 2, & x < 0. \end{cases}$

28) Вычислите значение функции: $y = \begin{cases} 3/x + ax, & x > 0, \\ 2x + 1, & x \le 0. \end{cases}$

29) Вычислите значение функции: $y = \begin{cases} \cos(x) + 2, & x > \pi/2, \\ \sin(x) - 1, & x \le \pi/2. \end{cases}$

30) Вычислите значение функции: $y = \begin{cases} 3\sqrt{x - 2} - 1, & x > 2, \\ 2(x - 2) + 3, & x \le 2. \end{cases}$

Задачи к лабораторной работе № 2.2

1) Вводятся координаты точки (*x*, *y*). Определить попадает ли точка в область заданную условиями: $y \le 5 \sin x,\ y \ge 0,\ 0 \le x \le \pi$.
2) Вводятся координаты точки *(x, y)*. Определить попадает ли точка в область заданную условиями: $y \le 3 \sin x,\ y \ge 0,\ 0 \le x \le \pi$.
3) Вводятся координаты точки *(x, y)*. Определить попадает ли точка в область заданную условиями: $x^2 + y^2 \le 4,\ x^2 + y^2 \ge 1$.
4) Введено трехзначное число. Найти сумму четных цифр.
5) Введено четырехзначное число. Содержится ли в записи этого числа цифра 7?
6) Введено трехзначное число. Если в записи числа встречается цифра 5, то записать число в зеркальном отображении.
7) Введено трехзначное число. Если сумма его цифр нечетна, то увеличить число вдвое.

8) Введено четырехзначное число. Найти сумму цифр, кратных трем.
9) Вводятся X и Y. Если хотя бы одно из этих чисел положительно, то найти их произведение. Иначе — найти их сумму.
10) Вводятся X и Y. Если X больше Y, то произвести их обмен.
11) Из чисел A, B, C, D выбрать максимальное.
12) Вводятся X и Y. Заменить большее из этих чисел разностью большего и меньшего.
13) Сколько среди заданных чисел A, B, C, D нечетных.
14) Вводятся A, B, C, D. Найти среднее арифметическое максимального и минимального.
15) Вводятся числа A, B, C. Упорядочить их по убыванию.
16) Из чисел A, B, C, D выбрать минимальное.
17) Определить, какие из чисел A, B, C, D принадлежат интервалу $(-10, 15)$.
18) Определить, есть ли среди цифр заданного трехзначного числа одинаковые.
19) Составить программу, определяющую является ли билет с шестизначным номером счастливым (счастливым является билет, у которого сумма первых трех десятичных цифр равна сумме трех последних).
20) Вводится трехзначное число. Определить, является ли оно числом Армстронга. Число Армстронга — натуральное число, которое равно сумме своих цифр, возведённых в степень, равную количеству его цифр. Например, число 153 — число Армстронга, потому что $1^3 + 5^3 + 3^3 = 153$.
21) Вводятся положительные a, b, c d. Определить, можно ли прямоугольник со сторонами a, b поместить внутри прямоугольника со сторонами c, d так, чтобы каждая из сторон первого прямоугольника была параллельна или перпендикулярна каждой стороне второго треугольника.
22) Составить программу, проверяющую знание таблицы умножения. В программе выбираются случайным образом целые числа X ($1 \le X \le 9$) и Y ($1 \le Y \le 9$) и предлагается пользователю ввести ответ. Результат выполнения программы: «Правильно» или «Неправильно».
23) Вводятся числа A, B, C. Упорядочить их по возрастанию.
24) Составить программу, которая по введенной начальной букве выводит название цветов радуги (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый).

25) Составить программу, которая по введенному порядковому номеру выводит название дня недели.
26) Составить программу, которая позволяет ввести номер месяца и вывести его название.
27) По числу текущего месяца определить день недели.
28) Напишите программу, которая по номеру дня недели — целому числу от 1 до 7 — выдает в качестве результата количество занятий в вашей группе в соответствующий день.
29) В зависимости от стажа работы педагогам введена надбавка в размере: для работающих от 5 до 10 лет — 10%; для работающих от 10 до 15 лет — 15%; для работающих свыше 15 лет — 20%. Составьте программу, которая по заданному стажу работы и размеру оклада определит размер заработной платы.
30) Составить программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 999), обозначающего денежную величину, дописывает слово «рубль» в правильной форме. Например, 5 рублей, 21 рубль, 173 рубля.

Цикл while.

Инструкция **`while`** является универсальным способом организации цикла по условию в языке Python. Она выполняет набор инструкций (тело цикла, обычно выделяемое отступами) до тех пор, пока условное выражение продолжает возвращать истину. Самая простая форма такого цикла имеет вид:

```
while (условие):          # двоеточие после условия
набор инструкций     # тело цикла
```

Интерпретатор продолжает вычислять условное выражение (условие) в строке заголовка, и снова и снова выполнять вложенные инструкции в теле цикла, пока условное выражение не вернет «ложь». Допустимо, но не рекомендуется, записывать весь цикл в одну строку.

```
x = 1
while x<=5:
print(x)
x = x+1
```

```
x = 1
while x<=5: print(x); x = x+1
```

В первом способе, поскольку и `print(x)` и `x=x+1` написаны с одинаковым отступом, они считаются телом цикла. В своей наиболее общей форме инструкция `while` состоит из строки заголовка с условным выражением, тела цикла, содержащего одну или более инструкций с

отступами, и необязательной части `else`, которая выполняется, когда управление передается за пределы цикла без использования инструкции `break`.

```
while <test>:                  # Условное выражение test
    <statements1>              # Тело цикла
else:                          # Необязательная часть else
    <statements2>
```

Блок `else` – Выполняется, только если цикл завершился обычным образом (без использования инструкции break). Дополнительные инструкции **break, continue, pass:**

`break` – Производит переход за пределы всей инструкции цикла.

`continue` – Переход в начало цикла (в строку заголовка).

`pass` – Ничего не делает: это пустая инструкция.

Пример, выход через `break` из тела бесконечного цикла:

```
x=5
while True: # условие выполнено всегда
    x=x+1
    print(x, end=' ')
    if x==10: break # выход из цикла
```

В результате мы получим набор чисел: `6 7 8 9 10.`

Или например, ели надо определить, количество цифр натурального числа n,

```
n = int(input('введи натуральное число'))
lenght = 0
while n>0:
    n//=10
    lenght+=1
print('length=', lenght)
```

Цикл while используется тогда, когда мы заранее не знаем, сколько раз должны повторять некоторое действие.

В Python используются стандартные операции сравнения, которые также имеют общепринятый смысл:

==	Тождественно равно	5 == 5; в результате будет True True == False; в результате будет False «abc» == «abc»; в результате будет True
!=	Не равно	12 != 5; в результате будет True
==	Тождественно равно	5 == 5; в результате будет True True == False; в результате будет False «abc» == «abc»; в результате будет True

<>	Не равно	12 <> 5 в результате будет True.
> , >=	Больше, больше или равно	
< , <=	Меньше, меньше или равно	

Цикл for в Python

Цикл `for` является универсальным итератором последовательностей в языке Python: он может выполнять обход элементов в любых упорядоченных последовательностях (списках)

```
for переменная in список: #двоеточие обязательно
        тело цикла
```

Циклы **`for`** начинаются со строки заголовка, где указывается переменная для присваивания («пробегающая» список), а также сам список, то есть объект, обход которого будет выполнен. Самый простой список – это упорядоченная последовательность чисел. Вслед за заголовком следует тело цикла – блок инструкций (обычно с отступами), которые требуется выполнить Для создания списков широко используется функция **`range`**, являющейся одной из встроенных функций Python.

Инструкция вида

```
range (N1, N2 [, Step])
```

создает упорядоченную последовательность целых чисел, от значения **`N1`** до **`N2`**, с заданным шагом `Step` (по умолчанию `Step` =1). В частности, инструкция `range(N)` создает список [0, 1, … , N –1]. Ниже приводится несколько вариантов её применения:

```
range(2, 12)
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> range(2, 12, 3)
[2, 5, 8, 11]
>>> range(11, 2, -2)
[11, 9, 7, 5, 3]
```

Таким образом. типичный цикл с целочисленным счетчиком цикла имеет вид

```
for X in range (N1, N2):
    тело цикла
```

В качестве примера рассмотрим вычисление чисел Фибоначчи: $F(0) = 1$, $F(1) = 1$, $F(n) = F(n–1) + F(n–2)$. Программа содержит всего 2 строчки (оцените минимализм Python)

```
X,Y,N = 1, 1, 5
for i in range(N+1): X,Y=Y,X+Y; print(i,X)
```

Лабораторная работа № 3
Циклы while и for в Python

Цель занятия: получение теоретических знаний и практических навыков работы с циклами на Python.

Задачи к лабораторной работе № 3

1) Даны целые числа K и N ($N > 0$). Вывести N раз число K.
2) Даны два целых числа A и B ($A < B$). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.
3) Даны два целых числа A и B ($A < B$). Вывести в порядке убывания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.
4) Одна штука некоторого товара стоит 20,4 руб. Напечатать таблицу стоимости для 1, 2, …, 10 штук этого товара.
5) Напечатать квадраты всех целых чисел от A до B ($A < B$) с шагом H.
6) Напечатать все положительные числа из диапазона от A до B ($A < B$) с шагом H.
7) Даны два целых числа A и B ($A < B$). Найти сумму всех целых чисел от A до B включительно.
8) Даны два целых числа A и B ($A < B$). Найти произведение всех целых чисел от A до B включительно.
9) Даны два целых числа A и B ($A < B$). Найти сумму квадратов всех целых чисел от A до B включительно.
10) Дано положительное вещественное число — цена 1 кг конфет. Вывести стоимость 1.2, 1.4, …, 2 кг конфет.
11) Дано целое число N (> 0). Найти квадрат данного числа, используя для его вычисления следующую формулу: $N^2 = 1 + 3 + 5 + \cdots + (2 \cdot N - 1)$.

12) Дано вещественное число A и целое число N (> 0). Вывести все целые степени числа A от 1 до N.

13) Дано целое число N (> 0). Найти наибольшее целое число K, квадрат которого не превосходит N: $K^2 \le N$.

14) Дано целое число N (>1). Найти наибольшее целое число K , при котором выполняется неравенство $3^K < N$.

15) Дано целое число N (>1) и две вещественные точки на числовой оси: A, B ($A < B$). Отрезок $[A,B]$ разбит на N равных отрезков. Вывести H — длину каждого отрезка, а также набор точек A, $A+H$, $A+2H$, $A+3H$, …, B, образующий разбиение отрезка $[A,B]$.

Математические функции. Модули math и random

Для того, чтобы использовать в вычислениях стандартные математические функции (тангенс, логарифм и т.д.) необходимо выполнить подключение специальной библиотеки таких функций, которая называется модулем. Это модуль входит в так называемую коллекцию «**Модули стандартной библиотеки**», которая обеспечивает поддержку распространенных задач программирования. Она поставляется со стандартным пакетом Python и является платформо-независимой. **Модули подключаются специальной инструкцией** `import`, которую желательно располагать в самом верху программы. Например, инструкция `import math` подключает стандартный математический пакет. После подключения программа получает доступ ко всем функциям, методам и классам, содержащимся в нём. Программист может вызвать любую функцию из подключенной библиотеки, используя перед именем функции префикс с именем модуля. Пример:

```
>>> import math
>>> math.pi
3.141592653589793
```

Этот способ не допускает пересечения имён, то есть можно использовать одно и то же имя функции в программе и в подключаемой библиотеке, и не бояться, что после её подключения функция будет переопределена.

Использование псевдонимов. Если название модуля слишком длинное, или по каким-то другим причинам, то для него можно создать «псевдоним», с помощью ключевого слова `as`.

```
>>> import math as m
>>> m.e 2.718281828459045
```

Теперь доступ ко всем атрибутам модуля math осуществляется только с помощью префикса m. Однако если после этого снова написать `import math`, тогда функции модуля будут доступны как под именем m, так и под именем math.

Инструкция `from`. При необходимости, можно подключить не весь модуль, а только его часть, воспользовавшись инструкцией `from`. Например, требуется использовать только пару функций из библиотеки `math`, а вся библиотека содержит более 40 функций, которые будут напрасно загружать память. Поэтому используют вариант с частичным подключением функций модуля `from` <имя модуля> `import` <список функций>

Например:

```
>>> from math import pi,e
>>> pi
    3.141592653589793
>>> e
2.718281828459045
```

Если после инструкции `import` написать символ звёздочки (*), то есть написать, например, `from math import *` , подключится все содержимое модуля (**при этом префикс не требуется**). Импорт через `from` требует аккуратности – в программе не должно быть идентификаторов, совпадающих с именем одной из импортируемых функций или констант, иначе возникнет «путаница». **Модуль `math`** содержит расширенный набор основных математических функций. В таблице ниже приведен список имен наиболее популярных из таких функций, смысл которых достаточно очевиден из их названия. При этом аргументы тригонометрических функций, естественно, берутся в радианах, `sqrt`, как всегда, – корень квадратный.

