Федеральное государственное образовательное бюджетное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Факультет информационных технологий и анализа больших данных

Департамент анализа данных и машинного обучения

Е.П. Догадина

СБОРНИК ЗАДАЧ ПО ДИСЦИПЛИНЕ «КОМПЬЮТЕРНЫЙ ПРАКТИКУМ»

1, 2 семестр

Для студентов,
обучающихся по направлениям
10.03.01 «Информационная безопасность»,
38.03.05 «Бизнес-информатика»
(программа подготовки бакалавра)

Москва 2021

Федеральное государственное образовательное бюджетное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Факультет информационных технологий и анализа больших данных

Департамент анализа данных и машинного обучения

Е.П. Догадина

СБОРНИК ЗАДАЧ ПО ДИСЦИПЛИНЕ «КОМПЬЮТЕРНЫЙ ПРАКТИКУМ»

1, 2 семестр

Для студентов,
обучающихся по направлениям
10.03.01 «Информационная безопасность»,
38.03.05 «Бизнес-информатика»
(программа подготовки бакалавра)

Одобрено Советом Департамента анализа данных и машинного обучения (протокол №11 от 25.05.2021 г.)

Москва – 2021

УДК 004.432
ББК 32.973
Д59

Авторы:

Догадина Е.П., канд. тех. наук, доцент, доцент Департамента анализа данных и машинного обучения факультета информационных технологий и анализа больших данных Финансового университета при Правительстве РФ

Рецензенты:

Коротеев М.В., канд. эк. наук, доцент Департамента анализа данных и машинного обучения факультета информационных технологий и анализа больших данных Финансового университета при Правительстве РФ

Д59 Догадина Е.П.

Сборник задач по дисциплине «Компьютерный практикум», 1, 2 семестр. Для студентов, обучающихся по направлениям 10.03.01 «Информационная безопасность», 38.03.05 «Бизнес-информатика», (программа подготовки бакалавра) – М.: Финансовый университет, департамент анализа данных и машинного обучения, 2021. – 102 с.

Цель методических указаний – предоставление необходимого методического обеспечения по выполнению практических заданий по дисциплине «Компьютерный практикум». Методические указания содержат краткие теоретические сведения, примеры решения задач, тестовые задания и задачи по основным разделам дисциплины «Компьютерный практикум».

Методические указания предназначены для бакалавров направления подготовки 10.03.01 «Информационная безопасность» и 38.03.05 «Бизнес-информатика».

УДК 004.432
ББК 32.973

Учебное издание

,
Догадина Елена Петровна

Сборник задач по дисциплине «Компьютерный практикум»

Для студентов, обучающихся по направлениям 10.03.01 «Информационная безопасность», 38.03.05 «Бизнес-информатика», (программа подготовки бакалавра)

Компьютерный набор, верстка Е. П. Догадина

Формат 60x90/16. Гарнитура *Times New Roman.*
Усл. п.л. 6,375. Изд. № - 2021. Заказ № ____
Электронное издание

Оглавление

Предисловие

Сборник задач предназначен для студентов, обучающихся по направлениям 10.03.01 «Информационная безопасность» и 38.03.05 «Бизнес-информатика», по дисциплине «Компьютерный практикум».

В задачнике содержатся материалы, необходимые для освоения студентами компетенций. В описание включены материалы по разработке программных продуктов на языке *Python*. Рассматриваются вопросы различных типов данных и инструкций в языке *Python*, работы с функциями и файлами, модульного и объектно-ориентированного программирования на *Python*. По каждой теме представлены примеры решения задач различного уровня сложности.

1. Числовые и строковые типы данных языка Python. Управляющие конструкции.

1.1. Работа с числами. Базовые числовые типы *int* и *float*

Для преобразования чисел из вещественных в целые и наоборот в *Python* определены функции *int*() и *float*(). Например, *int*(12.6) даст в результате 12, а *float*(12) даёт в результате 12.0 (десятичный разделитель — точка).

Представим математические операции над целыми и вещественными типами данных:

+ - сложение.

– - вычитание.

* - умножение.

/- деление. Результатом деления всегда является вещественное число**,** даже если производится деление целых чисел.

//- деление с округлением вниз. Вне зависимости от типа чисел остаток отбрасывается.

% - остаток от деления.

** -возведение в степень.

унарный - (минус) и унарный + (плюс).

При вычислении математических выражений действует общепринятый приоритет выполнения операций.

Кроме того, в *Python* для операций с числами используются функции *abs*() (вычисление абсолютного значения — модуля, *abs*(-5)=5), *pow*() (возведение в степень, *pow*(2,4) =16), *round*() (округление, *round*(15/4) =4, а *round*(15/4,1)=3,8). Эти функции являются «встроенными», что означает, что для их использования нет необходимости подключать дополнительные модули. Все прочие математические функции требуют подключения модуля *math*. Этот модуль содержит дополнительные функции для работы с числами, а также стандартные математические константы.

Для определения типа числа необходимо использовать *type*. Например, *type*(42.1)= *float*, *type*(2147483647)= *int*.

Пример 1.1.

Вычислите сумму цифр трехзначного числа.

```
In [4]: d=int(input())
        a=d%10   # определение единиц числа
        b=d//10%10   # определение десятков числа
        c=d//10//10%10   # определение сотен числа
        s=a+b+c
        print(s)

        358
        16
```

Python поддерживает только однострочные комментарии. В программе комментарии обозначаются знаком #.

Задачи

1. Значения переменных *A* и *B* ввести с клавиатуры и вывести на экран. После этого значения меняются местами, т.е. *A* нужно присвоить значение *B*, а *B* – значение *A*, и вновь значения переменных вывести на экран.

2. Значение x вводится с клавиатуры. Вычислите $y = (x+1)^2 * (\sqrt{x^3+1})$. Выведите на экран значения *x* и *y* с 3 знаками после запятой.

3. Вычислите сумму цифр пятизначного числа.

4. Для заданного трехзначного числа выведите число, у которого цифры идут в обратном порядке, например, для числа 123 ответ 321.

5. Ввести координаты 2 точек: (x_1, y_1) и (x_2, y_2). Вычислите расстояние между этими точками. Результат выведите с 5 знаками после запятой.

6. Разработать программу вычисления по известному радиусу площади круга и длины окружности.

7. Разработать программу по вычислению площади кольца по известным значениям его внешнего и внутреннего радиусов.

8. Разработать программу вычисления радиуса круга и его площади по известной длине окружности.

9. Разработать программу вычисления по известному радиусу объема шароподобного тела и длине его экваториальной параллели.

10. Имеются два *n* – мерных вектора x и y, которые задают координаты *n* точек на плоскости (случайные целые числа). Найти наиболее близкие друг другу точки.

11. Треугольник задан координатами своих вершин: (x_1, y_1), (x_2, y_2), (x_3, y_3). Значения переменных $x_1, y_1, x_2, y_2, x_3, y_3$ определите с помощью присваивания. Они могут быть нецелыми. Найти периметр и площадь треугольника.

12. Пусть пользователь вводит 5 чисел: *a, b, c, d, e*. Реализуйте программу расчета выражения вида $res = \frac{(a+bc)}{2d-e} + Ost[b^e / c]$, где *Ost* – означает остаток от деления. Учесть невозможность деления на 0.

1.2. Строки, функции и методы работы со строками

Строка (объект класса *str*) – это последовательность символов. Существует ряд специальных символов для работы со строками. Наиболее часто используются последовательности: \n – перевод строки, \r - возврат каретки, \t – табуляция, ' – апостроф, " – кавычки, \f - перевод формата.

Строку, введенную между утроенными апострофами или утроенными кавычками, можно разместить на нескольких строках, а также одновременно использовать кавычки и апострофы без необходимости их экранировать.

```
In [2]: print('''Строка1
        Одинарные кавычки '
        Двойные кавчки "
        Строка2 ''')
```

```
Строка1
Одинарные кавычки '
Двойные кавчки "
Строка2
```

Строки являются неизменяемыми последовательностями (массивами) и все функциональные возможности, существующие у неизменяемых последовательностей, могут использоваться и у строк.

Существует три формы оператора получения среза: *seq*[*start*], *seq*[*start*:*end*], *seq*[*start*:*end*:*step*], где *start, end, step* – индекс начала строки, индекс конца строки и шаг соответственно. Индексирование строк начинается с 0 и до (длины строки -1). Возможна отрицательная нумерация с конца строки.

Основные операции над строками являются:

+ конкатенация (операция позволяет объединить две строки в одну);

* повторение (повторяет строку указанное количество раз);

in проверка вхождения подстроки в строку;

Методы строк:

strip( ) - удаляет пробельные или указанные символы в начале и конце строки.

join() - преобразует последовательность (в частности список) в строку. Элементы добавляются через указанный разделитель. Формат метода: <Строка> = <Разделитель>.join(<Последовательность>).

find()- ищет подстроку в строке.

isdigit() - возвращает *True*, если строка содержит только цифры, в противном случае - *False*;

isalpha() - возвращает *True*, если строка содержит только буквы, в противном случае- *False*. Если строка пустая, то возвращается значение *False*.

islower()- возвращает *True*, если строка содержит буквы, и они все в нижнем регистре, в противном случае- *False*.

isupper() - возвращает *True*, если строка содержит буквы, и они все в верхнем регистре, в противном случае- *False*.

split() – разделяет строку на подстроки по указанному разделителю и возвращает созданный из них список.

index() – метод аналогичен методу *find*(), но если образец в строку не входит, то возникает ошибка.

replace() – создает новую строку, в которой фрагмент исходной строки, указанный в первом аргументе, заменяется на строку, указанную во втором аргументе. Третий аргумент определяет количество замен. По умолчанию заменяются все вхождения.

Для изменения регистра символов используют *upper*(), *lower*(), *title*(), *swapcase*(), *capitalize*(), *title*().

Пример 1.2.

In [2]:
```
#1.*Получить переменную, содержащую строку со значением, равным двум в одиннадцатой степени
#и вывести ее на экран.
s = str(2**11)
print(s, type(s))
```

```
2048 <class 'str'>
```

Пример 1.3.

In [3]:
```
#2.*Создать строковую переменную, содержащую текст:
#line1
#line2
#line3
z1='''line1
line2
line3'''
print(z1)
z1='line1 \nline2 \nline3'
print(z1)
```

```
line1
line2
line3
line1
line2
line3
```

Пример 1.4.

```
In [5]: #4.·Вывести 5й с начала и 5й с конца символ из строки 'Hello world'
        st = 'Hello world'
        print(st[4],' ', st[-5])
```

```
o   w
```

Пример 1.5.

```
In [4]: #3.·Создать строковую переменную, содержащую текст:
        #\\\\
        #\\\
        #\\
        #\
        z1='\\\\\\\ \n\\\\\ \n\\\ \n\\'
        print(z1)
```

```
\\\\
\\\
\\
\
```

Пример 1.6.

```
In [6]: #6.·Вывести символ, находящийся в середине заранее неизвестной строки,
        #полученной с помощью функции input()
        s=input()
        print(s[len(s)//2])
```

```
ДАДиМО
и
```

Пример 1.7.

```
In [7]: #7.·Вставить между всеми символами строки 'Hello world' стрелочку '->'
        #и сохранить результат в строковой переменной
        #(ожидаемый результат: 'H->e->l->l->o-> ->w->o->r->l->d')
        s='Hello world'
        '->'.join(s)
```

```
Out[7]: 'H->e->l->l->o-> ->w->o->r->l->d'
```

Задачи

1. Введите строку, состоящую из 2 цифр. Преобразуйте ее в целое и вещественное число. Выведите полученные 3 значения (строку, целое число, вещественное число) на экран в одной строке через запятую, затем пропустите строку и вновь выведите значения по одному на строке. Перед каждым значением выведите его тип.
2. Для строки 'Компьютерный практикум' двумя способами получить подстроку с 1го по 4й символы включительно.
3. Составить строку из всех четных символов строки 'Компьютерный практикум'.
4. Получить подстроку неизвестной заранее строки, содержащую половину символов строки и расположенную по середине строки.
5. Коротко записать создание строки 'oneoneoneoneoneonetwotwotwo'
6. Для введенной строки выведите (на отдельной строке):

• второй символ этой строки;

- предпоследний символ этой строки;
- первые 3 символа этой строки;
- всю строку, кроме последних двух символов;
- все символы с четными индексами (считая, что индексация начинается с 0, поэтому символы выводятся, начиная с первого);
- все символы с нечетными индексами, то есть начиная со второго символа строки;
- все символы в обратном порядке;
- все символы строки через один в обратном порядке, начиная с последнего;
- длину данной строки.

7. Не используя метод count, для заданной строки выполните:

- если символ * в данной строке отсутствует, выведите текст «нет символа»;
- если символ * встречается в строке только один раз, выведите его индекс;
- если символ * встречается два и более раз, выведите индекс его первого, второго и последнего вхождения, удалите первый и последний символ * из строки.

8. Дана строка, состоящая из слов, разделенных пробелами. В этой строке:

- удалите все лишние пробелы (в начале, в конце, между словами оставить ровно один пробел);
- поменяйте регистр символов (строчные сделать прописными, прописные – строчными);
- определите, сколько в ней слов.

9. Строка содержит фамилию, имя и отчество, записанные через пробелы. Например « Иванов Иван Иванович». Для этой строки:

- выведите информацию на экран в виде:

Фамилия	Иванов
Имя	Иван
Отчество	Иванович

- получите строки вида «Иванов И.И.» и «И.И. Иванов»

10. Напишите программу, которая осуществляет вывод кошки на экран.

Примечание: кошка выглядят примерно так

```
 /\__/\
/@ @\
(~~0~~)
```

11. Для двух произвольных строк провести сравнение на совпадение содержимого без учета регистра букв и начальных и конечных пробельных символов. Например, для двух

строк ' HeLLO WOrlD' и 'hello WORLD ' такое сравнение должно возвращать True (содержание строк совпадает).

1.3. Операторы сравнения. Логические операторы. Инструкция ветвления *if...else*

В языке определены следующие операторы сравнения:

a == b # проверка на равенство

a != b # проверка на неравенство

a < b # меньше

a <= b # меньше или равно

a > b # больше

a >= b # больше или равно

Для реализации в коде условий, при соответствии которым выполняется та или иная ветвь кода, применяются специальные операторы.

Синтаксис условного оператора:

if *выражение1:*
    инструкции1
elif *выражение2:*
    инструкции1
elif *выражениеN:*
    инструкцииN
else*:*
    инструкции

Части *elif* и *else* могут отсутствовать.

Общий вид синтаксиса тернарного условного оператора:

true_result ***if*** *инструкция1* ***else*** *false_result*

Пример 1.8.

Использование тернарного условного оператора.

```
In [2]: # пример:
        a = 15
        b = 8
        print(a if a > 5 else b)

        15

In [3]: c = a if a > 0 else 0
        c

Out[3]: 15
```

Пример 1.9.

Напишите программу, которая при попадании значения в диапазон печатает «Yes», в противном случае – «No». Границы диапазона и значение вводятся.

```
In [4]: x=float(input('Значение:'))
        y=float(input('Нижняя граница:'))
        z=float(input('Верхняя граница:'))
        if (y<x<z):
            print ('Yes')
        else:
            print ('No')
```

```
Значение:5
Нижняя граница:3
Верхняя граница:15
Yes
```

Пример 1.10.