`cos(X)`	`acos(X)`	`fabs(X)`	`e`	`log(X)`
`sin(X)`	`asin(X)`	`degrees(X)`	`exp(X)`	`log10()`
`tan(X)`	`atan(X)`	`radians(X)`	`cosh(X)`	`log2(X)`
`pi`	`pow(X, n)`	`factorial(n)`	`sinh(X)`	`log(X, b)`

Список всех функций модуля можно получить через инструкцию **`help`**,в данном случае через `help(math)` или `help("math")`. Также можно получить и справочную информацию о конкретной функции, например: `help(math.sin)` или `help('math.sin')`. Кроме того, в модуле имеется набор функций для преобразования чисел и осуществления над ними математических операций, в частности, `ceil()`– округление числа «вверх», `floor()` – округление числа «вниз» и т.д.

Ниже приводятся примеры использования функций модуля при различных вариантах его подключения.

```
import math
# возведение 2 в степень 3
n = math.pow(2, 3)
print(n) #ответ 8
# косинус 60 градусов
x=math.cos(math.radians(60))
print(x)
```

```
from math import *
# возведение 2 в степень 3
n = pow(2, 3)
print(n) #ответ 8
# косинус 60 градусов
x = cos(radians(60))
print(x)
```

Модуль `random` содержит различные варианты генератора случайных чисел (ГСЧ). Простой ГСЧ при каждом обращении возвращает некоторое число, выбранное случайным образом из заданного множества чисел. Вероятность появления конкретного числа определяется типом ГСЧ. Например, для задачи типа «орел - решка» генератор при каждом вызове должен с одинаковой вероятностью выбрасывать либо ноль, либо единицу. В языках высокого уровня ГСЧ обозначается словом random (*англ.* случайный), или аббревиатурой от этого слова. В частности, случайное число можно `X` можно получить с помощью одного из вариантов кода:

```
mport random
X=random.random()
print(X);
```

```
from random import *
X= random()
print(X)
```

В таблице ниже приведен список наиболее популярных функций, модуля `random`. Список всех функций модуля можно получить с помощью команды `dir(random)`

`choice(A)`	возвращает случайный элемент из списка A
`random()`	возвращает вещественное случайное число, равномерно распределенное на отрезке [0, 1]
`uniform(a,b)`	то же, но для отрезка [a, b]
`randint(n,m)`	возвращает случайное целое число в диапазоне от n до m
`gauss(x,Dx)`	возвращает вещественное случайное число, распределенное по Гауссу; x – среднее значение, Dx – стандартное отклонение
`shuffle(A)`	случайно перемешивает значения в списке A
`sample(A,n)`	возвращает случайную последовательность длиной n из набора A
`seed(z)`	инициализация ГСЧ. Если z не указано, используется системное время.

Пример случайного выбора 5 чисел из набора A:

```
>>> import random
>>> A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(random.sample(A, 5))
[3, 1, 7, 6, 2]
```

Комментарии: Синонимом ГСЧ с Гауссовым распределением является функция `normalvariate(x,Dx)` (нормальное распределение). С помощью `random.seed(…)` можно запускать новую или повторять старую случайную последовательность:

```
from random import *
seed(1)
for i in range(10):
    print(randint(1,20),' ', end='')
print( end='\n')
for i in range(10):
    print(randint(1,20),' ', end='')
print( end='\n')
seed(1)
for i in range(10):
    print(randint(1,20),' ', end='')
print(end='\n')
seed(1)
```

В результате получим:

```
5  19  3  9  4  16  15  16  13  7
4  16  1  13  14  20  1  15  9  8
5  19  3  9  4  16  15  16  13  7
```

Списки, кортежи, множества, словари

Работа с одномерными списками. Создание списка. Операции над списками. Методы списков

Список представляет собой последовательность элементов, пронумерованных от 0, как символы в строке. Для создания списка используется конструктор list. Создание списков осуществляется двумя способами.

Явно указав все элементы списка в тексте (элементы списка перечисляются в квадратных скобках через запятую):

```
# пустой список
lst=[]

# список, состоящий из чисел
Primes = [2, 3, 5, 7, 11, 13]

# список, состоящий из элементов разных типов
lst1= [111, 'Hello', '222', True]

#список, состоящий из строк
Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']
```

И с помощью функции list(),

```
lst = list()
lst1 = list('Hello')
lst2 = list(range(10,15))
print(lst, lst1, lst2, sep='\n')
```

```
[]
['H', 'e', 'l', 'l', 'o']
[10, 11, 12, 13, 14]
```

Функция list**()** используется для преобразования итерируемых последовательностей в списки, например:

```
>>> list('abc') #выведет['a', 'b', 'c'].
>>> list(range(6)) #выведет [0, 1, 2, 3, 4, 5]
```

Напомним, что функция `range(N, M [,h])` создает упорядоченную последовательность целых чисел, от значения `N` до `M-1`, с шагом `h`. Функция `sorted()` возвращает новый список, отсортированный по не убыванию (`sorted([8, 1, 3])=[1, 3, 8]`).

Аналогично строкам, доступ к элементам списка можно получить по их индексам, а также можно извлекать срезы. Длина списка определяется функцией `len()`.Например:

```
>>> X=['A', 'Б', 20, 1.8, [1, 0]] >>> len(X) #выведет 5
>>> X[1] #выведет 'Б'
>>> X[ : 2] #выведет ['A', 'Б']
>>> X[ -1] #выведет [1, 0]
```

Поэлементное заполнение списка, используя метод append(), который добавляет в конец списка указанное значение

```
A = []
for i in range(int(input('Введите количество элементов: '))):
    A.append(int(input('Введите элемент: ')))

print(f'Список: {A}')
```

```
Введите количество элементов: 4
Введите элемент: 5
Введите элемент: -7
Введите элемент: 42
Введите элемент: -67
Список: [5, -7, 42, -67]
```

В этом примере создается пустой список, далее считывается количество элементов в списке, затем по одному считываются элементы списка и добавляются в его конец. Операции над списками:

- Конкатенация списков (добавление одного списка в конец другого).
- Повторение списков (умножение списка на число), повторяет список указанное количеств раз. Например:

```
A = [1, 2, 3]
B = [4, 5]
C = A + B
D = B * 3
print ('C =',C)
print ('D =',D)
```

```
C = [1, 2, 3, 4, 5]
D = [4, 5, 4, 5, 4, 5]
```

Проверка на вхождение (in, not in). Оператор in проверяет наличие значения в списке, оператор not in — отсутствие значения. Возвращают значения True или False. Извлечение среза. Со списками, также как и со строками, можно делать срезы. А именно:

A[i : j] — срез из j – i элементов A[i], A[i + 1], …, A[j – 1].

A[i : j : –1] — срез из i – j элементов A[i], A[i-1], …, A[j + 1] (то есть меняется порядок элементов).

A[i : j : k] — срез с шагом k: A[i], A[i + k], A[i + 2*k], … . Если значение k<0, то элементы идут в противоположном порядке. Каждое из чисел i или j может отсутствовать, что означает «начало строки» или «конец строки».

Операции со списками в Python:

x in A — проверить, содержится ли элемент в списке. Возвращает True или False.

x not in A — то же самое, что not(x in A).

min(A) — наименьший элемент списка.

max(A) — наибольший элемент списка.

A.index(x) — индекс первого вхождения элемента *x* в список, при его отсутствии генерирует исключение ValueError.

A.count(x) — количество вхождений элемента *x* в список.

A.sort() — сортировка списка (меняет сам список, ничего не возвращает).
sum(A) — возвращает сумму элементов в списке.
A.append(x) — добавить элемент *x* в конец списка.
A.extend(L) — добавить все элементы списка *L* в конец списка *A*.
Синтаксис генераторов списков в Python:

```
[выражение for переменная in список],
```

где переменная — идентификатор некоторой переменной, список — список значений, который принимает данная переменная (как правило, полученный при помощи функции
range), выражение — некоторое выражение, которым будут заполнены элементы списка, как правило, зависящее от использованной в генераторе переменной.

Например, Создать список, заполненный квадратами целых чисел от 0 до *n* и от 1 до *n* + 1

```
n = 6
A = [i ** 2 for i in range(n)]
A1 = [i ** 2 for i in range(1, n + 1)]
print('A =', A)
print('A1 =', A1)
```

```
A = [0, 1, 4, 9, 16, 25]
A1 = [1, 4, 9, 16, 25, 36]
```

Пример. Создать список из строк, считанных со стандартного ввода: сначала нужно ввести число элементов списка (это значение будет использовано в качестве аргумента функции range), потом — заданное количество строк.

```
lst = [input('Введите элемент: ') for i in range(int(input('Введите количество элементов: ')))]
print('lst =', lst)
```

```
Введите количество элементов: 5
Введите элемент: q1
Введите элемент: 4w5
Введите элемент: 561
Введите элемент: 12.3
Введите элемент: hello
lst = ['q1', '4w5', '561', '12.3', 'hello']
```

Метод строки split() возвращает список строк, разрезав исходную строку на части по пробелам по умолчанию: A = input().split(). Если при запуске этой программы ввести строку 1 2 3, то список A будет равен ['1 ', '2 ', '3 ']. У метода split есть необязательный параметр, который определяет, какая строка будет использоваться в качестве разделителя между элементами списка. Например, метод split(';') вернет список, полученный разрезанием исходной строки по символам '; '. Метод строки join(lst) выводит список при помощи

однострочной команды. У этого метода один параметр: список строк. В результате получается строка, полученная соединением элементов списка (которые переданы в качестве параметра) в одну строку, при этом между элементами списка вставляется разделитель, равный той строке, к которой применяется метод

Многомерные списки: работа с двумерными массивами

В языке программирования Python таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, создать числовую таблицу из двух строк и трех столбцов можно так: A = [[1, 2, 3], [4, 5, 6]]. Здесь первая строка списка A[0] является списком из чисел [1, 2, 3]. То есть A[0][0] == 1, значение A[0][1] == 2, A[0][2] == 3, A[1][0] == 4, A[1][1] == 5, A[1][2] == 6.

Для обработки и вывода списка, как правило, используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
for i in range(len(A)):
    for j in range(len(A[i])):
        print(A[i][j], end=' ')
    print()
```

```
1 2 3
4 5 6
```

То же самое, но циклы не по индексу, а по значениям списка:

```
for row in A:
    for elem in row:
        print(elem, end=' ')
    print()
```

```
1 2 3
4 5 6
```

Для вывода одной строки можно воспользоваться методом join:

```
for row in A:
    print(' '.join(list(map(str, row))))
```

```
1 2 3
4 5 6
```

Лабораторная работа №4
Списки. Двумерные списки.

Цель занятия: получение теоретических знаний и практических навыков работы со списками на Python.

Задачи к лабораторной работе № 4.1

1) Дан список, включающий N элементов, и целые числа K и L ($1 \le K \le L \le N$). Найти среднее арифметическое элементов списка с порядковыми номерами от K до L включительно.
2) Дан список, включающий N элементов, и целые числа K и L ($1 < K \le L \le N$). Найти сумму всех элементов списка, кроме элементов с порядковыми номерами от K до L включительно.
3) Дан список, включающий N элементов, и целые числа K и L ($1 < K \le L \le N$). Найти среднее арифметическое всех элементов списка, кроме элементов с порядковыми номерами от K до L включительно.
4) Дан список. Вывести вначале все содержащиеся в данном списке четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов.
5) Дан список. Вывести вначале его элементы с четными номерами (в порядке возрастания номеров), а затем — элементы с нечетными номерами (также в порядке возрастания номеров).
6) Дан список. Найти произведение всех элементов списка, расположенных справа от максимального элементами, не включая максимальный элемент.
7) Дан список. Найти сумму всех элементов списка, расположенных между его минимальным и максимальным элементами, включая минимальный и максимальный элементы.
8) Дан список. Обнулить элементы списка, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).
9) Дан список. Увеличить все четные числа, содержащиеся в списке, на заданное число. Если четные числа в списке отсутствуют, то оставить список без изменений.
10) Дан список. Обнулить все нечетные числа, содержащиеся в списке. Если нечетные числа в списке отсутствуют, то оставить список без изменений.
11) Дан список. Найти минимальный элемент из его элементов с четными номерами.
12) Дан список. Найти номера тех элементов списка, которые больше своего правого соседа, и количество таких элементов. Найденные номера выводить в порядке их возрастания.
13) Дан список. Найти два соседних элемента, сумма которых максимальна, и вывести

эти элементы в порядке возрастания их индексов.

14) Дан список. Поменять местами его минимальный и максимальный элементы.
15) Даны целые числа N (> 2), A и B. Сформировать и вывести список размера N, первый элемент которого равен A, второй равен B, а каждый последующий элемент равен сумме всех предыдущих.
16) Дан список, включающий N элементов, и целые числа K и L ($1 \le K \le L \le N$). Найти среднее арифметическое всех четных элементов списка с порядковыми номерами от K до L включительно.
17) Дан список, включающий N элементов, и целые числа K и L ($1 < K \le L \le N$). Найти сумму всех нечетных элементов списка, кроме элементов с порядковыми номерами от K до L включительно.
18) Дан список, включающий N элементов, и целые числа K и L ($1 < K \le L \le N$). Найти среднее арифметическое всех нечетных элементов списка, кроме элементов с порядковыми номерами от K до L включительно/
19) Дан список. Вывести вначале два числа с максимальными значениями в порядке убывания их индексов, а затем — два числа с минимальными значениями в порядке убывания их индексов.
20) Дан список. Вывести вначале его элементы с нечетными номерами (в порядке возрастания номеров), а затем — элементы с четными номерами (также в порядке возрастания номеров).
21) Дан список. Найти произведение всех элементов списка, расположенных справа от минимального элемента, не включая минимальный элемент.
22) Дан список. Найти произведение всех элементов списка, расположенных между его минимальным и максимальным элементами, включая минимальный и максимальный элементы.
23) Дан список. Увеличить в два раза все элементы списка, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).
24) Дан список. Увеличить все нечетные числа, содержащиеся в списке, на заданное число. Если нечетные числа в списке отсутствуют, то оставить список без изменений.
25) Дан список. Обнулить все четные числа, содержащиеся в списке. Если четные числа в списке отсутствуют, то оставить список без изменений.
26) Дан список. Найти максимальный элемент из его элементов с нечетными номерами.

27) Дан список. Найти количество пар элементов списка, которые являются взаимно противоположными.
28) Дан список. Найти два соседних элемента, сумма которых нулю, и вывести эти элементы в порядке возрастания их индексов.
29) Дан список. Получить список, в котором не будет повторяющихся элементов..
30) Даны целые числа N (> 2), A и B. Сформировать и вывести список размера N, первый элемент которого равен A, а каждый последующий элемент увеличивается на B относительно предыдущего.

Задачи к лабораторной работе № 4.2.Сортировка списков

1) Дан список. Выбрать из списка все нечетные числа и упорядочить их по убыванию.
2) Дан список. Выбрать из списка все положительные числа и упорядочить их по возрастанию.
3) Дан список. Выбрать из списка все положительные числа и упорядочить их по убыванию.
4) Дан список. Выбрать из списка все четные числа и упорядочить их по возрастанию.
5) Дан список. Выбрать из списка все числа больше заданного числа k и упорядочить их по убыванию.
6) Дан список. Выбрать из списка все числа больше 10 и упорядочить их по возрастанию.
7) Дан список. Выбрать из списка все числа кратные 5 и упорядочить их по убыванию.
8) Дан список. Выбрать из списка все числа меньше заданного числа k и упорядочить их по возрастанию.
9) Дан список. Выбрать из списка все числа меньше 15 и упорядочить их по убыванию.
10) Дан список. Выбрать из списка все числа кратные 3 и упорядочить их по возрастанию.
11) Дан список. Выбрать из списка все числа кратные заданному числу k и упорядочить их по убыванию.
12) Дан список. Выбрать из списка все отрицательные числа и упорядочить их по возрастанию.
13) Дан список. Выбрать из списка все числа на нечетных позициях и упорядочить эти числа по убыванию.
14) Дан список. Выбрать из списка все двузначные числа и упорядочить эти числа по

возрастанию.

15) Дан список. Выбрать из списка все числа на четных позициях и упорядочить эти числа по возрастанию.
16) Дан список. Выбрать из списка все двузначные числа и упорядочить эти числа по возрастанию их суммы цифр.
17) Дан список. Выбрать из списка все двузначные числа и упорядочить эти числа по возрастанию последней цифры.
18) Дан список, состоящий из трехзначных чисел. Выполнить сортировку элементов списка по убыванию суммы цифр элементов списка.
19) Дан список, состоящий из трехзначных чисел. Выполнить сортировку элементов списка по возрастанию суммы цифр элементов списка.
20) Дан список, состоящий из двузначных чисел. Выполнить сортировку элементов списка по убыванию остатков от деления на число k.
21) Дан список строк. Отсортируйте его в алфавитном порядке.
22) Дан список строк. Отсортируйте его по длине строк — от самых коротких к самым длинным.
23) Дан список строк. Отсортируйте список в порядке убывания алфавитного порядка, но игнорируйте регистр букв.
24) Дан список чисел. Отсортируйте список, переместив все отрицательные числа в начало списка, а положительные — в конец.
25) Дан список строк. Отсортируйте список в порядке убывания алфавитного порядка по первой букве каждой строки, регистр букв учитывайте.
26) Дан список строк. Отсортируйте список в порядке количества символов в элементах строки.
27) Дан список. Выбрать из списка все двузначные числа и упорядочить эти числа по возрастанию первой цифры.
28) Дан список. Выбрать из списка все двузначные числа и упорядочить эти числа по убыванию первой цифры.

Задачи к лабораторной работе № 4.3.Вложенные списки

1) Дана целочисленная матрица размера $M \times N$. Найти номер первого из ее столбцов, содержащих только нечетные числа. Если таких столбцов нет, то вывести 0.
2) Дана матрица размера $M \times N$. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждой строке.
3) Дана матрица размера $M \times N$. Поменять местами строки, содержащие минимальный

и максимальный элементы матрицы.

4) Дана матрица размера $M \times N$. Зеркально отразить ее элементы относительно горизонтальной оси симметрии матрицы (при этом поменяются местами строки с номерами 1 и M, 2 и M — 1 и т. д.).
5) Дана матрица размера $M \times N$. Удалить строку, содержащую минимальный элемент матрицы.
6) Дана матрица размера $M \times N$. Удалить столбец, содержащий максимальный элемент матрицы.
7) Дана матрица размера $M \times N$. Зеркально отразить ее элементы относительно вертикальной оси симметрии матрицы (при этом поменяются местами столбцы с номерами 1 и N, 2 и N — 1 и т. д.).
8) Дана матрица размера $M \times N$. Продублировать строку матрицы, содержащую ее максимальный элемент.
9) Дана матрица размера $M \times N$. В каждом ее столбце найти количество элементов, больших среднего арифметического всех элементов этого столбца.
10) Дана матрица размера $M \times N$ (M и N — четные числа). Поменять местами левую верхнюю и правую нижнюю четверти матрицы.
11) Дана матрица размера $M \times N$. Для каждой строки матрицы с нечетным номером (1, 3, …) найти среднее арифметическое ее элементов.
12) Дана матрица размера $M \times N$. Найти номер ее строки с наибольшей суммой элементов и вывести данный номер, а также значение наибольшей суммы.
13) Дана матрица размера $M \times N$. Перед строкой матрицы, содержащей минимальный элемент матрицы, вставить строку из нулей.
14) Дана целочисленная матрица размера $M \times N$. Найти номер последней из ее строк, содержащих только четные числа. Если таких строк нет, то вывести 0.
15) Дана матрица размера $M \times N$. Поменять местами столбец с номером 1 и последний из столбцов, содержащих только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.
16) Дана матрица размера $M \times N$. Найти номер первого столбца, содержащего максимальное количество отрицательных элементов. Если таких столбцов нет, вывести 0.
17) Дана матрица размера $M \times N$. Найти номер последнего столбца, содержащего только нулевые элементы. Если таких столбцов нет, вывести 0.
18) Дана матрица размера $M \times N$. Найти номер последней из ее строк, содержащих только положительные элементы. Если таких строк нет, вывести 0.

19) Дана матрица размера $M \times N$. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждом столбце.
20) Дана матрица размера $M \times N$. Удалить столбец, содержащий минимальный элемент матрицы.
21) Дана матрица размера $M \times N$. Зеркально отразить ее элементы относительно главной диагонали (при этом поменяются местами строки и столбцы).
22) Дана матрица размера $M \times N$. Продублировать столбец матрицы, содержащий ее минимальный элемент.
23) Дана матрица размера $M \times N$. В каждой ее строке найти количество элементов, меньших среднего арифметического всех элементов этой строки.
24) Дана матрица размера $M \times N$. Найти номер ее столбца с наименьшей суммой элементов и вывести данный номер, а также значение наименьшей суммы.
25) Дана матрица размера $M \times N$. Перед столбцом матрицы, содержащим максимальное значение элемента матрицы, вставить столбец из нулей.
26) Дана матрица размера $M \times N$. Найти сумму элементов каждой ее строки и вывести номера строк, сумма элементов которых является простым числом.
27) Дана матрица размера $M \times N$. Заменить все отрицательные элементы матрицы их модулем.
28) Дана матрица размера $M \times N$. Заменить все элементы матрицы, которые превышают среднее арифметическое всех элементов матрицы, их квадратом.
29) Дана матрица размера $M \times N$. Найти номер последней из ее строк, содержащей только отрицательные элементы. Если таких строк нет, вывести 0.
30) Дана матрица размера $M \times N$. Поменять местами столбцы с наименьшим и наибольшим количеством отрицательных элементов. Если таких столбцов нет, вывести матрицу без изменений.

Лабораторная работа № 5
Кортежи. Работа с кортежами.

Цель занятия: получение теоретических знаний и практических навыков работы с кортежами на Python.

Кортеж (Tuple) – это упорядоченная, **неизменяемая** коллекция объектов произвольных типов, поддерживающая произвольное число уровней вложенности. Кортеж использует меньше памяти, чем список. При задании кортежа вместо квадратных скобок используются круглые. Примеры кортежей: `(1,2,3), ('aaa', -25, 3), ('x', -1, [1, 2, 3])`. Функцией `list` можно преобразовать кортеж в список:

```
>>> list((1,2,3))
[1, 2, 3]
```

Важно помнить, что кортеж не допускает изменений, в него нельзя добавить новый элемент, удалить или заменить существующие элементы.

С помощью оператора присваивания можно последовательно извлекать элементы кортежа

```
Vector = ("x", -1, [1,2,3])
x,y,z=(1, 2, Vector)
print(x,y,z)  #выведет 1  2  ('x', -1, [1, 2, 3])
```

Важно согласовать набор переменных слева от символа (=) с элементами кортежа. Допустимо использовать символ звездочка (*), чтобы присвоить набор элементов кортежа одной переменной

```
a,*b=(1, 2, Vector)
print(a,b) #выведет 1 [2, ('x', -1, [1, 2, 3])]
```

Выводы:

1. Кортеж определяется так же, как список, за исключением того, что набор элементов заключается в круглые скобки, а не в квадратные.
2. Элементы кортежа заданы в определённом порядке, как и в списке. Элементы кортежа индексируются с нуля, как и элементы списка, таким образом, первый элемент не пустого кортежа – это всегда a_tuple[0].
3. Отрицательные значения индекса отсчитываются от конца кортежа, как и в списке. Последний элемент имеет индекс –1.
4. Создание среза кортежа («slicing») аналогично созданию среза списка. Когда создаётся срез списка, получается новый список; когда создаётся срез кортежа, получается новый кортеж.

Особенности кортежа:

- ✓ Нельзя добавить элементы к кортежу. Кортежи не имеют методов append() или extend().
- ✓ Нельзя удалять элементы из кортежа. Кортежи не имеют методов remove() или pop().
- ✓ Можно искать элементы в кортежи, поскольку это не изменяет кортеж.
- ✓ Можно использовать оператор in, чтобы проверить существует ли элемент в кортеже.

✓ Встроенная функция tuple() принимает список и возвращает кортеж из всех его элементов, функция list() принимает кортеж и возвращает список.Кортежи в логическом контексте можно использовать в операторе if.
✓ В логическом контексте пустой кортеж является ложью.
✓ Любой кортеж, состоящий, по крайней мере, из одного элемента, - истина.
✓ Любой кортеж, состоящий, по крайней мере, из одного элемента, — истина. Значения элементов не важны.
✓ Чтобы создать кортеж из одного элемента, необходимо после него поставить запятую. Без запятой Python предполагает, что вы просто добавили еще одну пару скобок, что не делает ничего плохого, но и не создает кортеж.

Задачи к лабораторной работе №5

1) Напишите программу: а) для создания кортежа; б) для преобразования кортежа в словарь.
2) Напишите программу: а) для создания кортежа с различными типами данных; б) для распаковки списка кортежей в отдельные списки.
3) Напишите программу: а) для создания кортежа с числами и выведите на экран первый элемент; б) для того, чтобы перевернуть элементы кортежа в обратном порядке.
4) Напишите программу: а) для распаковки кортежа в несколько переменных; б) для преобразования списка кортежей в словарь.
5) Напишите программу: а) для добавления элемента в кортеж; б) для печати кортежа со строковым форматированием. Например, задан кортеж: (100, 200, 300), на экран вывести: это кортеж (100, 200, 300).
6) Напишите программу: а) для преобразования кортежа в строку; б) для замены последнего элемента в каждом кортеже в списке кортежей на заданное значение.
7) Напишите программу: а) чтобы получить четвёртый элемент кортежа с начала и четвёртый элемент с конца; б) для удаления пустого кортежа (или кортежей) из списка кортежей.
8) Напишите программу: а) для создания кортежа словарей; б) для сортировки кортежа по его элементу, являющегося числом с плавающей точкой.
9) Напишите программу: а) для поиска повторяющихся элементов кортежа; б) для подсчёта элементов в списке до тех пор, пока элемент не будет кортежем.
10) Напишите программу: а) чтобы проверить, включает ли кортеж заданный; б) преобразующую заданный список строк в кортеж.
11) Напишите программу: а) для создания кортежа из списка, исключая повторяющиеся элементы; б) для удаления элемента с заданным значением из кортежа.
12) Напишите программу: а) для получения уникальных элементов двух кортежей; б) для объединения двух кортежей в один.

13) Напишите программу: а) для проверки, является ли кортеж пустым; б) для нахождения количества вхождений элемента в кортеж.
14) Напишите программу: а) для создания списка кортежей из заданного списка; б) для сортировки списка кортежей по последнему элементу в каждом кортеже.
15) Напишите программу: а) для получения индекса заданного значения в кортеже; б) для создания нового кортежа, содержащего только четные элементы из заданного кортежа.
16) Напишите программу: а) для создания кортежа из строки, содержащей числа, разделенные запятой; б) для сортировки кортежа по длине строк в нем
17) Напишите программу: а) для создания кортежа из списка и его сортировки в обратном порядке; б) для преобразования строки в кортеж, разбивая ее на слова.
18) Напишите программу: а) для создания кортежа из заданного списка, исключая повторяющиеся элементы; б) для объединения двух кортежей, исключая повторяющиеся элементы.
19) Напишите программу: а) для сравнения двух кортежей на равенство; б) для объединения заданного кортежа с элементами другого кортежа.
20) Напишите программу: а) для создания кортежа из заданного списка чисел и его сортировки; б) для замены всех вхождений заданного элемента в кортеже на новое значение.
21) Напишите программу: а) для удаления заданного элемента из кортежа; б) для получения подкортежа из заданного кортежа, указывая начальный и конечный индексы.
22) Напишите программу: а) для создания кортежа из списка, исключая элементы, не являющиеся числами; б) для получения списка ключей и значений из словаря, а затем создания кортежа из этих списков.
23) Напишите программу: а) для создания кортежа из списка чисел и его сортировки по убыванию; б) для вывода на экран элементов кортежа в столбик.
24) Напишите программу: а) для создания кортежа из заданного списка строк и его сортировки по алфавиту; б) для нахождения максимального и минимального элементов в кортеже.
25) Напишите программу: а) для получения первых трех элементов кортежа; б) для вывода на экран уникальных значений из кортежа.
26) Напишите программу: а) для проверки, является ли кортеж пустым; б) для подсчета количества вхождений каждого элемента в кортеже.
27) Напишите программу: а) для получения среза из кортежа, начиная с третьего элемента и заканчивая предпоследним; б) для конкатенации двух кортежей.
28) Напишите программу: а) для создания кортежа из заданного списка, заменяя каждое четное значение на ноль; б) для проверки, все ли элементы кортежа удовлетворяют заданному условию.

29) Напишите программу: а) для получения пересечения двух кортежей; б) для проверки, есть ли заданный элемент в кортеже.
30) Напишите программу: а) для создания кортежа, содержащего элементы кортежей из заданного списка; б) для проверки, является ли кортеж симметричным, т.е. совпадают ли его элементы при обратном порядке.

Лабораторная работа №6
Множества. Работа с множествами.

Цель занятия: получение теоретических знаний и практических навыков работы с множествами на Python.

Множество – это неупорядоченная коллекция уникальных элементов, с которой можно сравнивать другие элементы, чтобы определить, принадлежат ли они этому множеству. Множество может состоять из различных элементов, порядок элементов во множестве не определен. Элементами множества может быть любой неизменяемый тип данных: числа, строки, кортежи. Изменяемые типы данных не могут быть элементами множества, в частности, нельзя сделать элементом множества список (но можно сделать кортеж) или другое множество. Каждый элемент может входить во множество только один раз, порядок задания элементов не важен. Задание множеств можно реализовать двумя способами:

Множество задается перечислением всех его элементов в фигурных скобках.