Пользователь поочередно вводит 2 строки. Проверить, что первая строка длиннее второй, но при этом находится ближе в лексикографическом порядке. Подобрать пару строк, удовлетворяющую условию.

```
In [6]: a=input()
        b=input()
        print((len(a)>len(b))and(a<b))
```

```
Компьютерный практикум
ФИТ
True
```

Задачи.

1. Определить время года по номеру месяца.
2. Определить минимальное значение среди чисел *a, b, c, d* (не использовать стандартные функции *max* и *min*).
3. Напишите программу, которая переводит оценку из 100-балльной системы в пятибалльную по правилам, принятым в университете.
4. Вычислить значение *F*: *F*=1, если цифра 7 входит в запись заданного трехзначного числа, и 0 – в противном случае.
5. Даны два отрезка [*a*;*b*] и [*c*;*d*]. Найдите их пересечение. Если отрезки не пересекаются, то выдайте сообщение.
6. По номеру года определите, является ли данный год високосным. (год является високосным, если его номер кратен 4, но не кратен 100, а также если он кратен 400).
7. Для отрезков длины *a, b, c* определить, можно ли из них составить треугольник и является ли этот треугольник прямоугольным.

8. Напишите программу для решения уравнения $ax^2 + bx + c = 0$. Коэффициенты *a, b, c* могут быть любыми числами.

9. Вводится целое число. Выведите его на экран и допишите к нему слова «рубль», «рубля» или «рублей» в зависимости от значения. Алгоритм:

• исключение: если число оканчивается на 11, 12, 13 или 14, добавляем слово «рублей»;

• если число оканчивается на 1, добавляем слово «рубль»;

• если число оканчивается на 2, 3 или 4, добавляем слово «рубля»;

• если число оканчивается на цифры 5, 6, 7, 8, 9 или 0, добавляем слово «рублей».

10. Пользователь поочередно вводит координаты точки в декартовой системе координат. Определить, какой четверти принадлежит данная точка или на какой оси она находится. Расположение точки вывести на экран. Найти произведение номера четверти на расстояние от этой точки до начала координат и вывести его на экран. Если точка лежит на оси, считать, что номер четверти равен 0.

11. Выведите значение заданного целого числа от 0 до 999 прописью. Например, «сто девяносто один» для числа 191, «одиннадцать» для числа 11.

12. Вычислите значение выражения, которое состоит из целых чисел и знаков «+» и «-». Выражение вводится как символьная строка.

13. Пользователь поочередно вводит 2 строки. Определите, какая строка длиннее и на сколько символов, а какая строка стоит раньше в лексикографическом порядке. Собрать результирующую строку, которая бы содержала 1 и 2 строки, разделенные переносом строки.

14. Реализовать калькулятор, который принимает от пользователя через пробел строку следующего вида: «*a op b*», где *a* и *b* – некоторые числа, а ‘*op’* определяет оператор и может принимать значения «+,-,*,/,**,%». В зависимости от оператора с помощью форматирования строк вывести результат в виде: «*a* + *b* = 3 + 2 = 5», где была получена строка «3 + 2». Для операций возведения в степень и деления по модулю использовать вместо знака оператора соответствующие выражения. Например, для строки «2 ** 3» должно быть выведено «a в степени b = 2 в степени 3 = 8».

15. Напишите программу, которая в зависимости от введенного пользователем числа *N*, осуществляет вывод *N* кошек на экран. Пусть 1<= *N* <=10.

Примечание: кошки при *N*=3 выглядят примерно так

```
 /\__/\    /\__/\    /\__/\
/@ @\     /@ @\     /@ @\
(~~0~~)  (~~0~~)  (~~0~~)
```

16. Имеются два *n* – мерных вектора x и y, которые задают координаты *n* точек на плоскости (случайные целые числа). Найти наиболее близкие друг другу точки.

17. Из множества целых чисел от 1 до *N* выделить множество *N*2 числа, кратные 2, множество *N*3 – кратные 3, множество *N* 6 – кратные 6 (т.е. кратные и 2 и 3), множество *N* 23 – кратные либо 2, либо 3.

18. Даны *m* (*m*>1) слов. Найти общее количество заданной буквы в этих словах.

19. Даны *m* (*m*>1) слов. В каком из них доля (в %) заданной буквы больше.

20. Написать функцию вычисления y с учетом заданного шага *h*:

$$y = \begin{cases} 3x^2 & x > 4{,}5 \\ e^{-x} & 1 \le x \le 4{,}5, \\ -cos^2(2x) & x < 1 \end{cases} \qquad x \in [-\frac{\pi}{2}; 2\pi]$$

1.4. Инструкция цикла *while*

Цикл *while* реализуется следующим образом:

while *выражение:*
  инструкции
else*:*
  инструкции_else

Блок *else* может отсутствовать.

Цикл выполняется следующим образом. Вычисляется значение выражения в заголовке цикла. Если оно имеет значение *True* (условие выполняется), то выполняются инструкции тела цикла. После этого вновь возвращаемся к заголовку цикла и проверяем условие. Если условие в заголовке цикла имеет значение *False*, то цикл завершает работу. Далее будут выполнены инструкции, указанные после *else*, если они есть.

Использование инструкции *break* в теле цикла приводит к немедленному прекращению цикла, при этом не выполняется ветка *else*. Следующей будет выполняться инструкция, следующая сразу за циклом.

Использование инструкции *continue* в теле цикла приводит к тому, что все оставшиеся инструкции тела цикла пропускаются, происходит переход на строку заголовка и проверка условия цикла. Далее все выполняется как обычно

Пример 1.11.

Рассчитать двойной факториал для произвольного числа. Двойной факториал *n!!* числа n рассчитывается как произведение всех чисел, меньших исходного на числа, кратные двум (вплоть до 1 или 2). Например, 7!! = 7 * 5 * 3 * 1 = 105

```
In [7]: x = int(input())
        s = 1
        while x > 0:
            s *= x
            x -= 2
        s
```

```
7
```

```
Out[7]: 105
```

Пример 1.12.

Реализовать проверку на ввод вводимого пользователем значения: предлагать ввод до тех пор, пока не будет введена непустая строка. Если введено DZ, выводить сообщение "Homework". Если введенная строка в лексикографическом порядке стоит перед строчными латинскими символами, выводить сообщение "ClassWork" и предлагать ввод снова. Иначе выводить отформатированную исходную строку с заполнителем "*" и шириной 30 символов с выравниванием посередине.

```
In [11]: s = ' '
         while s != '':
             s = input()
             if s == 'DZ':
                 print('Homework')
                 break
             elif s < 'a':
                 print("ClassWork")
                 continue
             else:
                 print('{:_^30s}'.format(s))
                 continue
```

```
hi
_____________hi_____________
Hi
ClassWork
DZ
Homework
```

Задачи.

1. Проверить, является ли введенное целое число простым.
2. Напишите программу, которая выводит на экран значения функции $sin(x)$ на заданном интервале $[a,b]$ с шагом 0.1 в виде таблицы, состоящей из двух столбцов. В первом столбце выводится значение x, во втором – значение $sin(x)$. Все значения выводятся с 2 знаками после запятой.
3. Напишите программу, которая вводит с клавиатуры числа до тех пор, пока не будет введено число 0. В конце работы программы на экран выводятся количество чисел (0 не считаем), минимальное и максимальное из введенных чисел, их сумму и среднее значение.

4. Напишите программу, которая вычисляет минимальное целое положительное число среди введенных значений. Признаком конца ввода является символ «.».

5. Напишите программу, которая находит наибольший общий делитель двух чисел, используя модифицированный алгоритм Евклида: нужно заменять большее число на остаток от деления большего на меньшее до тех пор, пока этот остаток не станет равен нулю; тогда второе и есть НОД.

6. Известна денежная сумма. Выдать её купюрами 500, 100, 10 и монетой 1 руб., если это возможно. Например, для выдачи суммы 3875 рублей, потребуется 7 купюр по 500 рублей, 3 купюры по 100 руб., 7 монет по 10 рублей и 5 монет по 1 рублю.

1.5. Инструкция цикла *for*

Синтаксис цикла вида *for*:

for *переменная* **in** *объект*:
    инструкции
else:
    инструкции_else

Блок *else* может отсутствовать.

Для цикла *for* применимы функции *range* и *enumerate*.

Функция *range* возвращает целочисленный итератор. Способы обращения к функции *range*:

- *range*(*stop*) – с одним аргументом (*stop*) итератор представляет последовательность целых чисел от 0 до (*stop* - 1).
- *range* (*start, stop*) – с двумя аргументами (*start, stop*) итератор представляет последовательность целых чисел от *start* до (*stop* - 1).
- *range* (*start, stop, step*) – с тремя аргументами итератор представляет последовательность целых чисел от *start* до (*stop* - 1) с шагом *step*.

Пример 1.13.

Вывести на экран последовательность от 3 до 6 с шагом 1. Вывести на экран последовательность от 3 до 9 с шагом 2.

```
In [4]: for i in range(3, 7):
            print(i)

3
4
5
6
```

```
In [5]: for i in range(3,10,2):
            print(i)

3
5
7
9
```

Пример 1.14.

Вычислить сумму четных чисел от 1 до *n*. Значение *n* вводится с клавиатуры.

```
In [9]: n=int(input())
        s=0
        for i in range(1,n+1):
            if i%2==0:
                print("число для сложения ",i)
                s=s+i
        print("сумма ", s)

10
число для сложения  2
число для сложения  4
число для сложения  6
число для сложения  8
число для сложения  10
сумма  30
```

Функция *enumerate* обычно используется в циклах *for..in*, чтобы получить последовательность кортежей (*index, item*), где значения индексов отсчитывается от 0 или от значения *start*;

Синтаксис функции *enumerate*:

- *enumerate*(*i*) генерируется последовательность кортежей (*index, item*), где значения индексов отсчитывается от 0.
- *enumerate*(*i, start*) генерируется последовательность кортежей (*index, item*), где значения индексов отсчитывается от значения *start.*

Пример 1.15.

Вывести пару (элемент и его индекс) с помощью *enumerate*. Значения индексов отсчитывать с 0 и с 1.

In [2]:
```
st = 'fa.ru'
for index, item in enumerate(st):
    print(f"Символ '{item}' имеет индекс {index}")
```

```
Символ 'f' имеет индекс 0
Символ 'a' имеет индекс 1
Символ '.' имеет индекс 2
Символ 'r' имеет индекс 3
Символ 'u' имеет индекс 4
```

In [3]:
```
st = 'fa.ru'
start=1
for index, item in enumerate(st,start):
    print(f"Символ '{item}' имеет индекс {index}")
```

```
Символ 'f' имеет индекс 1
Символ 'a' имеет индекс 2
Символ '.' имеет индекс 3
Символ 'r' имеет индекс 4
Символ 'u' имеет индекс 5
```

Задачи.

1.Напишите программу, которая для введенного *n* вычисляет значение *n*! (не использовать функции *Python*).

2. Напишите программу, которая вводит с клавиатуры 5 чисел и вычисляет их сумму, произведение и среднее значение.

3. Имеется 2 банка. Первый банк ежемесячно начисляет 1.1% от первоначально вложенной суммы (простые проценты). Второй банк ежемесячно начисляет 1% от накопленной к этому моменту суммы (сложные проценты). В каждый из банков в месяц 0 (начальный момент) положили 100 000 руб. Напишите программу, которая выводит на экран в виде таблицы, накопленные в этих банках суммы для 24 месяцев. Определите в программе и выведите на экран в какой банк выгоднее вложить деньги на 1 год и на 2 года.

4. С клавиатуры вводятся n целых чисел. Найти среди них наименьшее нечетное число.

5. С клавиатуры вводятся n целых чисел. Найти минимальное положительное и максимальное отрицательное числа.

6. С клавиатуры вводятся n целых чисел. Вычислить сумму положительных чисел.

7. С клавиатуры вводятся n целых чисел. Вычислить сумму отрицательных чисел, кратных трем.

8. Известна денежная сумма. Выдать её купюрами 500, 100, 10 и монетой 1 руб., если это возможно. Например, для выдачи суммы 3875 рублей, потребуется 7 купюр по 500 рублей, 3 купюры по 100 руб., 7 монет по 10 рублей и 5 монет по 1 рублю.

9. Разработать программу вычисления дня недели для произвольной даты, например, 9 мая 1945 года, 12 апреля 1961 года.

10. Для заданной математической функции $y=f(x)$ разработать программу вычисления значения функции на интервале от 3,1 до 6.

$$y = \frac{x^2 - 9x + 7}{x - 3}$$

Тесты по разделу 1

1. Правильные утверждения относительно понятия и обработки строк в программе на языке Python:

1) len(s) - вычисление длины строки s как числа символов
2) s1+s2 - конкатенация или сцепление строк - присоединение к концу строки s1 строки s2
3) s[i] – выбор i-го по порядку символа строки
4) max(s) - вычисление и вывод символа с наибольшим значением (кодом) в строке
5) min(s) - вычисление и вывод символа с наименьшим значением (кодом) в строке

2. Транслятор языка программирования Python представляет собой:

1) Интерпретатор
2) Компилятор
3) Интерпретатор со встроенным транслятором (смешанная реализация интерпретатора и компилятора)
4) Динамическая компиляция

3. Правильные утверждения относительно операции вывода данных в программе на языке Python:

1) Для вывода данных на экран используется функция *write*()
2) Выводимые данные в функции *print*(), указываемые в качестве ее параметров, разделяются запятыми
3) Выведенные на экран данные по умолчанию разделяются одним пробелом
4) Параметр *sep*=”, “, указанный в *print*(), разделяет выведенные на экран данные указанными символами: в утверждении - запятой и двумя пробелами
5) Параметр *end*=”*n*”, указанный в *print*(), переводит курсор в области вывода на новую строку

4. Установить соответствие между побитными операторами Python и их описанием:

Оператор	Описание
<<	Побитный сдвиг вправо на заданное количество

&	Побитный сдвиг влево на заданное количество
|	Побитная операция И над числами
>>	Побитная операция ИЛИ над числами

5. *Правильные утверждения относительно нелинейных конструкций языка Python:*

1) Разветвления в программе реализуются с помощью оператора *if*
2) Оператор *while* реализует цикл с заранее неизвестным числом повторений
3) Оператор *for* реализует цикл с заранее известным числом повторений
4) Оператор *while* реализует цикл с верхним окончанием
5) В языке *Python* есть оператор построения цикла с нижним окончанием

6. *Установить соответствие между операторами Python и их описанием:*

Оператор	Описание
**	Умножение
%	Целочисленное деление
*	Возведение в степень
//	Остаток от деления

7. *Определите соответствие между значениями в столбцах получив четыре пары значений цифра-буква.*

1)	int(x, 10)	a) 0x9ff
2)	bin(x)	b) 0o177
3)	hex(x)	c) 0b101010
4)	oct(x)	d) 00101

8. *Результат вычисления выражения в Python:*

0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1

1) 0.9999999999999999
2) 1.0000000000000000
3) 0.99999999999999
4) 1.00000000000000

9. *Поддерживают ли числовые типы int и float длинную арифметику?*

1) поддерживают int и float
2) поддерживает только тип int
3) поддерживает только тип float
4) не поддерживают int и float

10. Правильные утверждения относительно операции ввода данных в программу на языке Python:

1) Для ввода данных используется функция *input*()
2) Значением функции *input*() является строка символов
3) Результат функции *input* () следует присвоить вводимым переменным
4) Одной функцией *input* () с использованием метода *split*() нельзя ввести значения нескольких переменных, разделенных при вводе пробелами
5) Результат ввода данных следует преобразовать к требуемому типу с использованием одноименных функций *int*(), *float*(), *complex*()

11. Правильные утверждения относительно имени переменной в языке Python:

1) Имя переменной может состоять из букв, цифр и символа подчеркивания
2) Имя не может начинаться с цифры
3) В именах нужно учитывать регистр букв
4) В качестве имен нельзя использовать ключевые слова языка
5) Строчные и прописные буквы, указанные в именах, идентичны

12. Поставить в соответствие функции и ее значения:

1) *chr*()	a) Преобразует целое число в символ
2) *ord*()	b) Преобразует символ в целое число
3) *len*()	c) Возвращает длину строки
4) *str*()	d) Изменяет тип объекта на *string*

13. Определите соответствие методов строки string выполняемых ими действиям:

1) *string.strip*()	a) состоит ли строка из букв
2) *string.isspace*()	b) состоит ли строка только из пробельных символов
3) *string.split*()	c) состоит ли строка из цифр или букв
4) *string.isdigit*()	d) состоит ли строка из цифр
5) *string.isalpha*()	e) удаляет символы с левого и правого края строки
6) *string.isalnum*()	f) делит строку на список из подстрок

14. *Каждый объект в Python имеет три атрибута – это*

_____.

15. *Список ключевых слов в Python можно получить, подключив модуль* _____ .