```
A = {1, 2, 3}
```

с помощью метода (функции) set()

```
# пустое множество
s1 = set()

# преобразуем строку в множество
s2 = set('asdfghj')

# преобразуем список в множество
s3 = set ([1, 2, 3, 1, 2, 3])

# преобразуем словарь в множество
s4 = set({'a': 1, 'b': 2})

print(s1, s2, s3, s4, sep='\n')
```

```
set()
{'a', 's', 'g', 'f', 'd', 'h', 'j'}
{1, 2, 3}
{'a', 'b'}
```

Сформировать множество из строки можно следующим образом:

```
A = set('qwerty')
print(A)
```

```
{'e', 'q', 'r', 't', 'y', 'w'}
```

Операции над множествами:

✓ Чтобы проверить, принадлежит ли значение множеству, используйте оператор in. Он работает так же, как и для списков.

✓ Метод union() (объединение (|)) возвращает новое множество, содержащее все элементы каждого из множеств.

✓ Метод intersection() (пересечение (&)) возвращает новое множество, содержащее все элементы, которые есть и в первом множестве, и во втором.

✓ Метод difference() (разность (–)) возвращает новое множество, содержащее все элементы, которые есть во множестве a_set, но которых нет во множестве b_set.

Примеры применения операций:

```
a_set = {2, 4, 5, 9, 12, 21, 30, 51, 76, 127, 195}
```

```
30 in a_set
```

```
True
```

```
31 in a_set
```

```
False
```

```
b_set = {1, 2, 3, 5, 6, 8, 9, 12, 15, 17, 18, 21}
a_set.union(b_set)
```

```
{1, 2, 3, 4, 5, 6, 8, 9, 12, 15, 17, 18, 21, 30, 51, 76, 127, 195}
```

```
a_set.intersection(b_set)
```

```
{2, 5, 9, 12, 21}
```

```
a_set.difference(b_set)
```

```
{4, 30, 51, 76, 127, 195}
```

```
a_set.symmetric_difference(b_set)
```

```
{1, 3, 4, 6, 8, 15, 17, 18, 30, 51, 76, 127, 195}
```

Методы работы с множествами:

✓ copy() — создает копию множества.

✓ add() — добавляет во множество.

✓ remove() — удаляет из множества. Если элемент не найден, то возбуждается исключение KeyError.
✓ discard() — удаляет из множества, если он присутствует.
✓ pop() — удаляет произвольный элемент из множества и возвращает его. Если элементов нет, то возбуждается исключение KeyError.
✓ clear() — удаляет все элементы из множества

Задачи к лабораторной работе №6

1) Дана непустая последовательность символов. Построить и напечатать множества, элементами которых являются встречающиеся в последовательности: а) цифры от «0» до «9» и знаки арифметических операций; б) буквы от «A» до «F» и от «X» до «Z».
2) Используя множества, подсчитать общее количество цифр и знаков «+»,«– », «*» в строке, введённой с клавиатуры.
3) Составить программу формирования множества строчных латинских букв, входящих в строку, введённую с клавиатуры, и подсчёта количества знаков препинания в ней.
4) Используя множества, вывести общие буквы трёх предложений.
5) Используя множества, вывести наибольшие цифры трёх целых чисел.
6) Используя множества, подсчитать количество цифр в заданной строке и напечатать их.
7) Используя множества, вывести различные буквы трёх предложений, то есть такие, какие есть только в одном из них.
8) Даны три строки. Используя множества, определить, можно ли из символов первых двух строк получить третью строку.
9) Используя множества, напечатать по одному разу в алфавитном порядке все строчные русские гласные буквы, входящие в заданный текст.
10) Используя множества, напечатать в возрастающем порядке все цифры, входящие в десятичную запись данного десятичного числа.
11) Дан текст. Используя множества, определить, каких букв больше - гласных или согласных.
12) Дано натуральное число. Используя множества, напечатать в возрастающем порядке все цифры, которых нет в записи данного числа.
13) Написать программу, чтобы удалить из первого множества пересечение второго множества с первым.
14) Дан текст. Используя множества, напечатать в алфавитном порядке все согласные буквы, которые не входят в текст.
15) Проверить, нет ли у двух заданных множеств общих элементов.
16) Используя множества, определить, состоят ли две заданные строки из одинаковых символов.
17) Даны два списка. Используя множества, найти общие элементы в обоих списках.
18) Используя множества, найти разность двух заданных множеств.

18) Даны две строки. Используя множества, определить, есть ли общие символы в этих строках.
19) Дан список. Используя множества, найти количество уникальных элементов в списке.
20) Даны два множества. Используя множества, определить, является ли одно множество подмножеством другого.
21) Даны два списка. Используя множества, определить, есть ли общие элементы в этих списках.
22) Используя множества, определить, является ли заданная строка палиндромом.
23) Даны два множества. Используя множества, найти объединение данных множеств.
24) Дан список. Используя множества, определить, есть ли дублирующиеся элементы в этом списке.
25) Даны два множества. Используя множества, найти пересечение данных множеств.
26) Дан список. Используя множества, определить, является ли список упорядоченным в порядке возрастания.
27) Даны два множества. Используя множества, определить, есть ли общие элементы в этих множествах.
28) Дан список. Используя множества, определить, являются ли все элементы в списке уникальными.
29) Даны два множества. Используя множества, найти разность между этими двумя множествами.

Лабораторная работа №7
Словари. Операции над словарями. Методы для работы со словарями

Цель занятия: получение теоретических знаний и практических навыков работы со словарями на Python.

Словари в языке Python относится к типу данных, которые представляют собой неупорядоченные коллекции элементов, состоящих из пар ключ-значение. Словари создаются с помощью фигурных скобок {} и могут быть изменяемыми.

Способы создания словарей:

- ✓ использование фигурных скобок {}:

```
# пустой словарь
my_dict = {}

# словарь с несколькими парами ключ-значение
student = {
    "имя": "Анна",
    "возраст": 20,
    "группа": "А7"
}
```

✓ использование функции dict():

```
# пустой словарь
my_dict = dict()

# словарь с несколькими парами ключ-значение
student = dict(имя="Анна", возраст=20, группа="А7")
```

✓ использование функции zip():

```
keys = ["имя", "возраст", "группа"]
values = ["Анна", 20, "А7"]
student = dict(zip(keys, values))
print(student)
```

```
{'имя': 'Анна', 'возраст': 20, 'группа': 'А7'}
```

✓ использование генератора словаря:

```
student = {f"предмет_{i}": i * 10 for i in range(1, 6)}
print(student)
```

```
{'предмет_1': 10, 'предмет_2': 20, 'предмет_3': 30, 'предмет_4': 40, 'предмет_5': 50}
```

Операции над словарями:

✓ доступ к элементам по ключу:

```
my_dict = {"имя": "Анна", "возраст": 20}
name = my_dict["имя"]
age = my_dict.get("возраст")
print(name)
print(age)
```

```
Анна
20
```

✓ проверка наличия ключа в словаре:

```
my_dict = {"имя": "Анна", "возраст": 20}
name = my_dict["имя"]
age = my_dict.get("возраст")
print(name)
print(age)
```

```
Анна
20
```

✓ del: удаляет из словаря элемент с заданным ключом. Если элемент с таким ключом отсутствует, то возникает ошибка (возбуждается исключение KeyError).

Основные методы работы со словарями:

✓ keys(): возвращает список всех ключей в словаре.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
keys = my_dict.keys()
print(keys)
```

```
dict_keys(['имя', 'возраст', 'группа'])
```

✓ values(): возвращает список всех значений в словаре.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
values = my_dict.values()
print(values)
```

```
dict_values(['Анна', 20, 'А7'])
```

✓ items(): возвращает список кортежей (пар ключ-значение) в словаре.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
items = my_dict.items()
print(items)
```

```
dict_items([('имя', 'Анна'), ('возраст', 20), ('группа', 'А7')])
```

✓ get(): возвращает значение, связанное с указанным ключом. Если ключ не существует, возвращает указанное значение по умолчанию.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
age = my_dict.get("возраст", 0)
grade = my_dict.get("класс", "-")
print(age, grade, sep='\n')
```

```
20
-
```

✓ pop(): удаляет элемент из словаря по указанному ключу и возвращает его значение. Если ключ не найден, возвращает указанное значение по умолчанию.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
name = my_dict.pop("имя")
grade = my_dict.pop("класс", "-")
print(name, grade, sep='\n')
```

```
Анна
-
```

✓ update(): обновляет словарь, добавляя пары ключ-значение из другого словаря или итерируемого объекта.

```
my_dict = {"имя": "Анна", "возраст": 20}
additional_info = {"группа": "А7", "рейтинг": 4.5}
my_dict.update(additional_info)
print(my_dict)
```

```
{'имя': 'Анна', 'возраст': 20, 'группа': 'А7', 'рейтинг': 4.5}
```

✓ clear(): удаляет все элементы из словаря.

```
my_dict = {"имя": "Анна", "возраст": 20, "группа": "А7"}
my_dict.clear()
print(my_dict)
```

```
{}
```

Задачи к лабораторной работе №7

1) Напишите программу: а) для сортировки (по возрастанию и убыванию) словаря по значению; б) для объединения двух словарей и суммирующую значения для общих ключей; в) для проверки наличия нескольких ключей в словаре.
2) Напишите программу: а) для добавления ключа в словарь; б) для печати всех уникальных значений в словаре; в) для подсчёта количества элементов в значениях словаря, представляющих собой списки.
3) Напишите программу: а) для объединения словарей в один новый словарь; б) для сортировки словаря по значению; в) чтобы очистить список значений в словаре, если значениями словаря являются списки.
4) Напишите программу: а) чтобы проверить, существует ли данный ключ в словаре; б) для создания и отображения всех комбинаций букв, выбирая каждую букву из другого ключа в словаре, например, задан словарь {'1': ['a', 'b'], '2': ['c', 'd']}, результат: ac, ad, bc, bd; в) для замены значений словаря их средним значением.
5) Напишите программу: а) для перебора словарей с использованием цикла for; б) чтобы найти три наибольших значения соответствующих ключей в словаре; в) для сопоставления значений ключей в двух словарях.
6) Напишите программу для создания и печати словаря, содержащего число (от 1 до n) в форме (x, $x^* x$), например, при $n = 5$ получить словарь {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}; б) для объединения значений в списке словарей; в) для удаления пустых элементов из заданного словаря.
7) Напишите программу: для печати словаря, где ключами являются числа от 1 до 15 (оба включены), а значения представляют собой квадрат ключей; б) для создания словаря из строки, где значение — число вхождений символа в строку; в) для фильтрации словаря на основе значений.

8) Напишите программу: а) для объединения двух словарей; б) для печати словаря в табличном формате; в) для преобразования нескольких списков во вложенный словарь.
9) Напишите программу: а) для перебора словарей с использованием цикла for; б) для подсчёта суммы значений, связанных с ключом в списке словарей; в) для фильтрации высоты и ширины учащихся, которые хранятся в справочнике.
10) Напишите программу: а) для суммирования всех элементов в словаре; б) для преобразования списка во вложенный словарь ключей; в) чтобы проверить, все ли значения в словаре одинаковы.
11) Напишите программу: а) для перемножения всех элементов в словаре; б) для сортировки списка в алфавитном порядке в словаре; в) для создания словаря, группирующего последовательность пар ключ-значение в словарь списков.
12) Напишите программу: а) для удаления ключа из словаря; б) для удаления пробелов из ключей словаря; в) для разделения данного словаря списков на список словарей.
13) Напишите программу: а) для сопоставления двух списков в словаре; б) чтобы получить три самых дорогих товара в магазине; в) для удаления указанного словаря из заданного списка.
14) Напишите программу: а) для сортировки заданного словаря по ключу; б) чтобы получить ключ, значение и элемент в словаре; в) для преобразования строковых значений данного словаря в целочисленные (или числа с плавающей точкой).
15) Напишите программу: а) чтобы получить максимальное и минимальное значение в словаре; б) для печати в отдельные строки ключей и значений словаря; в) для создания словаря ключей *x*, *y* и *z*, где каждый ключ имеет значение списка из 11–20, 21–30 и 31–40 соответственно. Получите доступ к пятому значению каждого ключа из словаря.
16) Напишите программу, которая принимает список словарей и объединяет их в один словарь.
17) Напишите программу, которая проверяет, все ли значения в словаре являются уникальными.
18) Напишите программу, которая находит ключ с наибольшим значением в словаре
19) Напишите программу, которая находит сумму всех значений в словаре.
20) Напишите программу, которая заменяет все значения в словаре на их квадраты.
21) Напишите программу, которая находит наибольшее значение в словаре и выводит соответствующий ключ.
22) Напишите программу, которая удаляет все ключи со значениями, меньшими заданного числа.
23) Напишите программу, которая находит ключи с самыми длинными значениями в словаре.

24) Напишите программу, которая сортирует словарь по ключам в обратном порядке.
25) Напишите программу, которая удаляет все элементы из словаря, у которых ключи не начинаются с заданной буквы.
26) Напишите программу, которая находит пересечение двух словарей.
27) Напишите программу, которая находи все ключи, содержащие заданную подстроку.
28) Напишите программу, которая находит сумму всех численных значений в словаре.
29) Напишите программу, которая находит ключ с минимальным значением в словаре.
30) Напишите программу, которая находит разницу между двумя словарями.

Лабораторная работа №8
Подпрограммы - функции. Анонимные функции.

Цель занятия: получение теоретических знаний и практических навыков работы с подпрограммами (функциями) на Python.

Для эффективного структурирования больших программ в ней удобно выделять отдельные смысловые блоки, реализующие логически замкнутый алгоритм (подзадачу). Эти блоки обычно называют подпрограммами и определенным образом выделяют в коде «основной» программы. В языке Python такие конструкции называют функциями. При этом основная программа содержит как обычные операторы, так и команды управления этими блоками (вызовы функций), с соответствующим обменом данными. При вызове функции (подпрограммы) в нее передаются данные (параметры), и выполняются некоторые действия. После выполнения подпрограммы работа продолжается с той команды, которая непосредственно следует за её вызовом.

Функция должна быть описана до того, как она будет вызываться. Параметры, используемые в функции, разделяются на локальные, которые объявляются и используются только внутри неё, и глобальные, которые объявляются в основной программе и доступны как ей, так и функции. Передача данных между основной программой и функцией может осуществляться только через глобальные параметры: либо непосредственно, по их уникальному имени, либо через механизм формальных параметров. Формальные параметры являются аргументами функций. При вызове функции эти аргументы заменяются определенными значениями – фактическими параметрами.

Функция (подпрограмма) в Python создается с помощью инструкции **def** В общем виде набор инструкций, объявляющих функцию с заданным именем name, имеет следующий формат:

```
def name(аргументы):
тело функции
```

Данная составная инструкция состоит из строки заголовка с ключевым словом def и следующего за ней блока инструкций, выделяемым отступами (или идущим после двоеточия). Этот блок инструкций образует ***тело*** функции, то есть программный код, который выполняется интерпретатором всякий раз, когда производится вызов функции. ***Не забудьте двоеточие в конце строки заголовка.*** Функция может содержать внутри себя другую функцию, причем тело «внутренней» функции **должно иметь дополнительный отступ**. В строке заголовка инструкции def, кроме имени функции указывается и список из её аргументов (параметров) в круглых скобках. Этот список может быть пустым. С именем функции будет связан объект функция, который вызывается в соответствующем месте про граммы. Имена аргументов в строке заголовка будут связаны с объектами, передаваемыми в функцию в точке вызова.

Рассмотрим простую программу с функцией без аргументов:

```
def a_funct():
print("North Caucasus State Academy")
a_funct()#вызов функции
a_funct()#вызов функции
```

В результате будет два раза напечатан текст: North Caucasus State Academy. Тело функции часто содержит инструкцию return – она завершает работу функции (и может, например, её досрочно завершить) и передает результат вычислений вызывающей программе.

```
def имя_функции(arg1, arg2,... argN):
...
return <набор возвращаемых значений>
```

Рассмотрим простой пример:

```
def fff(a,b):
return a+2*b
print( fff(3,6) ) # напечатает 15
print( fff(b=3, a=2) ) # напечатает 8
```

Обратите внимание на последнюю строку, в результате получится именно число 8, хотя порядок аргументов выглядит нарушенным. А что будет, если вызвать эту же функцию с нечисловыми аргументами? print(fff('aaa','bbb')) Будет напечатано, как и следовало ожидать, aaabbbbbb. Таким образом, можно зафиксировать важное правило – тип аргументов функции никак не объявляется. Это соответствует концепции динамической типизации, принятой в Python, в частности, не

требуется объявлять типы переменных, аргументов или возвращаемых значений. В примере ниже функция возвращает кортеж значений

```
def fff(a,b):
    return (a, b, a+2*b)
print( fff(3,6) )                # результат (3, 6, 15)
```

В языке Python с аргументами функций можно обращаться достаточно свободно. Например, параметрам функции можно присвоить значения по умолчанию:

```
def fff(a=1, b=2):
    return a+b
print( fff(b=4, a=5))            # напечатает 9
print( fff())                    # напечатает 3
print( fff(3))                   # напечатает 5
```

Обратите внимание, число параметров при вызове функции может быть меньше чем указано в заголовке инструкции `def`. Значения по умолчанию в Python называют **ключевыми аргументами,** отличая их от «обычных» аргументов. Вот еще один пример:

```
def mixed_arg(a, x=2, y=3):
    print(a+x+y)
mixed_arg (1)                    # напечатает 6
mixed_arg (1,5)                  # напечатает 9
```

Таким образом, можно обратиться к функции, используя имена параметров как ключи, независимо от порядка их расположения. Следует отметить, что нельзя ставить обязательные аргументы после необязательных, если имена параметров не указаны, поэтому оба нижеследующих вызова дадут ошибку:

```
mixed_arg (x=2, y=1, 1)
mixed_arg (x=2, 1)
```

Локальные и глобальные параметры. Переменные, объявленные внутри тела функции, имеют локальную область видимости и называются локальными, а те, что объявлены вне какой-либо функции, – глобальными. Это означает, что доступ к локальным переменным имеют только те функции, внутри которых они были объявлены, в то время как доступ к глобальным переменным можно получить по всей программе и в любой функции. Рассмотрим пример:

```
def fff(a, b):
    x=10
    return x+a+2*b
print( fff(3,6) ) # напечатает 25
```

Переменные `a,b,x` внутри функции `fff` – это локальные переменные, они (и их имена) доступны только программному коду внутри инструкции `def` и существует только во время выполнения функции. Фактически любые имена, которым тем или иным способом были присвоены некоторые значения внутри функции, по умолчанию классифицируются как

локальные переменные. Если в основной программе после `print(fff(3,6))` вставить строчку `print(a)` или `print(x)`, то будет получено сообщение об ошибке «`name 'a' is not defined`» или «`name 'x' is not defined`», то есть основная программа «не видит» эти переменные. Конечно, можно создать в основной программе свою переменную `x`, записав, например, `x=20;` и тогда `print(x)`, естественно, не будет ошибкой. Теперь сделаем переменную `x` глобальной также и для подпрограммы, то есть просто не будем определять её внутри функции :

```
def fff(a, b):
return x+a+2*b
x=20
print( fff(3,6) ) # напечатает 35
```

Таким образом, для того чтобы получить доступ к глобальной переменной на чтение, достаточно лишь указать её имя. А что, если необходимо изменить глобальную переменную внутри функции? Для этого потребуется инструкция `global`. Например:

```
a = 1
def bbb():
    global a
    a = 25
print(a)      # 1
bbb()
print(a)      # 25
```

```
def fff(a,b):
     global x
     y=x+a+2*b
     x=100
     return y
x=20
print(fff(3,6)) #выведет 35
print(x)        #выведет 100
```

Первая программа выведет сначала 1, потом 25, а вторая – 35 и 100. Инструкция `global` указывает, что функция будет изменять одно или более глобальных имен, то есть имен, которые находятся в области видимости (в пространстве имен) вмещающего модуля. **Рекурсивный вызов**, когда подпрограмма обращается сама к себе, осуществляется обычным образом. Например, числа Фибоначчи можно получить с помощью следующей программы:

```
def Fib(n):
if n<2: return 1
return Fib(n-1)+Fib(n-2)
    print(Fib(5))                    # результат равен 8
```

При отладке больших программ удобно использовать инструкцию `pass`, которая «ничего не делает»:

```
>>> def fun(): pass
>>> print(fun())
None
```

Анонимные функции в Python создаются с использованием ключевого слова lambda, за которым следует список аргументов в скобках, двоеточие и

выражение, которое будет выполнено при вызове функции. Возвращаемое значение указывается после ключевого слова lambda. Рассмотрим примеры: функция, возвращающая квадрат числа:

```
square = lambda x: x ** 2
print(square(5))
```

```
25
```

функция, складывающая два числа:

```
add = lambda a, b: a + b
print(add(5, 3))
```

```
8
```

использование анонимной функции внутри другой функции:

```
def apply_operation(x, y, operation):
    return operation(x, y)

result = apply_operation(4, 2, lambda a, b: a * b)

print(result)
```

```
8
```

Анонимные функции полезны в случаях, когда не требуется создание именованных функций или когда нужна функция только для использования в одном месте программы.

Задачи к лабораторной работе №8

Задачи 8.1

1) Описать функцию Mean, вычисляющую среднее арифметическое AMean и среднее геометрическое GMean двух положительных чисел *X* и *Y* формулам:

$$AMean = (X + Y)/2,\ GMean = X * Y.$$

2) Описать функцию Circle, находящую площадь и длину окружности круга радиуса *R*.

3) Описать функцию TrianglePS, вычисляющую по стороне *a* равностороннего треугольника его периметр $P = 3a$ и площадь $S = 3/4a2$.

4) Описать функцию RingS, находящую площадь кольца, заключённого между двумя окружностями с общим центром и радиусами *R* и R_2.

5) Описать функцию RectPS, вычисляющую периметр *P* и площадь *S* прямоугольника со сторонами, параллельными осям координат, по координатам (x_1, y_1), (x_2, y_2) его противоположных вершин.

6) Описать функцию TriangleP, находящую периметр равнобедренного треугольника по его основанию *a* и высоте *h*, проведенной к основанию. Для нахождения боковой стороны *b* треугольника использовать *теорему Пифагора*: $b^2 = (a/2)^2 + h^2$

7) Описать функцию InvertDigits, меняющую порядок следования цифр целого положительного числа *K* на обратный.

8) Описать функцию SumRange, находящую сумму всех целых чисел от *A* до *B* включительно (*A* и *B* — целые). Если *A*>*B*, то функция возвращает 0.

9) Описать функцию TypeTrl, которая позволяет по заданным координатами вершин треугольника определить вид треугольника (равнобедренный, равносторонний, прямоугольный или обычный).

10) Описать функцию Quarter, определяющую номер координатной четверти, в которой находится точка с ненулевыми вещественными координатами (x, y).

11) Описать функцию DigitCountSum, находящую количество цифр целого положительного числа K , а также их сумму S .

12) Описать функцию Calc, выполняющую над ненулевыми вещественными числами A и B одну из арифметических операций и возвращающую её результат. Вид операции определяется целым параметром *Op*: 1 — вычитание, 2 — умножение, 3 — деление, остальные значения — сложение.