16. *Результат выполнения команды* `>>> float(True)`

равен _____ .

17. *Результат выполнения команды* `>>> x=int('37',8);x`

равен _____ .

18. *Результат выполнения кода:*

```
s='??five++--??'
x=s.rstrip('?+-')
print(x)
```

равен _____.

19. *Результат выполнения команды* `>>> bool(0) and bool(1)`

равен _____.

20. *Результат выполнения команды* `>>> bool(0) or bool(1)`

равен _____.

21. *Представленный код предназначен для проверки того, что введенное число* является _____.

```
x=abs(int(input('Введите целое число: ')))
d=x//2
while d>1:
    if x%d==0:
        print('x имеет делители')
        break
    d-=1
else:
    print('x=',x, '- _____ число')
```

22. *Определите правильные операции сравнения:*

1) x < y
2) x <= y
3) x =< y
4) x > y

5) x <> y

23. Определите правильные логические операторы для переменных логического типа x и y (принимающих значение True или False).

1) x or y
2) x & y
3) x || y
4) x and y
5) x && y

24. Какие из выражений в инструкции ветвления допустимы в языке Python:

1) " if "
2) " ifelse "
3) " if else "
4) " elseif "
5) " else if "

25. Какие из выражений в инструкции цикла while допустимы в языке Python:

1) " do while"
2) " while"
3) " while do"
4) " else"
5) " index"

26. Какие из выражений в инструкции цикла for допустимы в языке Python:

1) " for"
2) " else"
3) " index"
4) " break"
5) " stop"

2. Списки, кортежи, множества и словари.

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

Способы создания списка:

- явно указав все элементы списка в тексте (элементы списка перечисляются в квадратных скобках через запятую)

```
In [2]: lst=[] #пустой список
        lst1=[1,2,5,88] #список, состоящий из чисел
        lst2=['a','bb','ccc']  #список, состоящий из строк
        lst3 = [2, True, 'my string', '7'] #список, состоящий из элементов различных типов
        print(lst)
        print(lst1)
        print(lst2)
        print(lst3)

        []
        [1, 2, 5, 88]
        ['a', 'bb', 'ccc']
        [2, True, 'my string', '7']
```

- с помощью функции *list*()

```
In [6]: lst=list() #пустой список
        lst1=list(range(1,5)) #список чисел
        lst2=list('семинар')  #список строк
        print(lst)
        print(lst1)
        print(lst2)

        []
        [1, 2, 3, 4]
        ['с', 'е', 'м', 'и', 'н', 'а', 'р']
```

- поэлементное заполнение списка, используя метод *append*(), который добавляет в конец списка указанное значение.

```
In [16]: lst = [] # создание пустого списка
         lst.append(15) # добавление элемента в конец списка
         lst.append(False)
         lst.append('my_stirng')
         lst

Out[16]: [15, False, 'my_stirng']
```

```
In [7]: lst2 = []
        for i in 'семинар':
            lst2.append(i)
        lst2

Out[7]: ['с', 'е', 'м', 'и', 'н', 'а', 'р']
```

Операции над списками:

- конкатенация (+). Операция позволяет объединить два списка в один.

- повторение (*). Повторяет список указанное количество раз.

- 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. Возвращает указанный фрагмент списка. Формат операции: [<Начало>: <Конец>:<Шаг>]. Все параметры в операции среза являются необязательными. Если **не** указан параметр <Начало>, то используется значение 0. Если **не** указан параметр <Конец>, то возвращается фрагмент до конца списка. Если **не** указан параметр <Шаг>, то используется значение 1.

Методы списков: *append*() – добавляет один объект в конец списка; *extend*() – добавляет элементы последовательности в конец списка; *insert*() – добавляет один элемент перед элементом в указанной позиции; *index*() – возвращает индекс элемента, имеющего указанное значение; *count*() – возвращает количество элементов с указанным значением; *pop*() – удаляет элемент с указанным индексом и возвращает его; *remove*() – удаляет первый элемент с указанным значением; инструкция *del* может удалять из списка как единичные элементы, так и элементы, получаемые при помощи среза.

Метод *sort*() сортирует сам список и не возвращает никакого значения. В некоторых случаях необходимо получить отсортированный список, а текущий список оставить без изменений. Дпя этого следует воспользоваться функцией *sorted*(). Функция имеет формат: *sorted*(<Последовательность>[,<*key*>=None, <*reverse*>=*False*]).

Пример 2.1.

Если в заданный список входит слово "*repeat*", а последний элемент - число, то заменить список на столько же копий всех элементов, не считая последних двух (последние два просто добавить). Например, список [*'input'*, *'string'*, *'repeat'*, 3] должен быть заменен на [*'input'*, *'string'*, *'input'*, *'string'*, *'input'*, *'string'*, *'repeat'*, 3].

```
In [1]: lst = ['input', 'string','repeat' , 3]
        if ('repeat' in lst) and (type(lst[len(lst)-1]) == int):
            lst1 = lst[:-2]*lst[-1] + lst[-2:]
        lst1

Out[1]: ['input', 'string', 'input', 'string', 'input', 'string', 'repeat', 3]
```

Пример 2.2.

Выполнить циклический сдвиг слов в произвольной строке (слова разделены пробелами) на заднное пользователем число слов. Решить с помощью списков слов. Пример "один два три четыре пять шесть семь", 3 -> "четыре пять шесть семь один два три".

```
In [11]: lst1 = input().split()
         ind = int(input())
         for i in range(ind):
             lst1.append(lst1[0])
             lst1.remove(lst1[0])
         ' '.join(lst1)
```

```
один, два, три, четыре, пять, шесть, семь
3
```

```
Out[11]: 'четыре, пять, шесть, семь один, два, три,'
```

Пример 2.3.

Из произвольной строки создать список. Вместо каждой буквы "s" (без учета регистра), стоящей не на первом и не на последнем месте, вставить в список строку из двойного предыдущего символа и одного следующего. Например, из строки "*test_Stringss*" должен получиться список ['t', 'e', 'eet', 't', '_', '__t', 't', 'r', 'i', 'n', 'g', 'ggs', 's'].

```
In [10]: s=input('произвольная строка= ')
         lst=list(s)
         for ind,symb in enumerate(s):
             if (symb=='s' or symb=='S') and ind!=0 and ind!=len(lst)-1:
                 lst[ind]=lst[ind-1]*2+lst[ind+1]
                 print(lst)
         print(lst)
```

```
произвольная строка= test_StringsS
['t', 'e', 'eet', 't', '_', 'S', 't', 'r', 'i', 'n', 'g', 's', 'S']
['t', 'e', 'eet', 't', '_', '__t', 't', 'r', 'i', 'n', 'g', 's', 'S']
['t', 'e', 'eet', 't', '_', '__t', 't', 'r', 'i', 'n', 'g', 'ggS', 'S']
['t', 'e', 'eet', 't', '_', '__t', 't', 'r', 'i', 'n', 'g', 'ggS', 'S']
```

Пример 2.4.

Создать список из двух произвольных строк *s*1 и *s*2, их длин и результата проверки соблюдения лексикографического порядка (идет ли первая строка раньше второй). В зависимости от параметра *output* путем обращения к элементам списка выводить либо длины строк (*output='lengths'*, в формате "Длины строк: 4 и 5"), либо описание порядка (*output='order'*, в формате "Строка 'первая' идет ПОСЛЕ строки 'вторая'").

In [2]:
```
s1 = input('первая строка= ')
s2 = input('вторая строка= ')
lst = [s1, s2, len(s1), len(s2), s1<s2]
out = input('output= ')
if out == 'lengths':
    print('Длины строк: ',len(s1),' ',len(s2))
elif out=='order':
    if lst[4]:
        print('Строка '+s1+' идет ДО строки ', s2)
    else:
        print('Строка '+s1+' идет ПОСЛЕ строки ', s2)
else:
    print('Неизвестная команда')
```

```
первая строка= компьютерный
вторая строка= практикум
output= order
Строка компьютерный идет ДО строки  практикум
```

Задачи.

1. Сформировать и вывести на экран список из 5 случайных целых чисел от -10 до 10. Для созданного списка:

- вычислить среднее арифметическое значение элементов списка;
- переставить элементы списка в обратном порядке;
- создать новый список из отрицательных элементов исходного списка.

В исходном списке удалить отрицательные элементы. Порядок элементов должен быть сохранен.

2. Имеются 2 списка целых чисел, упорядоченные по возрастанию. Получите новый список, содержащий все элементы исходных списков, в котором элементы также упорядочены в порядке возрастания, не используя сортировку.

3. Треугольник задан в виде списка координат вершин (вершина представляется в виде списка из двух чисел). Проверьте, действительно ли заданные точки определяют треугольник, вычислите длины сторон, площадь и периметр треугольника.

4. Создайте 2 списка из 5 случайных целых чисел от 1 до 10.

- определите, сколько различных чисел содержат списки;
- определите, сколько различных чисел содержится одновременно как в первом списке, так и во втором,
- выведите все числа, которые входят как в первый, так и во второй список в порядке возрастания,
- удалите из первого списка числа, входящие во второй список

5. В произвольной строке, вводимой пользователем, найдите все слова, начинающиеся с гласных букв с использованием регулярного выражения. Представьте ответ в виде списка,

состоящего из перевернутых (записанных от конца к началу) найденных слов. Словом в строке считать подстроку, ограниченную пробелами.

6. Запросить у пользователя 2 целых числа и сохранить их в переменных *max_val*, *repeat*. Создать список из целых чисел со значениями от 1 до *max_val* включительно повторяющийся *repeat* раз.

7. Организовать заполнение списка заданной длины пользователем через input(). Если на вход подается число (целое или через точку), то записывать в список целое число с округлением вверх. Если "True" или "False", то как boolean. Иначе - как строку.

8. Заданы две заранее неизвестные строки в которых слова разделены пробелами. Из двух строк составить одну, в котрой слова из первой и второй строки будут чередоваться при этом порядок слов будет сохранен. Для преобразований использовать списки слов. Пример: 'один два три', 'альфа бетта гамма' -> 'один альфа два бетта три гамма'

9. Пользователь вводит произвольную строку в виде нескольких слов. Составить список, в котором слова находятся в лексикографическом порядке по возрастанию. Составить строку, в которой все слова списка, сформированного ранее, соединены знаком «&». Словом в строке считать подстроку, ограниченную пробелами.

10. Имеется предложение, слова в котором, разделяются пробелами. С помощью списков получить слово, состоящее из первых букв слов в предложении.

11. Дан список A3, состоящий из 2N элементов. Разбейте его на списки B и C по N элементов каждый так, чтобы каждый элемент B не превосходил каждого элемента C.

12. В списке L3 состоящем из 2n + 1 различных элементов. Найдите средний элемент списка. Под средним элементом понимают такой, для которого в списке n элементов больше его и n элементов меньше.

13. Создать в цикле список значений x от -π до +π (с небольшим произвольным шагом). Для заданного списка рассчитать значения функций $y1 = 2sinx$ и $y2 = cos2x$.

2.2. Многомерные списки

Создание вложенных списков:

1 способ:

```
In [2]: lst = [[1, 2, 3], [6, 5, 4], [0, 10, 20]]
        lst

Out[2]: [[1, 2, 3], [6, 5, 4], [0, 10, 20]]
```

2 способ:

```
In [3]: lst = [[1, 2, 3],
               [6, 5, 4],
               [0, 10, 20]]
        lst
Out[3]: [[1, 2, 3], [6, 5, 4], [0, 10, 20]]
```

Обращение к элементам вложенного списка:

```
In [4]: lst[1]
Out[4]: [6, 5, 4]

In [5]: lst[1][0]
Out[5]: 6
```

Длина и тип элементов вложенных списков не обязательно должны быть одинаковыми:

lst = [[1, 0, 0], *True*, [0, 1, 0, 5], ['*Ivanov*', '*Ivan*']]

Создание вложенных списков из случайных элементов:

```
In [22]: S=[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0]]
         from random import*
         n=5
         m=6
         for i in range(n):
             for j in range(m):
                 S[i][j]=randint(-10,10)
         print(S)
         [[-1, 4, -3, -4, -3, -6], [5, 8, -9, -2, -3, -10], [1, 3, -9, -1, -7, -5], [-1,
         -6, 1, -1, -7, 3], [-2, -6, 4, -4, 4, 10]]

In [23]: #перестановка 2 и 3 строки
         S[1],S[2]=S[2],S[1]
         S
Out[23]: [[-1, 4, -3, -4, -3, -6],
          [1, 3, -9, -1, -7, -5],
          [5, 8, -9, -2, -3, -10],
          [-1, -6, 1, -1, -7, 3],
          [-2, -6, 4, -4, 4, 10]]

In [24]: #перестановка 3 и 4 столбца
         for i in range(len(S)):
             S[i][2],S[i][3]=S[i][3],S[i][2]
         S
Out[24]: [[-1, 4, -4, -3, -3, -6],
          [1, 3, -1, -9, -7, -5],
          [5, 8, -2, -9, -3, -10],
          [-1, -6, -1, 1, -7, 3],
          [-2, -6, -4, 4, 4, 10]]
```

Пример 2.5.

Создать копию матрицы, записанной в виде вложенных списков, так что при изменении элементов исходной матрицы, скопированная матрица меняться не будет.

In [6]:
```
import copy
lst=[[0,1,2],[0,1,3],[2,3,4]]
print(lst,' -исходная')
lst2=copy.deepcopy(lst)
print(lst2,' -копия')
lst[0][0]=1000
print(lst,' -исходная',lst2,' -копия')
```

```
[[0, 1, 2], [0, 1, 3], [2, 3, 4]]  -исходная
[[0, 1, 2], [0, 1, 3], [2, 3, 4]]  -копия
[[1000, 1, 2], [0, 1, 3], [2, 3, 4]]  -исходная [[0, 1, 2], [0, 1, 3], [2, 3,
4]]  -копия
```

Пример 2.6.