13) Описать функцию LuckyNum, которая позволяет определить, является ли четырёхзначный номер счастливым. Счастливым называется номер, у которого сумма первых двух цифр номера равна сумме последних двух цифр. Получить все четырёхзначные счастливые номера.

14) Описать функцию DegToRad, находящую величину угла в радианах, если дана его величина D в градусах ($0 < D < 360$). Воспользоваться следующим соотношением: $180° = \pi$ радианов.

15) Описать функцию IsLeapYear, определяющую, является ли год Y високосным. *Високосным* считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400.

16) Описать функцию SphereVS, вычисляющую объем V и площадь поверхности S сферы радиусом r.

17) Описать функцию CylinderVS, находящую объем V и площадь поверхности S цилиндра с радиусом основания r и высотой h.

18) Описать функцию ConeVS, вычисляющую объем V и площадь поверхности S конуса с радиусом основания r и высотой h.

19) Описать функцию RectangularBoxVS, находящую объем V и площадь поверхности S прямоугольного параллелепипеда со сторонами a, b и c.

20) Описать функцию EllipseS, вычисляющую площадь эллипса с полуосями a и b.

21) Описать функцию TrapezoidPS, находящую периметр P и площадь S трапеции с основаниями a и b, и высотой h.

22) Описать функцию RhombusS, вычисляющую площадь ромба с диагоналями d_1 и d_2.

23) Описать функцию ParallelogramPS, находящую периметр P и площадь S параллелограмма со сторонами a и b, и углом между ними α.

24) Описать функцию PrimeNumbers, которая выводит все простые числа, меньшие или равные заданному числу N.

25) Описать функцию BinaryToDecimal, которая преобразует заданное двоичное число в десятичное число.

26) Описать функцию PerfectNumber, определяющую, является ли заданное положительное число совершенным числом. Совершенным числом называется число, равное сумме своих делителей (кроме самого числа).

27) Описать функцию GCD, вычисляющую наибольший общий делитель двух заданных чисел.

28) Описать функцию MinMax, находящую минимальное и максимальное значение из заданного списка чисел.

29) Описать функцию PrimeFactors, которая находит все простые множители заданного числа N и выводит их.

30) Описать функцию ArmstrongNumber, определяющую, является ли заданное положительное число числом Армстронга. Число Армстронга — это число, которое равно сумме своих цифр, возведенных в степень, равную количеству цифр в числе

Задача 8.2. Функции обработки матриц

1) Описать функцию RemoveRowCol, удаляющую из матрицы A размера $M \times N$ строку и столбец, которые содержат элемент $A_{K, L}$ (предполагается, что $M > 1$ и $N > 1$; если $K > M$ или $L > N$, то матрица не изменяется).
2) Описать функцию Split, формирующую по заданному списку A два новых списка B и C; при этом список B содержит все чётные числа из списка A, а список C — все нечётные числа (в том же порядке).
3) Описать функцию SwapRow, осуществляющую перемену местами строк вещественной матрицы A размера $M \times N$ с номерами K_1 и K_2.
4) Описать функцию Transp, выполняющую транспонирование (то есть зеркальное отражение относительно главной диагонали) квадратной матрицы A порядка M.
5) Описать функцию MtrProd, вычисляющую произведение двух матриц.
6) Описать функцию SumMtr, вычисляющую сумму двух матриц.
7) Дана квадратная матрица. Вычислить сумму элементов главной или побочной диагонали в зависимости от выбора пользователя. Сумма элементов любой диагонали должна вычисляться в одной и той же функции.
8) Описать функцию RemoveRows, удаляющую из матрицы A размера $M \times N$ строки с номерами от K_1 до K_2 включительно (предполагается, что $1 < K_1 \le K_2$). Если $K_1 < M$, то матрица не изменяется; если $K_2 > M$, то удаляются строки матрицы с номерами от K_1 до M.
9) Описать функцию InvertStr, возвращающую инвертированную подстроку строки S, содержащую в обратном порядке N символов строки S, начиная с её K-го символа. Если K превосходит длину строки S, то возвращается пустая строка; если длина строки меньше $K + N$, то инвертируются все символы строки, начиная с её K-го символа.
10) Описать функцию SwapCol, осуществляющую перемену местами столбцов матрицы A размера $M \times N$ с номерами K_1 и K_2.

11) Описать функцию CompressStr, выполняющую сжатие строки S по следующему правилу: каждая подстрока строки S, состоящая из более чем четырёх одинаковых символов C, заменяется текстом вида «$C\{K\}$», где K — количество символов C (предполагается, что строка S не содержит фигурных скобок «{» и «}»). Например, для строки $S = bbbccccce$ функция вернет строку $bbbc\{5\}e$.
12) Описать функцию IsIdent, проверяющую, является ли строка S допустимым идентификатором, то есть непустой строкой, которая содержит только латинские буквы, цифры и символ подчеркивания «_» и не начинается с цифры.
13) Описать функцию Chessboard, формирующую по целым положительным числам M и N матрицу A размера $M \times N$, которая содержит числа 0 и 1, расположенные в «шахматном» порядке, причём $A_1 = 0$.
14) Описать функцию Bell, меняющую порядок элементов списка L на следующий: наименьший элемент списка располагается на первом месте, наименьший из оставшихся элементов — на последнем, следующий по величине располагается на втором месте, следующий — на предпоследнем и т. д. (в результате график значений элементов будет напоминать колокол).
15) Описать функцию RemoveCols, удаляющую из матрицы A размера $M \times N$ столбцы с номерами от K_1 до K_2 включительно (предполагается, что $1 < K_1 \le K_2$). Если $K_1 > \mathrm{N}$, то матрица не изменяется; если $K_2 > N$, то удаляются столбцы матрицы с номерами от K_1 до N.
16) Описать функцию SplitEvenOdd, которая разделяет список чисел на два списка: список четных чисел и список нечетных чисел.
17) Описать функцию ReverseTuples, которая принимает список кортежей и возвращает новый список, где каждый кортеж перевернут в обратном порядке.
18) Описать функцию ReplaceDuplicates, которая заменяет все дублирующиеся элементы в списке на заданное значение.
19) Описать функцию SquaredSum, которая вычисляет сумму квадратов элементов списка.
20) Описать функцию UniqueIntersection, которая принимает два списка и возвращает список уникальных общих элементов обоих списков.
21) Описать функцию SplitByCondition, которая разделяет список на два списка: один содержит элементы, удовлетворяющие заданному условию, а второй содержит остальные элементы.

22) Описать функцию RemoveDuplicatesList, которая удаляет все повторяющиеся элементы из списка.
23) Описать функцию TransposeMatrix, которая принимает матрицу в виде списка списков и возвращает ее транспонированную версию.
24) Описать функцию MergeDicts, которая принимает произвольное количество словарей и объединяет их в один словарь.
25) Описать функцию CountSublists, которая принимает список и возвращает количество вложенных списков внутри него.
26) Описать функцию SwapElements, которая принимает список и индексы двух элементов и меняет их местами.
27) Описать функцию CountOccurrences, которая подсчитывает количество вхождений каждого элемента в списке и возвращает словарь, где ключи — элементы списка, а значения — количество их вхождений.
28) Описать функцию SortNestedList, которая принимает список вложенных списков и сортирует его по возрастанию суммы элементов в каждом вложенном списке.
29) Описать функцию FlattenList, которая принимает вложенный список и возвращает одноуровневый список, содержащий все элементы из вложенных списков.
30) Описать функцию RemoveEmptyStrings, которая удаляет пустые строки из списка строк.

Задача 8.3. Рекурсивные функции.

1) Описать рекурсивные функции Fact и Fact2, вычисляющие значения факториала $N!$ и двойного факториала $N!!$ соответственно ($N > 0$ — параметр целого типа).
2) Описать рекурсивную функцию PowerN, находящую значение n-й степени числа x по формуле:

$$x^0 = 1, x^n = x \cdot x^{n-1} \text{ при } n > 0, \qquad x^n = \frac{1}{x^{-n}} \text{ при } n < 0$$

$$(x \geq 0 - \text{вещественное число}, \ n \ - \text{целое}).$$

3) Описать рекурсивную функцию SqrtK(x, k, n), находящую приближенное значение корня k -й степени из числа x по формуле:

$$y(0) = 1, \qquad y(n+1) = y(n) - (y(n) - \frac{x}{y^{k-1}(n)})/k,$$

где $y(n)$ обозначает SqrtK(x, k, n) (x — вещественный параметр, k и n — целые; x > 0, k > 1, n > 0).

4) Описать рекурсивную функцию FibRec, вычисляющую N-е число Фибоначчи $F(N)$ по формуле:

$$F(1) = F(2) - 1,\ F(k) = F(k-2) + F(k-1),\ k = 3,4,\ldots$$

С помощью этой функции найти пять чисел Фибоначчи с указанными номерами и вывести эти числа вместе с количеством рекурсивных вызовов функции FibRec, потребовавшихся для их нахождения.

5) Описать рекурсивную функцию $C(m, n)$, находящую число сочетаний из n элементов по m, используя формулу:

$$C(0, n) = C(n, n) = 1,\ C(m,\ n) = C(m,\ n-1) + C(m-1, n-1)$$

при $0 < m < n$ (m и n — целые параметры; $n > 0$, $0 \le m \le n$).

Дано число N и пять различных значений M. Вывести числа $C(M, N)$ вместе с количеством рекурсивных вызовов функции C, потребовавшихся для их нахождения.

6) Описать рекурсивную функцию NOD(A, B), находящую наибольший общий делитель двух натуральных чисел A и B, используя алгоритм Евклида:

$$\mathrm{NOD}(A, B) = \mathrm{NOD}(B \bmod A, A), \text{ если } A \neq 0,$$

$$\mathrm{NOD}(0, B) = B.$$

7) С помощью этой функции найти наибольшие общие делители пар A и B, A и C, A и D , если даны числа A, B, C, D .

8) Описать рекурсивную функцию MinRec, которая находит минимальный элемент массива A размера N, *не используя оператор цикла*. С помощью этой функции найти минимальные элементы трёх заданных массивов.

9) Описать рекурсивную функцию Digits, находящую количество цифр в строке S *без использования оператора цикла*. С помощью этой функции найти количество цифр в заданной строке.

10) Описать рекурсивную функцию Simm, проверяющую, является ли симметричной строка S , *без использования оператора цикла*. С помощью этой функции проверить заданную строку.

11) Задано положительное и отрицательное число в двоичной системе. Составить программу вычисления суммы этих чисел, используя функцию сложения чисел в двоичной системе счисления.

12) Описать рекурсивную функцию Root, которая методом деления отрезка пополам находит с точностью ε корень уравнения $f(x) = 0$ на отрезке $[a, b]$ (считать, что $\varepsilon > 0$, $a < b$, $f(a) - f(b) < 0$ и $f(x)$ — непрерывная и монотонная на отрезке $[a, b]$ функция).

13) Дано число *n*, десятичная запись которого не содержит нулей. Получите число, записанное теми же цифрами, но в противоположном порядке. При решении этой задачи нельзя использовать циклы, строки, списки разрешается только рекурсия и целочисленная арифметика. Функция должна возвращать целое число, являющееся результатом работы программы, выводить число по одной цифре нельзя.
14) Описать рекурсивную логическую функцию Simm, проверяющую, является ли симметричной часть строки *S* , начинающаяся *i*-м и заканчивающаяся *j*-м её элементами.
15) Дано натуральное число *N*. Вычислите сумму его цифр. При решении этой задачи нельзя использовать строки, списки и циклы.
16) Дано слово, состоящее только из строчных латинских букв. Проверьте, является ли это слово палиндромом. Выведите *yes* или *no*. При решении этой задачи нельзя пользоваться циклами, также нельзя использовать срезы с шагом, отличным от 1.
17) Описать рекурсивную функцию BinarySearch, которая находит индекс заданного элемента в отсортированном массиве.
18) Описать рекурсивную функцию GCD, находящую наибольший общий делитель двух чисел *A* и *B* с помощью алгоритма Евклида.
19) Описать рекурсивную функцию Palindrome, которая проверяет, является ли заданная строка палиндромом.
20) Описать рекурсивную функцию SumDigits, вычисляющую сумму цифр числа.
21) Описать рекурсивную функцию BinaryToDecimal, которая переводит число из двоичной системы счисления в десятичную.
22) Описать рекурсивную функцию CountVowels, находящую количество гласных букв в строке *S* без использования оператора цикла. С помощью этой функции найти количество гласных букв в заданной строке.
23) Описать рекурсивную функцию CountConsonants, находящую количество согласных букв в строке *S* без использования оператора цикла. С помощью этой функции найти количество согласных букв в заданной строке.
24) Описать рекурсивную функцию CountSpaces, находящую количество пробелов в строке *S* без использования оператора цикла. С помощью этой функции найти количество пробелов в заданной строке.
25) Описать рекурсивную функцию CountUppercase, находящую количество прописных букв в строке *S* без использования оператора цикла. С помощью этой функции найти количество прописных букв в заданной строке.
26) Описать рекурсивную функцию CountLowercase, находящую количество строчных

букв в строке *S* без использования оператора цикла. С помощью этой функции найти количество строчных букв в заданной строке.

27) Описать рекурсивную функцию CountDigits, находящую количество цифр в строке *S* без использования оператора цикла. С помощью этой функции найти количество цифр в заданной строке.
28) Описать рекурсивную функцию CountSymbols, находящую количество определенного символа в строке *S* без использования оператора цикла. С помощью этой функции найти количество определенного символа в заданной строке.
29) Описать рекурсивную функцию IsSubstringInOrder, которая проверяет, является ли одна строка подстрокой другой строки в определенной последовательности символов.
30) Описать рекурсивную функцию IsAlternating, которая проверяет, является ли заданная строка последовательностями символов, чередующихся по типу (буква-цифра-буква-цифра и т.д.).
31) Описать рекурсивную функцию CountOccurrence, находящую количество вхождений подстроки в строке *S* без использования оператора цикла. С помощью этой функции найти количество вхождений подстроки в заданной строке.

Лабораторная работа №9
Файлы. Работа с файлами.

Цель занятия: получение теоретических знаний и практических навыков работы с файлами на Python.

Работа с текстовыми файлами в Python позволяет считывать данные из файлов, записывать данные в файлы и выполнять другие операции с файлами. В Python работа с текстовыми файлами осуществляется с использованием функций модуля io или операторов with и open. Вот основные операции, которые можно выполнить с текстовыми файлами в Python:

1. Открытие файла:

Для работы с текстовым файлом сначала нужно открыть его с помощью функции open(). Функция принимает имя файла и режим доступа в качестве параметров. Режимы доступа могут быть "r" (чтение), "w" (запись) или "a" (добавление). Например:

```
file = open("file.txt", "r")
```

2. Чтение данных из файла:

После открытия файла вы можете считать его содержимое с помощью метода read() или построчно с помощью метода readline(). Например:

```
file = open("file.txt", "r")

# считать все содержимое файла
data = file.read()

# считать одну строку
line = file.readline()
```

Для чтения текстового файла используется функция open с аргументом "r" (read), а для записи или создания нового файла — функция open с аргументом "w" (write) или "a" (append) для дописывания данных в конец файла. Пример чтения текстового файла:

```
with open("file.txt", "r") as file:
    content = file.read()
    print(content)
```

В этом примере мы создаем файловый объект file, указывая имя файла "file.txt" и режим чтения "r". Далее, при помощи метода read(), мы считываем содержимое файла в переменную content и выводим ее на экран.

3. Запись данных в файл:

Пример записи в текстовый файл:

```
with open("file.txt", "w") as file:
    file.write("Hello, World!")
```

В этом примере мы создаем или открываем файл "file.txt" в режиме записи "w". При помощи метода write() мы записываем строку "Hello, World!" в файл. Пример дописывания данных в конец файла:

```
with open("file.txt", "a") as file:
    file.write("\nThis is a new line.")
```

В этом примере мы открываем файл "file.txt" в режиме дописывания "a". Мы записываем строку "\nThis is a new line." в файл, добавляя новую строку с помощью символа перевода строки \n. При работе с текстовыми файлами можно также использовать другие методы, например:

✓ readline() — для чтения отдельных строк из файла;

✓ readlines() — для чтения всех строк из файла и возвращения их в виде списка.

Также важно помнить о закрытии файла после завершения работы с ним. В примерах выше оператор with автоматически осуществляет закрытие файла, но при использовании функции open без оператора with следует вызвать метод close() для закрытия файла.

Далее на примере представлены основные методы работы с файлами.

```
import os
import chardet
import time
# Путь к файлу
file_path = 'D:/path/1/example.txt'
# Определение кодировки файла с помощью chardet
with open(file_path, 'rb') as file:
    raw_data = file.read()
    result = chardet.detect(raw_data)
    encoding = result['encoding']
    confidence = result['confidence']
    print(f"Определенная кодировка: {encoding} с уверенностью {confidence}")
# Примеры записи и добавления в файл
# Перезапись содержимого файла
with open(file_path, 'w', encoding='utf-8') as file:
    file.write("Привет, Рита!")
print("Перезапись файла завершена.")
# Добавление новой строки в файл
with open(file_path, 'a', encoding='utf-8') as file:
    file.write('\nЭто новая строка для моего файла')
print("Добавление новой строки завершено.")
# Проверка существования файла
if os.path.exists(file_path):
    print("Файл существует")
# Получение информации о файле
file_size = os.path.getsize(file_path)
print(f"Размер файла: {file_size} байт")
modification_time = os.path.getmtime(file_path)
print(f"Последнее изменение: {time.ctime(modification_time)}")
```

Работа с файлами CSV

Файлы формата **CSV** (**Comma-Separated Values**) являются текстовыми файлами, в которых значения разделяются запятыми. Работа с файлами CSV в Python осуществляется с помощью встроенной библиотеки csv. Вот основные операции, которые можно выполнить при работе с файлами CSV в Python:

1. Чтение данных из файла CSV:

Для чтения данных из файла CSV в Python используйте класс csv.reader. Он позволяет считывать данные построчно или как список значений. Вот пример чтения данных из файла:

```
import csv
```

```
with open("data.csv", "r") as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        # распечатать каждую строку как список значений
        print(row)
```

2. Запись данных в файл CSV:

Чтобы записать данные в файл CSV, вам нужно открыть файл в режиме записи ("w" или "a") и использовать класс csv.writer. Он позволяет записывать данные строка за строкой или по списку значений. Вот пример записи данных в файл:

```
import csv
```

```
with open("data.csv", "w") as file:
    csv_writer = csv.writer(file)

    # записать заголовки
    csv_writer.writerow(["Name", "Age"])

    # записать строку данных
    csv_writer.writerow(["Alice", 25])
```

3. Работа с разделителями и кавычками:

В файле CSV могут использоваться различные разделители и символы кавычек. По умолчанию, в Python разделителем является запятая и кавычками — двойные кавычки. Однако, вы можете указать другие разделители и кавычки при необходимости, передав их в соответствующие аргументы функции csv.reader или csv.writer.

4. Работа с заголовками:

Если файл CSV содержит заголовки, вы можете использовать методы next() или __next__() для пропуска первой строки. Это позволит вам начать чтение или запись данных с первой фактической строки.

5. Обработка CSV-данных:

Считанные или записанные данные CSV обычно представляются в виде списков значений для каждой строки. Вы можете использовать индексирование или итерацию для обработки отдельных значений или строк в файле CSV.

6. Дополнительные возможности:

Библиотека csv предоставляет и другие полезные функции, такие как установка разделителя десятичной запятой, обработка NULL-значений, пропуск строк без данных и т.д. Подробности можно найти в документации Python. При работе с файлами CSV важно учитывать особенности форматирования данных, такие как кавычки и экранирование символов, чтобы избежать проблем при чтении или записи.

Задачи к лабораторной работе 9

1) Дано целое число K ($0 < K < 10$) и текстовый файл, содержащий более K строк.

1) Создать новый текстовый файл, содержащий K последних строк исходного файла.

2) Дано целое число K ($0 < K < 10$) и текстовый файл, содержащий более K строк. Удалить из файла последние K строк.

3) Дано целое число *K* и текстовый файл. Удалить из каждой строки файла первые *K* символов (если длина строки меньше *K*, то удалить из нее все символы).

4) Даны два текстовых файла. Добавить в конец каждой строки первого файла соответствующую строку второго файла. Если второй файл короче первого, то оставшиеся строки первого файла не изменять.

5) Дано целое число *K* и текстовый файл. Удалить из файла строку с номером *K*. Если строки с таким номером нет, то оставить файл без изменений.

6) Дан текстовый файл. Создать символьный файл, содержащий все знаки препинания, встретившиеся в текстовом файле (в том же порядке).

7) Дан текстовый файл, каждая строка которого изображает целое число, дополненное слева и справа несколькими пробелами. Вывести количество этих чисел и их сумму.

8) Дана строка *S* и текстовый файл. Заменить в файле все пустые строки на строку *S*.

9) Дан текстовый файл, содержащий текст, выровненный по левому краю. Выровнять текст по центру, добавив в начало каждой непустой строки нужное количество пробелов (ширину текста считать равной 50). Строки нечетной длины перед центрированием дополнять слева пробелом.

10) Дан текстовый файл. Найти количество абзацев в тексте, если абзацы отделяются друг от друга одной или несколькими пустыми строками.

11) Дан текстовый файл, содержащий текст, выровненный по левому краю. Выровнять текст по правому краю, добавив в начало каждой непустой строки нужное количество пробелов (ширину текста считать равной 50).

12) Даны два текстовых файла. Добавить в начало первого файла содержимое второго файла.

13) Дано целое число *K* и текстовый файл. Вставить пустую строку перед строкой файла с номером *K*. Если строки с таким номером нет, то оставить файл без изменений.

14) Дан текстовый файл. Продублировать в нем все пустые строки.

15) Дан непустой текстовый файл (число строк > 2). Удалить из него первую и последнюю строки.

Задача 9.1. Файлы. Обработка данных.

1) Дан файл *f*, компоненты которого являются действительными числами. Найти произведение компонент файла.

2) Дан файл *f*, компоненты которого являются целыми числами. Ни одна из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* сначала шли положительные, потом отрицательные числа.

3) Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*, являющиеся точными квадратами.

4) Дан файл *f*, компоненты которого являются действительными числами. Найти сумму наибольшего и наименьшего из значений компонент.

5) Дан файл, содержащий различные даты. Каждая дата — это число, месяц и год. Найти год с наименьшим номером.

6) Дан файл *f*, компоненты которого являются целыми числами. Найти количество чётных чисел среди компонент.

7) Дан файл *f*. В файле не менее двух компонент. Определить, являются ли два первых символа файла цифрами. Если да, то установить, является ли число, образованное этими цифрами чётным.

8) Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*, являющиеся чётными числами.

9) Дан файл *f*, компоненты которого являются действительными числами. Найти наибольшее из значений модулей компонент с нечётными номерами.

10) Дан файл, содержащий различные даты. Каждая дата — это число, месяц и год. Найти все весенние даты.

11) Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*, делящиеся на 3 и не делящиеся на 7.

12) Дан файл *f*, компоненты которого являются действительными числами. Найти наименьшее из значений компонент с чётными номерами.

13) Записать в файл *g* все чётные числа файла *f*, а в файл *h* все нечётные. Порядок следования чисел сохраняется.

14) Дан файл *f*, компоненты которого являются целыми числами. Ни одна из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* не было двух соседних чисел с одним знаком.

15) Дан файл *f,* компоненты которого являются целыми числами. Ни одна из компонент файла не равна нулю. Файл *f* содержит столько же отрицательных чисел, сколько и положительных. Переписать компоненты файла *f* в файл *g* так, чтобы в файле *g* числа шли в следующем порядке: два положительных два отрицательных, два положительных, два отрицательных и т.д. (предполагается, что число компонент в файле *f* делится на 4).

16) Дан файл *f*, содержащий строки с именами студентов и их оценками. Найти средний балл каждого студента и записать его в новый файл *g*.

17) Дан файл *f*, содержащий информацию о продажах товаров. Найти суммарную выручку за каждый месяц и записать ее в новый файл *g*.

18) Дан файл *f*, содержащий строки с текстом. Найти количество слов в каждой строке и записать результат в новый файл *g*.

19) Дан файл *f*, содержащий список сотрудников компании и их заработные платы. Найти среднюю зарплату всех сотрудников и записать ее в новый файл *g*.

20) Дан файл *f*, содержащий список дат рождения людей. Найти самую старшую и самую младшую даты рождения и записать их в новый файл *g*.

21) Дан файл *f*, содержащий строки с информацией о товарах в интернет-магазине. Найти товар с самой высокой стоимостью и записать его название в новый файл *g*.

Лабораторная работа №10
Исключения и их обработка

Цель занятия: получение теоретических знаний и практических навыков обработки исключений при разработке программ на Python.

Исключения — это извещения интерпретатора, возбуждаемые в случае возникновения ошибки в программном коде или при наступлении какого-либо события. Если в коде не предусмотрена обработка исключения, то программа прерывается и выводится сообщение об ошибке.

Существуют три типа ошибок в программе:

✓ синтаксические — это ошибки в имени оператора или функции, отсутствие закрывающей или открывающей кавычек и т. д., т. е. ошибки в синтаксисе языка. Как правило, интерпретатор предупредит о наличии ошибки, а программа не будет выполняться совсем. Пример синтаксической ошибки:

```
print("Нет завершающей кавычки!)
```

```
  File "<ipython-input-131-a3916d1aed16>", line 1
    print("Нет завершающей кавычки!)
                                     ^
SyntaxError: EOL while scanning string literal
```

✓ логические — это ошибки в логике работы программы, которые можно выявить только по результатам работы скрипта. Как правило, интерпретатор не предупреждает о наличии ошибки. А программа будет выполняться, т. к. не содержит синтаксических ошибок. Такие ошибки достаточно трудно выявить и исправить;

✓ ошибки времени выполнения — это ошибки, которые возникают во время работы скрипта. Причиной являются события, не предусмотренные программистом. Классическим примером служит деление на ноль.

```
print(10/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-132-fe01563e1bc6> in <module>
----> 1 print(10/0)

ZeroDivisionError: division by zero
```

В языке Python исключения возбуждаются не только при ошибке, но и как уведомление о наступлении каких-либо событий. Например, метод index () возбуждает исключение ValueError, если искомый фрагмент не входит в строку:

```
"Строка".index("текст")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-133-0ed78ddd04aa> in <module>
----> 1 "Строка".index("текст")

ValueError: substring not found
```

Инструкция try...except...else...finally. Для обработки исключений предназначена инструкция try. Формат инструкции:

```
try:
    <блок, в котором перехватываются исключения>
[except [<исключение_1> [ as <объект исключения>]]:
    <блок, выполняемый при возникновении исключения>
...
[except [<исключение_N> [ as <объект исключения>]]:
    <блок, выполняемый при возникновении исключения>]]
[else: <блок, выполняемый, если исключение не возникло>]
[finally: <блок, выполняемый в любом случае>].
```

Инструкции, в которых перехватываются исключения, должны быть расположены внутри блока try. В блоке except в параметре <исключение_1> указывается класс обрабатываемого исключения.