Имеется переменная *listlist*, содержащая список списков. Заранее неизвестно, сколько в переменной списков и какой они длины (их длины в общем случае различны). Из этих списков составить список, содержащий поочередно элементы каждого из вложенных списков, при этом, порядок этих элементов сохраняется. Процедура создания списка останавливается, когда хотя бы в одном из исходных вложенных списков будут использованные все элементы. Кроме составленного списка получить список из всех элементов, не вошедших в составленный список. Пример: *listlist* =[[1,2,3],['*a*', '*b*'], [30, 40, 50, 60]] -> [1, 'a', 30, 2, 'b', 40], [3, 50, 60]

In [10]:
```
listlist=[[1,2,3],['a', 'b'], [30, 40, 50, 60]]
lst=[]
lst2=[]
for i in range(0,len(listlist)):
    lst.append(len(listlist[i]))
k = min(lst)
for i in range(k):
    for j in range(0,len(listlist)):
        lst2.append(listlist[j][i])
        listlist[j][i]= None
lst=[]
for i in range(len(listlist)):
    for j in range(len(listlist[i])):
        if listlist[i][j]!=None:
            lst.append(listlist[i][j])
print(lst2,lst)
```

```
[1, 'a', 30, 2, 'b', 40] [3, 50, 60]
```

Задачи.

1. Создать матрицу из *n* строк и *m* столбцов (*n*=5, *m*=6). Заполнить матрицу случайными целыми числами от -10 до 10. Вывести матрицу на экран. Для созданной матрицы:

- вычислить количество строк, содержащих нули;
- проверить, есть ли в исходной матрице одинаковые строки;
- вычислить транспонированную матрицу;
- удалить строку и столбец с заданными номерами. Номера строки и столбца вводятся;
- вставить строку и столбец с заданными номерами. Номера вводятся. Строку и столбец заполнить нулями.

2. Результат сессии, состоящей из 3 экзаменов (История, Алгебра, Информатика), для студента задается в виде списка, содержащего фамилию студента и 3 оценки по

пятибалльной системе (0-неявка, 2-не- 39 удовл., 3-удовл., 4-хорошо, 5-отлично). Результаты группы сохраняются в виде списка списков указанного вида. Для группы выведите на экран:

- таблицу с результатами экзаменов;
- фамилии студентов, имеющих задолженности, и названия несданных ими предметов;
- средний балл по каждой дисциплин;
- количество неявок, неудовлетворительных, удовлетворительных, хороших и отличных оценок по дисциплине Информатика.

3. Задан список с вложенными списками (элементы - числа). Для всех списков с длиной больше 3: оставить только три первых элемента, прибавив к третьему все удаленные элементы. Например, из [[1,2], [3,4,4,3,1], [4,1,4,5]] получить [[1, 2], [3, 4, 8], [4, 1, 9]].

4. Реализовать умножение двух матриц, записанных как список списков.

5. Используя распаковку из списка списков сделать список, содержащий последовательность первых и последних значений вложенных списков и сохраняющий порядок их следования. Пример: [[1, 2, 3], [4, 5, 6, 7], [9, 2]] -> [1, 3, 4, 7, 9, 2]

6. Результат сессии, состоящей из 3 экзаменов (История, Математика, Информатика), для студента задается в виде списка, содержащего фамилию студента и 3 оценки по пятибалльной системе (0-неявка, 2-неудовл., 3-удовл., 4-хорошо, 5-отлично). Результаты группы сохраняются в виде списка списков указанного вида. Для группы выведите на экран: таблицу с результатами экзаменов; фамилии студентов, имеющих задолженности, и названия несданных ими предметов; количество неявок, неудовлетворительных, удовлетворительных, хороших и отличных оценок по дисциплине Информатика.

7. Создайте матрицу размера $n \times m$ элементами которой являются дни недели, случайно выбранные из списка

2.3. Кортежи. Создание кортежа.

Кортежи, так же, как и списки, являются упорядоченными последовательностями элементов. Кортежи во многом аналогичны спискам, но имеют одно очень важное отличие - изменить кортеж нельзя. Можно сказать, что кортеж - это список, доступный "только для чтения".

In [2]:
```
k=()   #пустой кортеж
kor=tuple()  #пустой кортеж
k1=(1,2,5,88)   #кортеж, состоящий из чисел
k2=('a','bb','ccc')  #кортеж, состоящий из строк
k3 = (2, True, 'my string', '7')   #кортеж из элементов различных типов
```

Когда справа от оператора присваивания указывается последовательность (в данном случае - это кортежи), а слева указан кортеж, мы говорим, что последовательность справа распаковывается.

a, *b*, *c* = (10, 11, 12)

print(*f* 'a = {*a*}, *b* = {*b*}, *c* = {*c*}')

Задачи.

1. Создание кортежа из слова 'Hello'
2. Создание кортежа из списка.
3. Взятие среза в кортеже.
4. Конкатенация трех кортежей.

2.4. Словари. Создание словаря. Операции над словарями. Методы для работы со словарями

Словарь – набор объектов, каждый из которых является парой вида ключ : значение.

Способы создания словаря:

- явно указав все элементы словаря в тексте (элементы словаря перечисляются в фигурных скобках через запятую)

d = {*'a'*: 1, *'b'*: 2}

*d*1={} # пустой словарь

- поэлементное заполнение словаря.

```
In [1]: d1={}   #пустой словарь
        d1['a'] = 1 # добавление элемента в словарь
        d1['b'] = 2
        d1

Out[1]: {'a': 1, 'b': 2}
```

```
In [3]: d1={}   #пустой словарь
        d1[1] = 'БИ20-1' # добавление элемента в словарь
        d1[2] = 'ИБ20-3'
        d1

Out[3]: {1: 'БИ20-1', 2: 'ИБ20-3'}
```

- с помощью функции *dict*()

dict() # пустой словарь

dict(*a*=1, *b*=2)

Если данных в виде пар нет, то их можно создать, используя функцию *zip*(). Функция *zip*() возвращает последовательность, которую можно преобразовать в список или словарь:

```
In [4]: k = ['a', 'b', 'c']
        v = [1, 2, 3]
        lst = list(zip(k, v)) #получаем список с помощью zip()
        print(lst)
        d=dict(zip(k, v)) #получаем словарь с помощью zip()
        print(d)

        [('a', 1), ('b', 2), ('c', 3)]
        {'a': 1, 'b': 2, 'c': 3}
```

list(*zip*(*'hello'*, [1, 2])) # *zip*() заканчивает работу как только кончается один из итерируемых объектов.

Операции над словарями:

- доступ по ключу ([]).

d = {1: *'int'*, *'a'*: *'string'*, (1, 2): *'tup1e'*}

d[*'a'*] # *'string'*

- проверка существования ключа можно с помощью оператора *in*.

'a' in *d* #*True*

Так как словари относятся к изменяемым типам данных, мы можем изменить элемент по ключу. Если элемент с указанным ключом отсутствует в словаре, то он будет добавлен в словарь.

```
In [5]: d={'a': 1, 'b': 2, 'c': 3}
        d['a'] = 11 # изменение значения по ключу
        d['p']=10 # добавление значения
        d

Out[5]: {'a': 11, 'b': 2, 'c': 3, 'p': 10}
```

- *del* – Удаляет из словаря элемент с заданным ключом. Если элемент с таким ключом отсутствует, то возникает ошибка (возбуждается исключение *KeyError*).

d = {1: *'int'*, *'a'*: *'string'*, (1, 2): *'tup1e'*}

del d[*'a'*] # *d* = {1: *'int'*, (1, 2): *'tup1e'*}

Основные методы словарей:

- *get*() – возвращает значение элемента словаря. Ключ указывается в первом параметре метода. Если в словаре нет такого ключа, то возвращается значение второго (необязательного) параметра. По умолчанию значение второго параметра равно *None*.

- *clear*() – удаляет все элементы словаря.

- *pop*() – удаляет элемент с ключом, указанном в первом параметре, и возвращает значение, соответствующее этому ключу.

- *update*() – добавляет элементы в словарь.

- *keys*() – возвращает объект, содержащий все ключи словаря.

- *values*() – возвращает объект, содержащий все значения словаря.

- *items*() – возвращает объект, содержащий все ключи и значения в виде кортежей.

Перебор элементов словаря:

Перебор всех элементов словаря выполняется с помощью цикла *for*.

Если в заголовке цикла указать просто имя словаря, то переменная цикла на каждой итерации будет равна ключу очередного элемента словаря.

```
In [13]: d={'a': 11, 'b': 2, 'c': 3, 'p': 10}
         # обход ключей словаря в цикле for:
         for k in d:
             print((k,d[k]), end=' ')
```

```
('a', 11) ('b', 2) ('c', 3) ('p', 10)
```

Если в заголовке цикла использовать метод *keys*(), то переменная цикла на каждой итерации будет равна ключу очередного элемента словаря.

```
In [14]: d={'a': 11, 'b': 2, 'c': 3, 'p': 10}
         # обход ключей словаря в цикле for:
         for k in d.keys():
             print((k,d[k]), end=' ')
```

```
('a', 11) ('b', 2) ('c', 3) ('p', 10)
```

Если использовать в заголовке цикла метод *values*(), то переменная цикла на каждой итерации будет равна значению очередного элемента словаря.

```
In [16]: d={'a': 11, 'b': 2, 'c': 3, 'p': 10}
         # обход значений словаря в цикле for:
         for k in d.values():
             print(k, end=' ')
```

```
11 2 3 10
```

Если использовать в заголовке цикла метод *items*(), то переменные цикла (*k* и *v*) на каждой итерации будут равны ключи и соответствующему значению очередного элемента словаря.

```
In [21]: d={'a': 11, 'b': 2, 'c': 3, 'p': 10}
         # получение пары ключ:значение словаря в цикле for:
         for k,v in d.items():
             print((k,v),end=' ')
```

```
('a', 11) ('b', 2) ('c', 3) ('p', 10)
```

Пример 2.7.

Для строки *stroka* создать словарь, где для всех символов, встречающихся в строке, хранится число: сколько раз символ встретился в строке *stroka*.

```
In [24]: stroka = """Сборник задач по дисциплине «Компьютерный практикум.
         Цель методических указаний – предоставление необходимого
         методического обеспечения по выполнению практических заданий
         по дисциплине «Компьютерный практикум».  """
         d={}
         for i in stroka:
             d[i]=d.get(i,0)+1
         print(d,end=' ')
```

```
{'С': 1, 'б': 3, 'о': 17, 'р': 7, 'н': 12, 'и': 20, 'к': 10, ' ': 23, 'з': 3,
'а': 10, 'д': 8, 'ч': 5, 'п': 13, 'с': 7, 'ц': 2, 'л': 5, 'е': 18, '«': 2, 'К':
2, 'м': 7, 'ь': 3, 'ю': 3, 'т': 8, 'ы': 3, 'й': 4, 'у': 3, '.': 2, '\n': 3,
'Ц': 1, 'х': 3, '–': 1, 'в': 2, 'г': 2, 'я': 1, '»': 1}
```

Пример 2.8.

Написать код, который создает новый словарь с именем *dic*4, содержащий все пары ключ-значение из словарей *dic*1, *dic*2, *dic*3.

```
In [1]: dic1={1:10, 2:20}
        dic2={3:30, 4:40}
        dic3={5:50,6:60}
        dic4={}
        dic4.update(dic1)
        dic4.update(dic2)
        dic4.update(dic3)

        dic4
```

```
Out[1]: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

Пример 2.9.

Просуммировать все значения из словаря dic4.

```
In [25]: dic4={1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
         s=0
         for i in dic4.values():
             s+=i
         s
```

```
Out[25]: 210
```

Задачи.

1. Дана строка, состоящая из слов, разделенных пробелами. Выведите все слова, встречающиеся в этой строке, на экран. Рядом выведите, сколько раз встречается слово в исходной строке. Слова должны быть отсортированы по убыванию их количества появления в тексте.

2. Имеется строка с названиями товаров вида «яблоки, груши, яблоки, киви, сливы, киви». Товары перечислены через запятую, товары могут повторяться. Выведите название товара, который встречается в этой строке чаще всего. Если таких товаров несколько, то названия всех товаров.

3. Число от 2 до 55, которое может содержать только цифры 2, 3, 4, 5 записано прописью, например, «тридцать три». Вычислите квадратный корень из этого числа.

4. Имеется список названий месяцев: [‘января’, ‘февраля’, ‘марта’, ‘апреля’, ’мая’, ‘июня’, ‘июля’, ‘августа’, ‘сентября’, ‘октября’, ‘ноября’, ‘декабря’]. Создайте по этому списку словарь, в котором название месяца будет ключом, а номер месяца (от 1 до 12) – значением. Используя полученный словарь преобразуйте строку с датой вида «1 января 2021» в строку «1.01.2021»

5. На основе переданной строки (не содержащей повторяющихся символов) создать словарь, в котором каждому символу строки будет соответствовать номер символа в строке. Пример: строка *'abcdef'*

6. Определить сколько элементов заданного списка содержится в словаре. Пример: определить сколько элементов списка *l*1 содержится в словаре *d*1. *l*1 = *list*(*'abcd'*)

*d*1 = {*'c'*: 5, *'d'*: 6, *'e'*: 7, *'f'*: 8, *'g'*: 9, *'h'*: 10}

7. Из введенного пользователем кода Морзе, составьте слово, используя латинский алфавит. Короткую паузу (между элементами в букве) в азбуке Морзе обозначить через пробел, длинную (между буквами в слове) – через тройной пробел.

8. Задана строка, в которой через запятую перечислены слова. Создать словарь, в котором ключами будут слова из строки, а значениями - текст "номер {номер-слова-в-строке} в строке". Например, 'ten,one,five,two,three,four' преобразовать в {'three': 'номер 5 в строке', 'one': 'номер 2 в строке', 'ten': 'номер 1 в строке', 'two': 'номер 4 в строке', 'five': 'номер 3 в строке', 'four': 'номер 6 в строке'}.

9. phones_list = [{'name':'Ivan', 'city':'Moscow', 'phones':['232-19-55', '+7 (916) 230-00-75']},
{'name':'Anna', 'city':'Samara', 'phones':['200-11-15']},
{'name':'Anna', 'city':'Vologda', 'phones':['+7 (931) 711-00-75']},
{'name':'Nikolay', 'city':'Moscow', 'phones':['+7 (916) 778-71-05', '331-66-11', '783-33-85']},
{'name':'Ivan', 'city':'Moscow', 'phones':['+7 (916) 205-41-05', '232-19-55']},
{'name':'Anna', 'city':'Samara', 'phones':['+7 (916) 105-13-56']}
]

9.1 Переменная phones_list хранит структуру данных со списком контактов, содержащим номера телефонов. Из списка необходимо удалить дубликаты (записи о людях с совпадающим именем и городом). При удалении дубликатов необходимо телефоны из удаляемой записи добавить в сохраняемую запись (если их там еще нет). То есть номера телефонов, которые уже есть в сохраняемой записи, переносить из дубликата не надо (чтобы избежать дублирования номеров телефонов).

9.2 Переменная phones_list хранит структуру данных со списком контактов, содержащим номера телефонов. Преобразовать этот список в словарь, ключами в котором являются

города, а значениями - словари, в которых номерам телефонов людей, проживающих в данном городе, сопоставлены имена этих людей.
Пример результата преобразования для данных из переменной phones_list:

{'Moscow': {'+7 (916) 205-41-05': 'Ivan', '+7 (916) 230-00-75': 'Ivan', '+7 (916) 778-71-05': 'Nikolay', '232-19-55': 'Ivan', '331-66-11': 'Nikolay', '783-33-85': 'Nikolay'},
'Samara': {'+7 (916) 105-13-56': 'Anna', '200-11-15': 'Anna'},
'Vologda': {'+7 (931) 711-00-75': 'Anna'}}

{'Moscow': {'+7 (916) 205-41-05': 'Ivan',
'+7 (916) 230-00-75': 'Ivan',
'+7 (916) 778-71-05': 'Nikolay',
'232-19-55': 'Ivan',
'331-66-11': 'Nikolay',
'783-33-85': 'Nikolay'},
'Samara': {'+7 (916) 105-13-56': 'Anna', '200-11-15': 'Anna'},
'Vologda': {'+7 (931) 711-00-75': 'Anna'}}

10. Перемножить все значения из словаря dic4. dic4={1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

11. Результат сессии, состоящей из 3 экзаменов (История, Математика, Информатика), для студента задается в виде списка, содержащего фамилию студента и 3 оценки по пятибалльной системе (0-неявка, 2-неудовл., 3-удовл., 4-хорошо, 5-отлично). Результаты группы сохраняются в виде словаря. Для группы выведите на экран: таблицу с результатами экзаменов; средний балл по каждой дисциплин; средний балл для каждого студента.

2.5. Множества. Создание множества. Операции над множествами. Методы для работы с множествами

Множество - это неупорядоченная коллекция уникальных элементов, с которой можно сравнивать другие элементы, чтобы определить, принадлежат ли они этому множеству. Множество может содержать только элементы неизменяемых типов, например, числа, строки, кортежи.

Способы создания множества:

- явно указав все элементы множества в тексте (элементы множества перечисляются в фигурных скобках через запятую)

s = {*'a'*, 1, *'b'*, 2}

- с помощью функции *set*()

set () # пустое множество

set (‘*abcdef*’) # преобразуем строку в множество {*'a'*, *'b'*, *'c'*, *'d'*, *'e'*, *'f'*}

set ([1, 2, 3, 1, 2, 3]) # преобразуем список в множество {1, 2, 3}

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

Операции над множествами:

- объединение множеств (|).