Пример 10.1. Обработать исключение, возникающее при делении на ноль.

```
x = 0
try: # перехватываем исключения
    x = 1/0 # ошибка: деление на 0
except ZeroDivisionError: # указываем класс исключения
    print(x)

print("Обработали деление на 0")
```

```
0
Обработали деление на 0
```

Если в блоке try возникло исключение, то управление передается блоку except. В случае если исключение не соответствует указанному классу, управление передается следующему блоку except. Если ни один блок except не соответствует исключению, то исключение «всплывает» к обработчику

более высокого уровня. Если исключение нигде не обрабатывается в программе, то управление передается обработчику по умолчанию, который останавливает выполнение программы и выводит стандартную информацию об ошибке. Таким образом, в обработчике может быть несколько блоков except с разными классами исключений.

Пример 10.2. Обработать исключение, возникающее при делении на ноль с несколькими блоками except с разными классами исключений.

```
x = 0
try: # обрабатываем исключения
    try: # оложенном обработчик
        x = 1/0 # ошибка: деление на 0
    except NameError:
        print("Неопределенный идентификатор")
    except IndexError:
        print("Несуществующий индекс")
    print("Выражение после вложенного обработчика")
except ZeroDivisionError:
    print("Обработка деления на 0")

print(x)
```

```
Обработка деления на 0
0
```

Пример 10.3. Обработать исключение, возникающее при делении на ноль, в инструкции except указать сразу несколько исключений.

```
x = 0
try: # обрабатываем исключения
    x = 1/0 # ошибка: деление на 0
except (NameError, IndexError, ZeroDivisionError):
    x = 0
print(x)
```

```
0
```

Пример 10.4. Обработать исключение, возникающее при делении на ноль и получить информацию об обрабатываемом исключении можно через второй параметр в инструкции except.

```
x = 0
try: # обрабатываем исключения
    x = 1/0 # ошибка: деление на 0
except (NameError, IndexError, ZeroDivisionError) as err:
    print(err.__class__.__name__) # название класса исключения
    print(err) # текст сообщения об ошибке
```

```
ZeroDivisionError
division by zero
```

Если в инструкции except не указан класс исключения, то такой блок перехватывает все исключения. На практике следует избегать пустых

инструкций except, т.к. можно перехватить исключение, которое является лишь сигналом системе, а не ошибкой.

```
x = 0
try: # обрабатьшаем исключения
    x = 1/0 # ошибка: деление на 0
except:
    print(x) # выведем:0
```

```
0
```

Если в обработчике присутствует блок else, то инструкции внутри этого блока будут выполнены только при отсутствии ошибок. При необходимости выполнить какие-либо завершающие действия вне зависимости от того, возникло исключение или нет, следует воспользоваться блоком finally.

Пример 10.5. Обработать исключение, возникающее при делении на ноль с блоками else и finally.

```
x = 0
try: # обрабатываем исключения
    x = 1/0 # ошибка: деление на 0
except ZeroDivisionError:
    print("Деление на 0")
    x = 0
else:
    print('Блок else')
finally:
    print('Блок finally')
print(x) # выведем: 0
```

```
Деление на 0
Блок finally
0
```

При наличии исключения и отсутствии блока except инструкции внутри блока finally будут выполнены, но исключение не будет обработано. Оно продолжит «всплывание» к обработчику более высокого уровня. Если пользовательский обработчик отсутствует, то управление передается обработчику по умолчанию, который прерывает выполнение программы и выводит сообщение об ошибке.

```
x = 0
try: # обрабатьшаем исключения
    # x = 1/0 # ошибка: деление на
    x = 1/10
finally:
    print('Блок finally')
```

```
Блок finally
```

Классы встроенных исключений. Основное преимущество использования классов для обработки исключений заключается в возможности указания базового класса для перехвата всех исключений соответствующих классов-потомков. Например, для перехвата деления на ноль мы использовали класс ZeroDivisionError. Если вместо этого класса

указать базовый класс ArithmeticError, то будут перехватываться исключения классов FloatingPointError, OverflowError и ZeroDivisionError.

Рассмотрим основные классы встроенных исключений:

✓ BaseException — является классом самого верхнего уровня;

✓ Exception — именно этот класс, а не BaseException, необходимо наследовать при создании пользовательских классов исключений;

✓ AssertionError — возбуждается инструкцией assert;

✓ AttributeError — попытка обращения к несуществующему атрибуту объекта;

✓ EOFError — возбуждается функцией input() при достижении конца файла;

✓ IOError — ошибка доступа к файлу;

✓ ImportError — невозможно подключить модуль или пакет;

✓ IndentationError — неправильно расставлены отступы в программе;

✓ IndexError — указанный индекс не существует в последовательности;

✓ KeyError — указанный ключ не существует в словаре;

✓ Keyboardlnterrupt — нажата комбинация клавиш Ctrl+C (или Ctrl+Break в некоторых операционных системах);

✓ NameError — попытка обращения к идентификатору до его определения;

✓ Stoprteration — возбуждается методом __next__() как сигнал об окончании итераций;

✓ SyntaxError — синтаксическая ошибка;

✓ TypeError — тип объекта не соответствует ожидаемому;

✓ UnboundLocalError — внутри функции переменной присваивается значение после обращения к одноименной глобальной переменной;

✓ UnicodeDecodeError — ошибка преобразования последовательности байтов в строку;

✓ UnicodeEncodeError — ошибка преобразования строки в последовательность байтов;

✓ ValueError — переданный параметр не соответствует ожидаемому значению;

✓ ZeroDivisionError — попытка деления на ноль.

Инструкция assert. Инструкция assert возбуждает исключение AssertionError, если логическое выражение возвращает значение False. Инструкция имеет следующий формат:

```
assert <логическое выражение> [, <сообщение>].
```

Инструкция assert эквивалентна следующему коду:

```
if __debug__:
    if not <логическое выражение>:
        raise AssertionError(<Сообщение>).
```

Если при запуске программы используется флаг -o, то переменная __debug__ будет иметь ложное значение. Таким образом, можно удалить все инструкции assert из байткода.

```
try:
    x = -3
    assert x >= 0, "Сообщение об ошибке"
except AssertionError as err:
    print(err) # Выдает: Сообщение об ошибке
```

```
Сообщение об ошибке
```

Пример 10.6. Обойти и вывести на экран все элементы списка (например: [0, 1, 2, 3, 4]) при помощи итератора, функции next() и обработать исключительную ситуацию StopIteration.

```
m = [0, 1, 2, 3, 4, 12]
m = iter(m)
try:
    while True:
        print(next(m))
except StopIteration:
    print("Это конец")
```

```
0
1
2
3
4
12
Это конец
```

Пример 10.7. Реализовать функцию cathetus_2 , осуществляющую поиск второго катета по длине гипотенузы и первого катета с проверкой входных значений и генерацией исключений с содержательными сообщениями при получении некорректных параметров.

```python
def cathetus_2(k, g):
    assert type(k) == int, "Что-то не то с катетом"
    assert type(g) == int, "Что-то не то с гипотенузой"
    assert g > k, "Что-то не то с треугольником"

    import math

    return math.sqrt(g*g - k*k)
```

```python
def f(k, g):
    try:
        print(cathetus_2(k, g))
    except AssertionError as t:
        print(t)
```

```python
f(3, 5)
f(5, 5)
f("1", "5")
f(1, "5")
```

```
4.0
Что-то не то с треугольником
Что-то не то с катетом
Что-то не то с гипотенузой
```

Пример 10.8. Запросить у пользователя имя файла и попытаться открыть файл с этим именем. Прочитать и вывести первую строку из этого файла. Корректно отреагировать на ошибку, связанную с отсутствием этого файла и на другие потенциальные проблемы, которые могут возникнуть во время выполнения задачи.

```python
def redlion(i):
    try:
        f = open(i, mode='r')
        print(f.readline())
        f.close()
    except FileNotFoundError:
        print("404, lol")
```

```python
redlion("lol.txt")
```

```
404, lol
```

Задачи для лабораторной работы №10

Задача 10.1. Исключения. Пользовательские исключения. Инструкция assert.

1) Напишите функцию, которая возвращает факториал числа *n*. Проработать исключения, что аргумент должен быть целым, положительным и не равен 0.

2) Написать функцию сложения двух положительных чисел. Вызвать исключение AssertionError при вводе пользователем отрицательных чисел.

3) Напишите функцию, которая принимает имя файла. Обработать исключительную

2) ситуацию при условии, что такого файла не существует, или неверно задан путь к файлу.

3) Написать функцию сложения чисел. В функцию может передаваться любое количество чисел. Если аргумент не является число, то вызвать исключение.

4) Написать функцию вычисления 2/*s*. На вход функции поступает переменная *s*. Перехватить все исключительные ситуации.

5) Написать функцию вычисления квадратного уравнения. На вход функции поступают коэффициенты *a, b, c*. Перехватить все исключительные ситуации.

6) Написать функцию подсчета суммы главной диагонали. Если матрица не является

7) квадратной, должна вызываться ошибка.

8) Написать функцию подсчета суммы побочной диагонали. Если матрица не является квадратной, должна вызываться ошибка.

9) Написать функцию вычисления площади квадрата. Обработать исключительную ситуацию при условии, что стороны квадрата не равны.

10) Написать функцию вычисления периметра прямоугольника. Обработать исключительную ситуацию при условии, что стороны прямоугольника одинаковые.

11) Напишите функцию, которая принимает имя файла и дополнительно расширение файла. Если файл имеет расширение отличное от введенного, то возвращает текст «Файл имеет другое расширение», иначе возвращает содержимое файла.

12) Написать калькулятор для строковых выражений вида 'a <операция> b', где 'a' и 'b' — целые числа, а <операция> — одна из операций +, –, *, / (деление нацело), % (остаток от деления), ^ (возведение в степень). Пример calc('13 – 5') → 8. Проверять переданную строку на корректность ввода и выводить отладочную ошибку.

13) В функцию передаются три кортежа — координаты *x, y* точек. Необходимо найти площадь треугольника. Данные проверять в режиме отладки.

14) Элементами матрицы являются случайные положительные целые числа из заданного диапазона. Число строк и столбцов матрицы задается с клавиатуры. Написать функцию подсчета среднего арифметического элементов над главной диагональю и количество четных элементов под ней. Если матрица не является квадратной, должно генерироваться исключение.

16) Написать функцию сложения двух положительных чисел. Вызвать исключение AssertionError при вводе пользователем отрицательных чисел.

17) Написать функцию вычитания двух положительных чисел. Вызвать исключение AssertionError при вводе пользователем отрицательных чисел

18) Написать функцию деления двух положительных чисел. Вызвать исключение ZeroDivisionError при делении на ноль. 18) Написать функцию деления числа на себя. Вызвать исключение ValueError при попытке деления на ноль.

19) Написать функцию, которая принимает список чисел и возвращает их сумму. Обработать исключение TypeError если в переданном списке присутствуют нечисловые значения.

20) Написать функцию, которая принимает строку и возвращает количество символов в строке. Обработать исключение TypeError если в переданном аргументе не строка.

21) Создать функцию для чтения текстового файла и возврата его содержимого. Обработать исключение FileNotFoundError если файл не найден.

22) Написать функцию для возведения числа в степень. Обработать исключение TypeError если аргументы не являются числами.

23) Написать функцию, которая принимает список и возвращает его первый элемент. Обработать исключение IndexError если список пуст.

Список рекомендуемой литературы

1. Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 768 с.

2. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс). – ISBN 978-5-534-10971-9. – Текст: электронный // ЭБС Юрайт [сайт]. – URL: https://urait.ru/bcode/437489

3. Шелудько, В. М. Основы программирования на языке высокого уровня Python: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 146 с. – ISBN 978-5-9275-2649-9. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: http://www.iprbookshop.ru/87461.html

4. Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 107 с. – ISBN 978-5-9275-2648-2. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: http://www.iprbookshop.ru/87530.html

КОЧКАРОВ Ахмат Магомедович
СЕЛИМСУЛТАНОВА Рита Ильясовна

СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие для обучающихся на 2 курса
направление подготовки 01.03.02 Прикладная математика
и информатика (очной формы обучения)

Корректор Чагова О. Х.
Редактор Чагова О.Х.

Сдано в набор 30.08.2024 г.
Формат 60х84/16
Бумага офсетная
Печать офсетная
Усл. печ. л. 4,41
Заказ № 4962
Тираж 100 экз.

Оригинал-макет подготовлен
в Библиотечно-издательском центре СКГА
369000, г. Черкесск, ул. Ставропольская, 36