```
In [1]: A = {1,2,3,4,5}
        B = {9,8,7,6,4}
        C = A|B
        print(C)
        C1=A.union(B) #аналог A|B
        print(C1)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

- пересечение множеств (&).

```
In [2]: A = {1,2,3,4,5}
        B = {9,8,7,6,4}
        C = A&B
        print(C)
        C1=A.intersection(B) #аналог A&B
        print(C1)
```

```
{4}
{4}
```

- разность множеств (-).

```
In [3]: A = {1,2,3,4,5}
        B = {9,8,7,6,4}
        C = A-B
        print(C)
        C1=A.difference(B) #аналог A-B
        print(C1)
```

```
{1, 2, 3, 5}
{1, 2, 3, 5}
```

- симметричная разность множеств (^).

```
In [4]: A = {1,2,3,4,5}
        B = {9,8,7,6,4}
        C = A^B
        print(C)
        C1=A.symmetric_difference(B) #аналог A^B
        print(C1)
```

```
{1, 2, 3, 5, 6, 7, 8, 9}
{1, 2, 3, 5, 6, 7, 8, 9}
```

- *in* - проверка наличия элемента во множестве.

copy() – создает копию множества.

add(<Элемент>) – добавляет <Элемент> во множество:

remove(<Элемент>) – удаляет <Элемент> из множества. Если элемент не найден, то возбуждается исключение *KeyError*.

discard(<Элемент>) – удаляет <Элемент> 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

Обход элементов множества в цикле *for*:

```
s2 = set ([1, 2, 3, 4, 5])
for v in s2:
    print(v, end=' ')
```

Задачи.

1.Создайте два множества *A* и *B*, добавив в *A* 10 случайных целых чисел от 1 до 20, а в *B* – 5 таких же чисел. Для созданных множеств:

- найдите их объединение, пересечение, разность, множество элементов, которые не входят в пересечение;
- проверьте, является ли *A* подмножеством *B* и наоборот.

2.Создайте 2 списка из 5 случайных целых чисел от 1 до 10.

- определите, сколько различных чисел содержат списки;
- определите, сколько различных чисел содержится одновременно как в первом списке, так и во втором,
- выведите все числа, которые входят как в первый, так и во второй список в порядке возрастания,
- удалите из первого списка числа, входящие во второй список.

3. Для заданного натурального $n \geq 2$ найти все простые числа, которые меньше или равны n, используя алгоритм, который называют «решето Эратосфена»:

- пусть *R* – строимое множество простых чисел и *D* – множество, называемое решетом, в начале работы *R* – пустое множество, *D* – множество всех целых чисел от 2 до n;
- на каждом шаге алгоритма наименьший элемент *D* помещается в *R*, а из *D* удаляются все числа, кратные этому элементу;
- алгоритм заканчивает работу при пустом *D*.

4. Участники олимпиады решали 3 задачи. Известны фамилии тех, кто решил первую, вторую и третью задачи (для каждой задачи отдельный список). Найдите и выведите на экран фамилии тех, кто

- решил хотя бы одну задачу (любую);
- решил все задачи;
- решил ровно1 задачу (любую);
- решил ровно 2 задачи (любые);
- решил не больше 2 задач (любых).

5. Каждый участник международной конференции указал, какими языками он владеет (для хранения этой информации используйте словарь). Предполагается, что конференция проводится на русском языке, и выполняется синхронный перевод на английский. Определите:

- есть ли язык, на котором разговаривают все участники;
- фамилии участников, которые не владеют русским языком;
- на какие языки еще нужен перевод (язык учитывается, если участник не знает русского и английского), и сколько человек знают этот язык.

6. Из множества целых чисел от 1 до N выделить множество $N2$ числа, кратные 2, множество $N3$ – кратные 3, множество $N6$ – кратные 6 (т.е. кратные и 2 и 3), множество $N23$ – кратные либо 2, либо 3.

Тесты по разделу 2.

1. В Python к изменяемым типам данных относятся

1) Кортежи
2) Строки
3) Списки
4) Словари
5) Массивы

2. Какие структуры данных можно изменять?

1) list, tuple
2) tuple, dict
3) dict, list
4) str, tuple
5) set, list

3. Как создать список?

1) L1=list(1,2,3)

2) L1=[1,2,3]
3) L1=list[1,2,3]
4) Все варианты верны
5) L1=(1,2,3)
6) L1={1, 2, 3}

4. Что выведет этот код:

```
lake=['p', 51, False, '22']
lake.reverse()
lake.reverse()
print(lake[-2])
```

1) False
2) -2
3) [51]
4) 51
5) '22'

5. Что выведет этот код:

```
s=[10, 20, 30]
s.append(60)
s.insert(3,40)
print(s)
```

1) [10, 20, 30, 40]
2) [60, 10, 20, 30, 40]
3) [10, 20, 30, 60, 40]
4) [10, 20, 30, 40, 60]
5) [10, 20, 30, 40, 40, 40, 60]
6) нет правильных ответов

6. Как получить значение по ключу 'marks' словаря student={'n':'Elena','c':10,'marks':67}:

1) student('marks')
2) student.get(3)
3) student.get(2)
4) student.items('marks')

5) все не верно

6) все верно

7. *Какой способ создания пустого словаря является правильным?*

1) dict({})
2) dict('')
3) dict(())
4) все правильные
5) все не правильные
6) dict([])

8. *Результат выполнения команд*

```
In [1]: B={2,3,4,5}

In [2]: {2,3,4,5}<B
```

равен _____.

9. *Результат выполнения команды*

```
In [4]: B={2,3,4,5}

In [5]: {2,3,4,5}<=B
```

равен _____.

10. *Поставьте в соответствие выражение и значение интерпретатора для списка sp = [2, 8, 4, 6, 1, 4, 7, 4, 6]?*

1) min(sp)	a) 6
2) max(a1)	b) 8
3) sp.count(4)	c) 2
4) len(sp)	d) 9
5) sp.pop()	e) 1
6) len(sp)	f) 3

11. *Поставьте в соответствие адрес и значение интерпретатора для списка sp = [[2, 4, [4, [5]], 8], [1, [[4, 7], 5, [7, 9, 4]]]]*

1) sp[0][2][1]	a) 7
2) sp[0][3]	b) 4

3) sp[1][1][0][1]	c) 5
4) sp[1][1][2][2]	d) 8

12. *Поставьте в соответствие адрес и значение интерпретатора для кортежа sp = ((2, 4, (4, (5)), 8), (1, ((4, 7), 5, (7, 9, 4))))*

1) sp[0][2][1]	a) 4
2) sp[0][3]	b) 7
3) sp[1][1][0][1]	c) 5
4) sp[1][1][2][2]	d) 8

13. Выберете верное утверждение:

1) Можно удалить элемент, но нельзя добавить новый
2) Можно удалить элемент из кортежа
3) Можно добавить элемент в кортеж
4) Все не верно

14. Что выведет этот код:

```
sample = (4, 6, 9)
sample.append(7)
print(sample)
```

1) (4, 6, 9)
2) (4, 6, 9, 7)
3) [4, 6, 9, 7]
4) Ошибка

15. С помощью какой функции можно создать пустой кортеж:

1) taple()
2) tupl()
3) tuple()
4) typle()
5) Такой функции в python нет

16. Как получить последний элемент этого кортежа k = ("Python", 51, False, "22"):

1) k[3]

2) k[lake.index("22")]

3) k[-1]

4) Все варианты подходят

17. Как получить значение по ключу " key" словаря login = {"user": "student", "room": 92, " key": 24}

1) login ("key")

2) login.get(2)

3) login.get(3)

4) Все не верно

18. Как создать пустой словарь:

1) d = dict{}

2) d = {:}

3) d = {}

4) d = dict()

5) Все способы неверны

19. Какие типы данных могут быть ключами словаря:

1) str, int, tuple

2) int, list, str

3) float, list, tuple

4) int, float, str

5) Любой тип данных

6) Все способы неверны

20. Что выведет следующий код

```
p = {"name": " Jack", "salary": 90000}
print(p.get("age"))
```

1) "age"

2) 90000

3) None

4) Jack

5) {"name": " Jack", "salary": 90000}

6) Ошибка

21. Как вывести значение "d":

```
sample = {" one":["a","b"], "two":["c","d"]}
```

1) print(sample[1][1])
2) print(sample[2][1])
3) print(sample["two"][2])
4) print(sample[2][2])
5) print(sample["two"][1])
6) print(sample[two][1])

22. Что выведет следующий код:

```
num_set = {1, 2, 3, 4, 5, 6, 3, 7, 2}
len_num_set = len(set(num_set))
print(len_num_set)
```

1) 0
2) 1
3) 7
4) 9
5) 17

23. Что выведет код

```
tests = {'go', 'php', 'js', 'cpp', 'java'}
tests.remove('python')
```

1) {}
2) {'python'}
3) {'go', 'php', 'js', 'cpp', 'java', 'python'}
4) Ошибка

3. Генераторы

Генераторы списков.

Создание списка с помощью цикла:

lst = [10, 20, 30, 40]

lst_new = []

for el in lst:

    lst_new.append(*el* * 2)

lst_new # *lst_new* = [20, 40, 60, 80]

Создание списка при помощи генератора:

lst_gen = [*el* * 2 *for el in lst*]

lst_gen # lst_ gen = [20, 40, 60, 80]

Создание списка с помощью цикла с условием:

lst = [10, 20, 30, 40, 45, 23]

lst_new = []

for el in lst:

    if el % 2 == 0: # число *el* четное (остаток от деления на 2 равен 0)

        lst_new.append(*el* // 2)

lst_new # *lst_new* = [5, 10, 15, 20]

Создание списка при помощи генератора с условием:

lst_gen = [*el* // 2 *for el in lst if el*%2==0]

lst_gen # *lst_ gen* = [5, 10, 15, 20]

Пример 3.1.

Привести вложенный список к «плоскому» виду. Например, *lst* = [[1, 2, 3], [4, 5], [6], [7], [8, 9]] –>[1, 2, 3, 4, 5, 6, 7, 8, 9]

```
In [7]: lst = [[1, 2, 3], [4, 5], [6], [7], [8, 9]]
        lst_new = [i for j in lst for i in j]
        lst_new

Out[7]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Генераторы множеств.

Создание множества при помощи генератора:

set_mn = {10, 20, 30, 40}

set_gen = {*el* * 2 *for el in set_mn*}

set_gen # *set_* gen = {20, 40, 60, 80}

Создание множества при помощи генератора с условием:

set_gen = {*el* // 2 *for el in set_mn if el*%2==0}

set_gen # *set_gen* = {5, 10, 15, 20}

Генераторы словарей.

Создание словаря при помощи генератора:

keys = ['a', 'b'] # Список с ключами

values = [1, 2] # Список со значениями

d= {*k*: *v for* (*k*, *v*) *in zip*(*keys*, *values*)} # {'a': 1, 'b': 2}

Создание словаря при помощи генератора с условием:

{*k*: 2 * *v for k, v in d.items*() *if v* % 2 == 0} # {'b': 4}

Пример 3.2.

На основе переданной строки (не содержащей повторяющихся символов) создать словарь, в котором каждому символу строки будет соответствовать номер символа в строке. Использовать генераторы, решить задачу в одну строку. Пример: строка 'abcdef', словарь {'a': 1, 'b': 2, 'c': 3, 'd': 4}

```
In [15]: s=input()
         dic={v:k for k,v in enumerate(s,1)}
         dic

         abcdef

Out[15]: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
```

Пример 3.3.

Для строки *stroka* создать словарь, где для всех символов, встречающихся в строке, хранится число: сколько раз символ встретился в строке *stroka*.

```
In [16]: stroka = """Сборник задач по дисциплине «Компьютерный практикум.
         Цель методических указаний – предоставление необходимого
         методического обеспечения по выполнению практических заданий
         по дисциплине «Компьютерный практикум».  """
         dic={symb:stroka.count(symb) for symb in stroka}
         print(dic)

         {'С': 1, 'б': 3, 'о': 17, 'р': 7, 'н': 12, 'и': 20, 'к': 10, ' ': 23, 'з': 3,
         'а': 10, 'д': 8, 'ч': 5, 'п': 13, 'с': 7, 'ц': 2, 'л': 5, 'е': 18, '«': 2, 'К':
         2, 'м': 7, 'ь': 3, 'ю': 3, 'т': 8, 'ы': 3, 'й': 4, 'у': 3, '.': 2, '\n': 3,
         'Ц': 1, 'х': 3, '–': 1, 'в': 2, 'г': 2, 'я': 1, '»': 1}
```

Пример 3.4.

В словаре *dic*, полученном в предыдущей задаче, удалить все пары ключ-значение для символов, встречающихся менее 5 раз. Использовать генераторы, решить задачу в одну строку.

```
In [17]: dic1={k:v for k,v in dic.items() if v>5}
         print(dic1)

         {'о': 17, 'р': 7, 'н': 12, 'и': 20, 'к': 10, ' ': 23, 'а': 10, 'д': 8, 'п': 13,
         'с': 7, 'е': 18, 'м': 7, 'т': 8}
```

Задачи.

*l*1 = ['1', '123', '123', '12', '1', '123']

*l*2 = [2, 4, -2, -3, 0 , 11 , 3, -1]

*d*4 = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

d 5 = {*'a'*: 3, *'b'*: 4, *'c'*: 5, *'d'*: 6, *'e'*: 7, *'f'*: 8, *'g'*: 9}

d 6 = {*'e'*: 20, *'f'*: 21, *'g'*: 22, *'h'*: 23, *'i'*: 24, *'j'*: 25, *'k'*: 26, *'l'*: 27}

1. Используя список *l*1, создать список, в котором каждый элемент списка *l*1 будет заменен значением длины строки соответствующего элемента в списке *l*1. Использовать генераторы, решить задачу в одну строку.
2. Подсчитать количество строк в списке *l*1, длина которых больше 2х. Использовать генераторы, решить задачу в одну строку.
3. Создать список, в котором все числа списка *l*2 умножены на их номер в списке. Использовать генераторы, решить задачу в одну строку.
4. Создать список, в котором все отрицательные числа списка *l*2 исключены, остальные элементы сохранены на своих местах. Использовать генераторы, решить задачу в одну строку.
5. Создать список, в котором все отрицательные числа списка *l*2 заменены на их номер в списке, остальные элементы сохранены на своих местах. Использовать генераторы и тернарный оператор, решить задачу в одну строку.
6. На основе переданной строки (не содержащей повторяющихся символов) создать
7. Задан список. С помощью генераторов создать словарь, в котором ключами будут символы из заданного списка, а значениями – числа из того же списка (порядок следования символов сохранить, а порядок чисел - перевернуть). Словарь должен включать в себя все символы из списка. Если символов больше, чем чисел, то значением ключей, для которых чисел не хватает, устанавливать 0. Если символов меньше, то оставшиеся числа не учитывать. Решить задачу в одну строку. Например: [1,2,*'a'*,3,4,*'b'*,*'c'*,*'d'*,*'e'*] перевести в {*'a'*: 4, *'b'*: 3, *'c'*: 2, *'d'*: 1}
8. Задана строка, в которой через запятую перечислены имена людей (с заглавной буквы) и их текущие занятия (со строчной буквы) в произвольном порядке (например, "Иван ест, поет Оля" и т.д.). С помощью генераторов создать словарь, в котором ключами будут имена, а значениями – занятия. Решить задачу в одну строку. Например, "Маша гуляет, Коля работает, дома Ваня, закупается Женя" представить в виде {'Ваня': 'дома', 'Женя': 'закупается', 'Коля': 'работает', 'Маша': 'гуляет'}.
9. Имеется предложение, слова в котором, разделяются пробелами. С помощью генератора списков получить слово, состоящее из первых букв слов в предложении.

10. С помощью генератора создать в цикле список значений *x* от -π до +π (с небольшим произвольным шагом). Для заданного списка рассчитать значения функций $y1 = 2sinx$ и $y2 = cos2x$.

Тесты по разделу 3.

1. Какое из следующих утверждений НЕ верно?

1) в цикле "*for i in range* (4)" переменная "*i*" принимает значения 0, 1, 2, 3, 4
2) по умолчанию, функция range начинается с 0
3) параметры функции range могут быть отрицательными целыми числами
4) невозможно получить последовательность 6, 12, 18, 24 с помощью range
5) невозможно получить последовательность 24, 18, 12, 6 с помощью range

2. Выберите код, который вернет наибольшее значение?

1) *sum*(*list*(*range*(9,99,99)))
2) *sum*(*list*(*range*(10,1000,-1)))
3) *sum*(*list*(*range*(5)))
4) *sum*(*list*(*range*(0,10,5)))
5) *sum*(*list*(*range*(0,10,-1)))
6) *sum*(*list*(*range*(10,1)))

3. Какие выражения генерируют список:

1) *list*(*lambda x*: *x***2, *range*(10))
2) *list*(*map*(*lambda x*: *x***2, *range*(10)))
3) [*x***2 *for x in range*(10)]
4) [*map*(*lambda x*: *x***2, *range*(10))]

4. Правильные утверждения относительно генераторов в языке Python:

1) Генератор – это объект, который сразу при создании не вычисляет значения всех своих элементов.
2) Генератор – это объект, который при создании вычисляет значения всех своих элементов.
3) Генератор хранит в памяти только последний вычисленный элемент, правило перехода к следующему и условие, при котором выполнение прерывается.
4) Вычисление генератором следующего значения происходит лишь при выполнении метода *next*(); предыдущее значение при этом теряется.

4. Функции. Анонимные функции

Функции объявляются с помощью инструкции *def*:

def Имя_Функции ([*Имена_Параметров*]):

[*''' Строка_Документирования '''*]

Тело_Функции

[*return Значение*]

В квадратных скобках указаны элементы, которые могут отсутствовать.

Функция вызывается с помощью конструкции:

Имя_Функции ([*Значения_Параметров*])

In [4]:
```
def summa(x,y):
    """  Пример функции с параметрами,
    возвращающей сумму двух переменных"""
    return x + y
print(summa(3, 4))
print(summa(y=4, x=3))
s=summa(3, 4)
print(s)
```

Распаковка аргументов и параметров.

Если значения параметров, которые планируется передать в функцию, содержатся в кортеже или списке, то перед объектом следует указать символ *.

In [8]:
```
t1 = (5, 10 ,25)
t2 = [10, 20, 50]
t3 = (5, 10 ,25, 20)
def summa1(x=1, y=1, z=1):
    return z-y-x

print(summa1(*t1))
print(summa1(*t2))
print(summa1(*t3[:3]))
```

```
10
20
10
```

Если значения параметров находятся в словаре, то распаковать параметры можно, указав в списке параметров перед именем словаря две звездочки:

```
In [12]: d1={'x':5,'y':10,'z':25}
         def summa1(x=1, y=1, z=1):
             return z-y-x

         print(summa1(**d1))
         print(summa1(x=5,y=10,z=25))
```

```
10
10
```

Глобальные и локальные переменные

Глобальные переменные - это переменные, объявленные в программе вне функции. В *Python* глобальные переменные видны в любой части модуля, включая функции.

Локальные переменные - это переменные, которым внутри функции присваивается значение. Если имя локальной переменной совпадает с именем глобальной переменной, то все операции внутри функции осуществляются с локальной переменной, а значение глобальной переменной не изменяется. Локальные переменные видны только внутри тела функции.

```
In [16]: def fun1():
             print("fun1 ", a)
         def fun2():
             a=22
             print("fun2 ", a)
         def fun3():
             global a
             a=20
             print("fun3 ", a)
         a=1
         fun1()
         fun2()
         print(a)
         fun3()
         print(a)
```

Переменное число параметров в функции.

Если перед именем параметра в определении функции указать символ «*», то функции можно будет передать произвольное число параметров. Переданные параметры сохраняются в кортеже. Если перед именем параметра в определении функции указать две звездочки, то функции можно будет передать произвольное число именованных аргументов. Переданные параметры сохраняются в словаре.

Пример 4.1.

Написать функцию умножения, принимающую от одного до трех параметров. Представьте примеры использования этой функции с разным количеством параметров.

In [17]:
```
def fun(*a):
    p=1
    for x in a:
        p=p*x
    return p
print(fun(2,3,4))
print(fun(2,3))
print(fun(2))
```

```
24
6
2
```

Пример 4.2.

Напишите функцию, параметрами которой являются слово и любое количество произвольных строк. Функция возвращает список строк, в которых встречается указанное в первом параметре слово. Строки могут состоять из букв и пробелов. Регистр букв не имеет значения. Например, при вызове f(‘Два’, ‘двадцать пять’, ‘сорок два’) функция вернет список [‘сорок два’]. Вызовите функцию для случая, когда строки хранятся в заранее созданном списке.

In [23]:
```
def f(*stroka):
    print(stroka[0])
    k=stroka[0]
    for i in range(1,len(stroka)):
        print(stroka[i])
        a=stroka[i].split(" ")
        print(a)
        for j in range(len(a)):
            if k.lower()==a[j].lower():
                print("ok ",a)
x=f("дВа","двадцать три","сорок Два","три двА")
```

Пример 4.3.

Написать функцию вывода словаря с произвольным количеством параметров.

In [18]:
```
def d_summa(**d):
    for k, v in d.items():
        #  Перебираем словарь с переданными параметрами
        print(f"{k} => {v}", end="; ")
d1={'x':5,'y':10,'z':25}
d_summa(**d1)
```

```
x => 5; y => 10; z => 25;
```

Анонимные функции.

Анонимные функции можно создавать инструкцией

lambda параметры: выражение

Такая инструкция создаст ссылку на функцию, принимающую указанный список параметров и возвращающую результат вычисления выражения. Эту ссылку можно сохранить в переменной или передать в качестве параметра в другую функцию. Вызывается эта функция как обычная функция.

Пример 4.4.

Напишите функцию, которая находит максимум функции $f(x)$ в точках отрезка $[a,b]$ с постоянным шагом h. Параметрами функции являются f, a, b, h. Параметры a, b, h – необязательные, по умолчанию a=0, b=1, h=0.1. Используя эту функцию, найдите максимум функции $(2-x)sin(x/2)$ на отрезке [0, 4].

In [21]:
```
from math import sin
def maximal(F,a=0,b=1,h=0.1):
    M=F(a)
    while a<b:
        if F(a)<M:
            M=F(a)
        a=a+h
    return M
print(maximal(lambda x:(2-x)*sin(x/2),0,b=4))
```

Пример 4.5.

Отсортировать вложенный список по выбранному критерию. В качестве ключа сортировки используйте либо лямбда-функцию, либо обычную функцию.

```
a = [['did', 1, 2, 4],['u', 4, 3, 2], ['p', 5, 5, 5], ['huh?', 1, 1, 1]]
task = input('Отсортировать по имени(1), первой оценке(2) или в обратном порядке по сумме оценок(3)?')
def first_number (input_array):#функция для сортировки по первой оценке
        return input_array[1]
def summed_marks(input_array):#функция для сортировки по сумме оценок
    return input_array[1]+input_array[2]+input_array[3]
b = a #создание копии массива
if task == '1':#проверка на тип сортировки
    b.sort()#сортировка
    for items in b:
        print(*items)    #вывод
elif task == '2':#проверка на тип сортировки
    b.sort(key = first_number)#сортировка
    for items in b:
        print(*items)    #вывод
elif task == '3':#проверка на тип сортировки
    b.sort(reverse=True, key = summed_marks)#сортировка
    for items in b:
        print(*items)    #вывод
else:
    print('Wrong operation!')
```

Задачи на функции с глобальными и локальными переменными и функции с произвольным количеством параметров.

1. Написать задокументированную функцию умножения двух чисел. Для каких типов данная функция будет возвращать практически ценный результат? Написать примеры использования этой функции.

2. Написать функцию умножения, принимающую от одного до трех параметров. Функцию вызвать с приведенными ниже аргументами. Для случая *a*4 выбрать 3 первых и 3 последних значения.

*a*1 = (15, 10 ,5)

*a*2 = (3, 1)

*a*3 = [2, 35, 55]

*a*4 = (5, 10 ,15, 20)

3. Реализовать функцию умножения, принимающую неограниченное количество значений. Написать примеры использования этой функции, в том числе с аргументами из задания 3.

4. Реализовать функции для выполнения четырех арифметических операций, преобразующих целые числа в целое число. Создать словарь с функциями и соответствующими им символами операций. Для двух заранее заданных целых чисел (например, 25 и 4) выполнить выбранную пользователем арифметическую операцию.

5. Написать калькулятор для строковых выражений вида '<число> <операция> <число>', где <число> - целое число, например 113, <арифметическая операция> - одна из операций +,-,*,/(деление нацело),%(остаток от деления),^(возведение в степень). Пример calc('13 - 5') -> 8

6. Реализовать функцию, которая выводит на экран сообщение вида: 'Автомобиль марки: BMW, модели: X5, цвета: белый, 2006 года выпуска, с пробегом: 215 000 км, с номерным знаком: X012AM77, цена: 1 115 000 руб.' В функцию передаются именованные параметры изменяемой части сообщения, любой из параметров может быть не задан пользователем функции и автоматически заменен на разумное сообщение в соответствующей части строки. Получить вывод текстового сообщения для словарей *c*1, *c*2.

7. Написать функцию, которая преобразует целое число (от 0 до 99) в текстовое написание числа на русском языке. Пример: *to_text*(15) -> 'Пятнадцать'

8. Написать функцию, которая преобразует целое число (от 0 до 999) из текстового представления на русском языке в число типа *int*. Пример: *to_int*('тридцать три') -> 33

9. Написать функцию, которая преобразует целое число (от 0 до 999) из текстового представления на русском языке в число типа *int*. И сообщает об ошибках (выводит на экран описание типа ошибки и возвращает число -1). Пример: *to_int*('тридцать три') -> 33 Пример: *to_int*("сто сорок тридцать два сто") -> -1 Вывод: тридцать - некорректное расположение в числе.

10. Написать функцию, имеющую 3 параметра: первые 2 - числа, третий - операция, которая должна быть произведена над ними. Если третий параметр «+», то нужно сложить

числа, если «-» — вычесть, «*» — умножить, «/» — разделить (первое на второе). Функция возвращает результат выполнения операции над числами. Если операция не совпадает с указанными выше, то выводится сообщение "Неизвестная операция", и возвращается значение *None*.

11. Напишите функцию, которая для заданного радиуса r вычисляет площадь круга и длину окружности. Функция возвращает кортеж из 2 значений.

12. Для треугольника со сторонами x, y, z угол α между сторонами x, y можно вычислить следующим образом: $d = \cos\alpha = x^2 + y^2 - z^2 / (2xy)$; $\alpha = \arccos d = \pi/2 - \arcsin d$

13. Напишите функцию, которая имеет произвольное количество именованных параметров. Функция возвращает список всех имен параметров и сумму всех значений. Например, при вызове *f*(*a*=2, *b*=3, *c*=2, *d*=5) функция вернет (['*a*','*b*','*c*','*d*'], 12). Вызовите функцию для случая, когда параметры хранятся в заранее созданном словаре, например, *res* = {'Математика':92, 'Информатика':80, 'Физика':74}.

14. Напишите функцию, которая находит угол α для треугольника со сторонами x, y, z в градусах (воспользуйтесь функциями модуля *math*). Используя эту функцию напишите еще одну функцию, которая по заданным сторонам треугольника находит все его углы (в градусах). Функция возвращает кортеж из 3 чисел, причем первое число – угол, находящийся напротив стороны x, второе – угол напротив y, третье – угол напротив z.

15. Напишите функцию, которая находит наибольший общий делитель двух чисел, используя модифицированный алгоритм Евклида: нужно заменять большее число на остаток от деления большего на меньшее до тех пор, пока этот остаток не станет равен нулю; тогда второе и есть НОД. Функция должна возвращать найденное значение

16. Дана дробь n/m, n и m - натуральные числа. Напишите 2 функции, которые сокращают эту дробь, то есть находят числа p и q такие, что $n/m = p/q$, и дробь p/q — несократимая:

- аргументами функции являются числа *n*, *m*, функция возвращает кортеж (*p*, *q*);
- аргументом функции является список [*n*, *m*], функция не возвращает значения, а изменяет этот список на [*p*, *q*].

17. Написать функцию Smooth(R, n), заменяющую каждый случайный элемент вещественного массива R размера *n* на его среднее арифметическое со своими ближайшими соседями ("сглаживание массива"). Массив R – входной и выходной параметр, *n* – входной параметр. С помощью этой процедуры выполнить пятикратное сглаживание данного массива R размера *n*, выводя на экран результаты каждого сглаживания в виде массива и графика.

18. 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.

19. Написать функцию вычисления y:

$$y = \begin{cases} 3x^2 & x > 4{,}5 \\ e^{-x} & 1 \le x \le 4{,}5, \\ -cos^2(2x) & x < 1 \end{cases} \qquad x \in [-\frac{\pi}{2}; 2\pi]$$

20. и на заданном промежутке с шагом h (значение шага задаётся) построить график переменной y. Использовать модуль matplotlib.

21. Функция func(2, 4), в качестве аргумента принимает другую функцию (не встроенную в Python). В результате работы она выводит следующие данные: название переданной функции, наименование всех принимаемых ею параметров и их типы (позиционные, ключевые, целые вещественные, строковые). Например, для вызова функции с данными параметрами func(subfunc (17, a=9.5)), должно быть выдано: имя функции subfunc, первый параметр позиционный целого типа, второй параметр ключевой вещественного типа.

22. Создать три модуля. В первом расположена функция от трех параметров длин сторон параллелепипеда, она вычисляет объем параллелепипеда. Во втором модуле функция вычисляет площадь параллелепипеда. В третьем модуле функция вычисляет периметр сторон параллелепипеда. В основном файле вызывать функции из каждого модуля. Например, для значений 10, 20, 30 результаты вызова функций будет Объем = 6 000, Периметр = 200, Площадь = 2200.

23. Для заданной математической функции y=f(x) разработать программу построения графика функции на интервале от 4 до 8,9. На графике предусмотреть наличие заголовка и подзаголовка, осей, подписей осей и линий сетки.

$$y = \frac{x^2 - 18x + 77.5}{x - 9}$$

24. Разработать функцию по вычислению площади кольца по известным значениям его внешнего и внутреннего радиусов.

25. Напишите функцию для решения уравнений степени не выше второй (квадратные и линейные):

- если у функции три аргумента, их надо трактовать как a, b и c в уравнении $ax^2 + bx + c = 0$;

- если два — как коэффициенты b и c в уравнении $bx + c = 0$;

- если один — как коэффициент c в уравнении $c = 0$;

- если список коэффициентов пуст или коэффициентов больше трёх, то функция должна вернуть *None*. Функция возвращает список, содержаний все корни уравнения (два, один или ни одного). Если корнем является любое значение x, функция возвращает список, содержащий символ «*» (["*"]).

Задачи на анонимные функции.

1. Разработать лямбда-функцию вычисления площади трапеции по известным двум основаниям и высоте.

2. Разработать лямбда-функцию вычисления полной поверхности прямого кругового цилиндра по известному радиусу его основания и высоте.

3. Разработать лямбда-функции вычисления по известной площади круга его радиуса и длины окружности.

4. Задан список словарей: *d*l = [{'a': 10, 'b': 20, 'c': 1}, {'a': 5, 'b': 10, 'z': 10}, {'a': 3, 'y': 7}]. С помощью анонимной функции отсортировать словари по значению, содержащемуся по ключу *'a'*.

5. Написать калькулятор *calc_op*(*s*, *oper_d*), который принимает параметр *s* - строковые выражения вида '<число> <операция> <число>', где <число> - целое число, например 113, <операция> - операция, которая кодируется одной буковой (например: p,m и т.п.); *oper_d* - словарь операций, в котором каждой из рассматриваемых символов операций сопоставлена функция ее расчёта. Пример *calc_op*('2 *s* 10', {*'s*': *lambda x,y*: *x****y* }) -> 1024

Тесты по разделу 4.

1. *Какой тип данных возвращает функция f(x)?*

```
>>> def f(x):
            print(complex(x**(1/2)-1))


>>> f(16)
(3+0j)
>>>
```

2. *В каком виде будет представлен результат при вызове функции test1() и передаче ей значений нескольких позиционных и ключевых аргументов?*

```
def test1(*arg,**args):
    print(arg)
    print(args)
```

3. *Результат работы программы:*

```
def my_func(a, b):
    x = 5
    print(x)

if __name__ == '__main__':
    x = 10
    my_func(1, 2)
    print(x)
```

равен …

4. *Результат работы программы:*

```
def my_func(a, b):
    global x
    print(x)
    x = 5
    print(x)

if __name__ == '__main__':
    x = 10
    my_func(1, 2)
    print(x)
```

равен …

5. *Что выведет код:*

```
def generate_ints(N):
    for i in range(N):
        yield i
gen = generate_ints(3)
next(gen)
next(gen)
print(next(gen))
```

1) 0
2) 2
3) 3
4) *None*
5) *StopIteration*

6. *Какие выражения генерируют список:*

1) *list*(*lambda x*: *x***2, *range*(10))
2) *list*(*map*(*lambda x*: *x***2, *range*(10)))

3) [*x***2 *for x in range*(10)]

4) [*map*(*lambda x*: *x***2, *range*(10))]

7. В каком виде будет представлен результат при вызове функции test1() и передаче ей значений нескольких позиционных и ключевых аргументов?

```
def test1(*arg,**args):
    print(arg)
    print(args)
```

8. Первая строка определения функции f выглядит так:

def f(a, b, c=None, d="0"): какие из следующих вариантов вызова не приведут к ошибке на этапе присваивания фактических параметров формальным:

1) *f*()

2) *f*(3, 4)

3) *f*(3, 4, 5, 6)

4) *f*(3, 4, *d*=5, *c*=6)

5) *f*(3, 4, *d*=2)

9. Первая строка определения функции f выглядит так:

*def f(a, b, **k):*

какие из следующих вариантов вызова не приведут к ошибке на этапе присваивания фактических параметров формальным:

1) *f*()

2) *f*(3, 4)

3) *f*(3, 4, 5, 6)

4) *f*(3, 4, *d*=5, *c*=6)

5) *f*(3, 4, *d*=2)

10. Что выведет данный код:

```
def fan(*args):
   for argument in args:
      print (argument)
 sp=[1,2,3,4,5]
fan(*sp)
```

1) выведет список всех чисел в квадратных скобках

2) напечатает все числа, на одной строке

3) напечатает все числа, каждое с новой строки

4) ничего не выведет

5) выдаст ошибку

11. Что выведет данный код:

```
def call_fan(fun, arg):
    a = fun(arg*arg)
    return a
def mul_fan(x):
    return x*x
call_fan(mul_fan, 3)
```

1) 3

2) 6

3) 9

4) 27

5) 54

6) 81

12. Что выведет данный код:

```
def f():
    global a
    a = 1
    print(a)
a = 0
f()
print(a)
```

1) 0 и 0

2) 1 и 1

3) 0 и 1

4) 1 и 0

13. Как правильно нужно ввести данные, чтобы программа выполнилась без ошибок.

```
def calc1(s):
    lst=s.split()
```

```
    d={'-':lambda x,y : x-y, '+':lambda x,y : x+y, '*':lambda x,y : x*y,
   '/':lambda x,y : x//y,'^':lambda x,y : x**y, '%':lambda x,y : x%y}
    return d[lst[1]](int(lst[0]),int(lst[2]))
s=input()
calc1(s)
```

1) 2*6

2) 2 * 6

3) 2_*_6

4) 2,6,*

5. Исключения. Пользовательские исключения. Инструкция *assert*.

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.

Для обработки исключений предназначена инструкция *try*.

Формат инструкции:

try:

<Блок, в котором перехватываются исключения>

[*except* [<Исключение1> [*as* <Объект исключения>]]:

<Блок, выполняемый при возникновении исключения>

[...

except [<Исключение*N*>[*as* <Объект исключения>]]:

<Блок, выполняемый при возникновении исключения>]]

[*e1se*:

<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 <Исключение1> указывается класс обрабатываемого исключения.

Пример 5.1.

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

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

```
0 1 2 3 4 Это конец
```

Пример 5.2.

Реализовать умножение неограниченного количества чисел. Если аргумент не является число, то должна вызываться ошибка *ValueError*.

In [4]:
```
p=1
kol=int(input())
for i in range(kol):
    try:
        new=int(input())
        p=p*new
    except ValueError:
        print('Неверный ввод числа')
        break
print(p)
```

Пользовательские исключения.

Инструкция *raise* вызывает указанное исключение.

In [81]:
```
try:
    raise ValueError("Описание исключения")
except ValueError  as  msg:
    print(msg) # Выведет: Описание исключения
```

```
Описание исключения
```

In [12]:
```
try:
    raise ValueError
except ValueError:
    print('Сообщение об ошибке')
```

```
Сообщение об ошибке
```

Инструкция assert.

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

assert <Логическое выражение> [, <Сообщение>]

Пример 5.3.

Сгенерировать исключение при всех отрицательных значениях *x*.

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

Пример 5.4.

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

In [1]:
```
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)
def f(k,g):
    try:
        print(cathetus_2(k,g))
    except AssertionError as t:
        print(t)
f(3,5)
f(5,5)
f("1",5)
```

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

Задачи.

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 ситуацию при условии, что такого файла не существует, или не верно задан путь к файлу.

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

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

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

7.Написать функцию подсчета суммы главной диагонали. Если матрица не является 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. Если файла нет, то возвращает текст "404. File (filename) not found", иначе возвращает содержимое файла.

12. Написать калькулятор для строковых выражений вида '<число> <операция> <число>', где <число> - целое число, например 113, <арифмитическая 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 3 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.

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

Тесты по разделу 5.

1. *Инструкция raise используется для* _____ исключений.

```
try:
    raise Exception("some exception")
except Exception as e:
    print("ExceptionError: " + str(e))
```

2. *Если вызов функции предшествует ее определению, то возникает исключение* _____.

3. *Что выведет данный программный код?*

```
print("start")
try:
  val = 0
  tmp = 10 / val
  print(tmp)
except ValueError as ve:
  print("ValueError! {0}".format(ve))
except ZeroDivisionError as zde:
  print("ZeroDivisionError! {0}".format(zde))
```

except Exception as ex:

print("Error! {0}".format(ex))

finally:

print("Finally code")

print("stop")

4. Какой базовый класс исключений перехватывает исключения классов FloatingPointError, OverflowError и ZeroDivisionError?

5. Инструкция assert возбуждает исключение AssertionError, если логическое выражение возвращает значение_____.

6. Какое сообщение выведет программа

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

7. Какое исключение сработет при данном программном коде

print("Нет завершающей кавычки!)

8. Инструкция raise используется для _____

исключений.

```
try:
    raise Exception("some exception")
except Exception as e:
    print("ExceptionError: " + str(e))
```

9. *Если вызов функции предшествует ее определению, то возникает исключение*

_____.

6. Встроенные функции высшего порядка

Рассмотрим некоторые встроенные функции высшего порядка, которые часто используются на практике.

map(*f*, **args*) - преобразует элементы заданных последовательностей с помощью функции *f* в новый объект, поддерживающий итерацию. Полученный объект не является списком. Преобразовать его в список можно с помощью *list*().

Получить список длин строк для заданного списка строк.

```
In [3]: new=list(map(len,['aa','bbb','cccc']))
        new

Out[3]: [2, 3, 4]
```

Получить список, составленный из попарных произведений двух исходных списков.

```
a=[1,2,3]
b=[10,20,30]
new=list(map(lambda x,y: x*y,a,b))
new

Out[6]: [10, 40, 90]
```

filter (*f*, *seq*) Позволяет отобрать элементы последовательности, удовлетворяющие условию. Функция *f* должна возвращать *True* для элементов, включаемых в результат, и *False* в противном случае. Второй параметр - исходная последовательность. Если в первом параметре указать значение *None* вместо имени функции, то каждый элемент исходной последовательности будет проверен на соответствие значению *True*.

Удалить пустые элементы из списка.

#1 вариант

a=[1,0,'*a*',",[],3]

new=*list*(*filter*(*None*,*a*))

#2 вариант

a=[1,0,'*a*',",[],3]

new=*list*(*filter*(*lambda x*: *x if x*!=" *else* 0,*a*))

#3 вариант

a=[1,0,'*a*',",[],3]

new=*list*(*filter*(*lambda x*: *x*!=" *and x*!=[] *and x*!=0,*a*))

Убрать из списка все числа, которые при делении на 15, дают остаток 2.

#1 вариант

a=[1,2,12,15,17,32]

new=*list*(*filter*(*lambda x*: *x if x*%15!=2 *else* 0,*a*))

#2 вариант

a=[1,2,12,15,17,32]

new=list(*filter*(*lambda x*: *x*%15!=2,*a*))

reduce (*f, seq*[,*initial*]). Применяет указанную функцию *f* к парам элементов и накапливает результат. Функция *f* должна иметь 2 параметра. При первом обращении к *f* ее параметрами являются 2 первых элемента последовательности. При последующем обращении в качестве первого элемента используется значение, которое *f* вернуло на предыдущем шаге, в качестве второго элемента - очередной элемент последовательности. Функция *reduce*() находится в модуле *functools*. *initial* - начальное значение, к которому добавляется результат работы *reduce*.

from functools import reduce

new=reduce(*lambda x,y*: *x+y*,[1,2,3,4,5])

from functools import reduce

new=reduce(*lambda x,y*: *x+y*,[1,2,3,4,5],100)

sorted(*seq*, *key=None*, *reverse=False*) - возвращает новый список, содержащий все элементы исходной последовательности в порядке возрастания значений элементов (по умолчанию). В параметре *key* можно указать ссылку на функцию с одним параметром, возвращающую значение. В этом случае сортировка выполняется следующим образом: для каждого элемента последовательности вычисляется значение функции, указанной в *key*. Метод *sort*() применяется только для списков.

#отсортировать список строк с использованием различных значений параметров (по возрастанию, убыванию, длине)

l=['aaa','bb','cdd','d','a']

l1=sorted(l)

print(l1)

l2=sorted(l,reverse=True)

print(l2)

l3=sorted(l,key=len)

print(l3)

```
In [27]: #дан список со словами,которые содержат символы в разных регистрах. Отсортировать список
         l=['маМА','Юля','нАдЯ']
         l1=sorted(l)
         l1
Out[27]: ['Юля', 'маМА', 'нАдЯ']

In [26]: l=['маМА','Юля','нАдЯ']
         l1=sorted(l,key=str.lower)
         l1
Out[26]: ['маМА', 'нАдЯ', 'Юля']
```

#с помощью sorted сформировать последовательность чисел от 1 до 9 так, чтобы остатки от деления на 3 этих чисел шли

#в порядке возрастания

l=sorted(range(1,10), key=lambda x: x%3)

Пример 6.1.

При помощи *map*, *filter*, *reduce* возвести в квадрат числа от 1 до 100 и рассчитать их сумму, не включая в сумму числа кратные 9.

from functools import reduce

*new=reduce(lambda x,y: x+y, filter(lambda x: x%9!=0,list(map(lambda x: x**2,range(1,101)))))*

Задачи.

1. При помощи функций *map*/*filter*/*reduce* возвести в квадрат числа от 1 до 10, и рассчитать их сумму, не включая в сумму четные числа.

2. При помощи функций *map*/*filter*/*reduce* превратить список целых чисел в строку, содержащую строковое представление этих чисел, разделенных пробелами.

3. При помощи функций *map*/*filter*/*reduce* из несколько одинаковых подряд идущих элементов списка оставить только один. Например, [1, 2, 3, 4, 4, 4, 5, 6, 6, 7, 6, 1, 1] -> [1, 2, 3, 4, 5, 6, 7, 6, 1]

4. При помощи функций *map*/*filter*/*reduce* из списка списков извлечь элементы, содержащиеся во вложенных списках по индексу 1. Например, [[1, 2, 3], [2, 3, 4], [0, 1 , 1 , 1], [0, 0]] -> [2, 3, 1, 0]

5. Дан список A, состоящий из $2N$ элементов. Разбейте его на списки B и C по N элементов каждый так, чтобы каждый элемент B не превосходил каждого элемента C.

6. В списке $2n + 1$ различных элементов. Найдите средний элемент списка. Под средним элементом понимают такой, для которого в списке n элементов больше его и n элементов меньше.

7. Дан список из N элементов. Вывести индексы списка в том порядке, в котором соответствующие им элементы образуют возрастающую последовательность.

8. С помощью функции reduce() вычислить двойной факториал заданного натурального числа *n* (для нечетного *n*).

9. С помощью функции reduce() вычислить двойной факториал заданного натурального числа *n* (для четного *n*).

10. Дан список A3, состоящий из четного количества элементов. Используя функцию(функции) высшего порядка разбейте его на списки B, C так, чтобы в одном были положительные элементы, а в другом отрицательные.

11. Дан список S состоящий из N различных элементов. Вывести индексы четных элементов списка. Использовать функции высшего порядка.

12. Для списков вида [[‘Иванов’, 3,5,4], [‘Петров’, 4, 5, 5], [‘Сидоров’, 3, 3, 3], [‘Николаев’, 4, 4, 3]] напишите функцию, которая выводит на экран этот список в виде таблицы в отсортированном виде. Параметрами функции являются список, функция, которая определяет порядок сортировки (значение параметра *key* в *sort*), и параметр, определяющий, выполняется сортировка по возрастанию или убыванию. Функция не возвращает значение и не изменяет исходный список. Используя эту функцию, выведите исходный список, отсортированный:

• по фамилиям в алфавитном порядке;

• в порядке возрастания первой оценки;

• в порядке убывания суммы баллов.

Тесты по разделу 6.

1. Какие выражения генерируют список:

1) list(lambda x: x**2, range(10))
2) list(map(lambda x: x**2, range(10)))
3) [x**2 for x in range(10)]
4) [map(lambda x: x**2, range(10))]

2. Что выведет данный код:

```
from functools import reduce
sp = [1, 2, 3]
reduce(lambda x,y: x*y, sp)
```

1) 1
2) 2
3) 3
4) 6

3. *Из какого модуля нужно импортировать функцию reduce?*

```
from            import reduce
lst=[2,4,6,8,10,8,6,4,-22]
cumulat=reduce(lambda x,y: x+y, lst)
print(cumulat)
```

4. *Результат работы программы*

a=[123,456,789,0]

print(':'.join(map(str,a)))

равен_____

5. *Как определить тип полученного результата*

a=[123,456,789,0]

print(':'.join(map(str,a)))

1) type(print(':'.join(map(str,a))))

2) print(type(':'.join(map(str,a))))

3) print(':'.join(type(map(str,a))))

6. *Чему равен результат работы программы*

```
s = [[5,3],[1,2,34,5],[9,0,5,4],[11,23,4,6],[1,66,7]]
list(map(lambda x:x[1],s))
```

7. *Чему равен результат работы программы*

```
lst=[1,2,6,77,88,44,6677]
med=sorted(lst)[int(len(lst)/2)]
med
```

8. *Чему равен результат работы программы*

```
lst=[11,2,6,7,8,4,6]
med=sorted(lst, key=lambda x:x%2)
med
```

9. *Выберите правильный результат*

```
lst=[11,2,6,7,8,4,6]
med=sorted(lst, key=lambda x:x//2)
med
```

1) 2,4,6,6,7,8,11

2) 2,4,6,7,6,8,11

3)2,4,6,6,8,7,11

7. Работа с файлами.

7.1. Работа с текстовыми файлами.

Прежде чем работать с файлом, необходимо создать объект файла с помощью функции *open*() и правильно указывать путь к файлу. Путь может быть абсолютным или относительным. В *Windows* следует учитывать, что слэш является специальным символом. По этой причине слэш необходимо удваивать или вместо обычных строк использовать неформатированные строки.

```
f = r"C:\temp\new\file.txt"
f = "C:\\temp\\new\\file.txt"
```

Режимы доступа к текстовым файлам:

'*r*' – чтение из текстового файла. Если файл не существует, то выдается сообщение об ошибке (режим по умолчанию);

'*w*' – запись в текстовый файл. Если файл существует, то содержимое заменяется. Если файл не существует, то он создается;

'*a*' – дозапись в текстовый файл. Если файл существует, то данные дописываются в конец. Если файл не существует, то он создается.

Пример открытия файла на запись и закрытие файла:

```
f1 = open('new_file.txt', mode='w')
f1.close()
```

или

```
f1 = open('new_file.txt', 'w')
f1.close()
```

Чтение текстового файла может осуществляться с помощью методов: *read*() позволяет прочесть из файла указанное количество символов; *readline*() позволяет прочитать указанное количество символов текущей строки; *readlines*() читает текстовый файл в список, элементами которого являются строки файла.

Запись в текстовый файл может осуществляться с помощью методов: *write*() записывает строку в файл; *writelines*() записывает элементы списка строк в файл.

Пример 7.1.

Создать в текстовом редакторе файл, содержащий несколько строк. Определить максимальный и минимальный размер строки в файле и вывести их в другой файл.

```
f=open('D:\\Python\\Maf.txt', 'r')
s=open('D:\\Python\\Res.txt','a')
x=f.readline() #считываем первую строку файла
max=x
min=x
while x!='':    #пока не конец строки
    if len(max)<len(x):
        max=x
    if len(min)>len(x):
        min=x
    x=f.readline()   #считываем строку файла
s.write(str(len(min)))
s.write(min)
s.write(str(len(max)))
s.write(max)
print('Минимальное число символов в строке =',len(min),' ',min)
print('Максимальное число символов в строке =',len(max),' ',max)
```

Задачи по работе с текстовыми файлами.

1. Создайте текстовый файл, содержащий информацию о товарах и ценах на них. Каждая строка файла имеет вид: НАЗВАНИЕ ТОВАРА: ЦЕНА. Используя данный файл:

• найдите цену указанного товара, или выдайте сообщение о том, что цена не известна;

• добавьте в файл информацию о трех новых товарах;

• удалите из файла информацию о товаре;

• создайте новый файл, в котором товары будут упорядочены в порядке возрастания цен. Выведите на экран информацию о двух самых дешевых и двух самых дорогих товарах.

2. Создайте два текстовых файла, содержащие целые числа, в которых данные отсортированы по убыванию. Каждая строка содержит одно число. Сформируйте новый файл из чисел входных файлов, чтобы его значения были также отсортированы по убыванию (не использовать методы сортировки).

3. Написать функцию, которая просит пользователя построчно ввести содержимое текстового файла. Сначала запрашивается имя файла. После ввода пустой строки ввод заканчивается и файл сохраняется. Вызвать эту функцию, корректно обработать исключительные ситуации, возникающие при работе функции.

4. Написать функцию, которая просит пользователя построчно вывести содержимое текстового файла. Сначала запрашивается имя файла. После ввода пустой строки ввод заканчивается и файл сохраняется. Вызвать эту функцию.

5. Посчитать сколько раз в файле встречается символ «!». Вывести строки, в которых встречается данный символ.

6. Создайте матрицу размера $n \times m$ элементами которой являются случайные целые числа из диапазона [-10, 10]. Полученную матрицу сохраните в файл matrix.txt построчно.

7. В текстовом файле третью строку переписать в новый файл в обратном порядке.

8. Дописать в файл строку «В лесу родилась елочка» между второй и третьей строкой исходного файла.

9. Написать функцию, которая построчно выводит на экран содержимое текстового файла. У функции должен быть необязательный параметр, при помощи которого можно задать максимальное количество строк, выводимых на экран.

10. Написать функцию, которая составляет статистику использования слов в текстовом файле (в файле не используются переносы слов на новую строку). Итогом работы является словарь, в котором ключами являются слова, а значениями целые числа, определяющие сколько раз слово встретилось в файле.

11. Программа должна допускать следующие режимы работы:

- Генерация *n* действительных чисел из заданного диапазона и их запись в файл (.txt).

- Чтение всех чисел из файла и вывод их на консоль.

- Выполнение над числами из файла указанных действий и вывод результатов на консоль: вычислить сумму и произведение всех чисел; найти наибольшее число; найти наименьшее число; вычислить сумму положительных чисел; вычислить сумму отрицательных чисел.

12. Дан файл f, компоненты которого являются случайными целыми числами. Никакая из компонент файла не равна нулю. Файл f содержит столько же отрицательных чисел, сколько и положительных. Используя вспомогательный файл h, переписать компоненты файла f в файл g так, чтобы в файле g сначала шли нечетные потом четные числа.

7.2. Работа с файлами формата CSV.

Файл *CSV* (значения, разделенные запятыми) позволяет сохранять данные в табличной структуре с расширением *.csv*. Модуль *CSV* имеет несколько функций и классов, доступных для чтения и записи *CSV*: функция *csv.reader*; функция *csv.writer*; *DictReader* и *DictWriter*. *DictReader* и *DictWriter* - это классы, доступные в *Python* для чтения и записи в *CSV*. Хотя они и похожи на функции чтения и записи, эти классы используют объекты словаря для чтения и записи в *CSV*-файлы. *DictReader* создает объект, который отображает прочитанную информацию в словарь, ключи которого задаются параметром *fieldnames*. Этот параметр является необязательным, но, если он не указан в файле, данные первой строки становятся ключами словаря. *DictWriter* выполняет запись данных в файл *CSV* (параметр *fieldnames* определяет последовательность ключей, которые определяют порядок, в котором значения в словаре записываются в файл *CSV*.)

Диалект - это вспомогательный класс, используемый для определения параметров для конкретного экземпляра *reader* или *writer*. Диалекты и параметры форматирования должны быть объявлены при выполнении функции чтения или записи. Есть несколько атрибутов, которые поддерживаются диалектом: *delimiter* – строка, используемая для разделения полей (по умолчанию это “,”); *lineterminator* – строка, используемая для завершения строк, созданных *writer* (по умолчанию используется значение”\r\n”).

Пример 7.2.

Создать в *Excel* файл формата *.csv*, содержащий заголовок и несколько строк. Прочесть построчно данный файл.

```
import csv
with open('Proba_fail.csv', newline='') as File:
    reader = csv.reader(File)
    for row in reader:
        print(row)
```

```
['Дети;Возраст;Рост']
['Маша;5;116']
['Вася;4;110']
['Федя;9;142']
```

Пример 7.3.

Создать в *Excel* файл формата *.csv*, содержащий заголовок и несколько строк. В *header* прочесть заголовок, а в *rows* – остальные строки файла.

```
import csv
with open('Proba_fail3.csv') as f:
    f_csv = csv.reader(f)
    header = next(f_csv)
    rows = [r for r in f_csv]
header
rows
```

Пример 7.4.

Прочесть данные *csv*-файла с помощью *DictReader* и вывести результат в виде списка словарей.

```
results = []
with open('Proba_fail.csv') as File:
    reader = csv.DictReader(File)
    for row in reader:
        results.append(row)
    print (results)
```

```
[{'Дети;Возраст;Рост': 'Маша;5;116'}, {'Дети;Возраст;Рост': 'Вася;4;110'}, {'Де
ти;Возраст;Рост': 'Федя;9;142'}]
```

Пример 7.5.

Прочесть данные *csv*-файла с помощью *DictReader* и вывести результат в текстовом виде. Посчитать количество строк в файле.

```
import csv
with open('Proba_fail.csv') as f:
    f_csv = csv.DictReader(f, delimiter=';')
    count = 0
    # Считывание данных из CSV файла
    for row in f_csv:
        if count == 0:
            # Вывод строки, содержащей заголовки для столбцов
            print(f'Файл содержит столбцы: {", ".join(row)}')
        # Вывод строк
        print(f' {row["Дети"]} -ростом {row["Рост"]}', end='')
        print(f' Этому ребенку {row["Возраст"]} лет.')
        count += 1
    print(f'Всего в файле {count} строк.')
```

```
Файл содержит столбцы: Дети, Возраст, Рост
 Маша -ростом 116 Этому ребенку 5 лет.
 Вася -ростом 110 Этому ребенку 4 лет.
 Федя -ростом 142 Этому ребенку 9 лет.
Всего в файле 3 строк.
```

Пример 7.6.

Записать данные в *csv*-файл с помощью *writer*.

1 способ.

```
import csv
with open('Proba_fail3.csv','w') as f:
    f_csv = csv.writer(f, delimiter=';',lineterminator="\r")
    f_csv.writerow(["Имя", "Класс", "Возраст"])
    f_csv.writerow(["Женя", "3", "10"])
    f_csv.writerow(["Саша", "5", "12"])
    f_csv.writerow(["Маша", "11", "18"])
    #Обратите внимание, что при записи использовался, lineterminator="\r".
    #Это разделитель между строками таблицы, по умолчанию он "\r\n".
```

2 способ.

```
myData = [["aa", "ss", "dd"],
          ['zz', 'xx', 'cc'],
          ['A', 'D', 'B']]
f = open('Proba_fail1.csv', 'w')
with f:
    f_csv = csv.writer(f, delimiter=';',lineterminator="\r")#,lineterminator="\r
    f_csv.writerows(myData)
```

Пример 7.7.

Записать данные в *csv*-файл с помощью *DictWriter*, включая заголовок файла.

```python
import csv
with open('Proba_fail4.csv','w') as f:
    imya = ["Имя", "Класс", "Возраст"]
    f_csv = csv.DictWriter(f,delimiter=';',lineterminator="\r", fieldnames=imya)
    f_csv.writeheader()
    f_csv.writerow({"Имя": "Саша", "Возраст": "6"})
    f_csv.writerow({"Имя": "Маша", "Возраст": "15"})
    f_csv.writerow({"Имя": "Вова", "Возраст": "14"})
print("Всё ок")
```

Задачи по работе с файлами формата CSV.

1. Реализовать функцию *find_by_name*, в которую можно передать произвольное количество параметров, в результате функция вернет данные (в виде списка списков) только по людям, у которых в столбце *'First Name'* содержатся указанные имена.

2. Программа должна допускать следующие режимы работы:

- Генерация *n* действительных чисел из заданного диапазона и их запись в файл (*.csv*).

- Чтение всех чисел из файла и вывод их на консоль.

- Выполнение над числами из файла указанных действий и вывод результатов на консоль: вычислить сумму и произведение всех чисел; найти наибольшее число; найти наименьшее число; вычислить сумму положительных чисел; вычислить сумму отрицательных чисел.

3. Создайте матрицу размера $n \times m$ элементами которой являются дни недели, случайно выбранные из списка. Полученную матрицу сохраните в файл matrix.csv.

4. В *csv* файле с продуктами уменьшите стоимость тех товаров, которые имеют цену больше средней всех товаров, выписав отдельным столбцом на сколько она изменилась (или "*None*", если не изменялась).

5. Напишите функцию для записи в два файла данных, разделенных точкой с запятыми (*CSV, Comma Separated Values*). Функция должна получать в параметрах два имени файла и список кортежей. Кортежи должны содержать имя, возраст и пол. Функция должна создать строку заголовка в каждом файле, за которой следуют строки для каждого кортежа. Если пол мужской, то записываем кортеж в файл *man.txt*, если пол женский, то записываем очередной кортеж в файл *woman.txt*.

Если функции будет передан следующий список кортежей:
[('Сергей', 22, 'м'),
('Светлана', 32, 'ж'),
('Ирина', 27, 'ж'),
('Евгений', 45, 'м')]
то в файл man.txt должны быть записаны следующие данные:
Сергей, 22, м
Евгений, 45, м
а в файл woman.txt должны быть записаны следующие данные:
Светлана, 32, ж
Ирина, 27, ж

7.3. Работа с модулем pickle.

При создании объекта файла с помощью функции *open*() после режима чтения или записи может следовать модификатор:

b - файл будет открыт в бинарном режиме. Файловые методы принимают и возвращают объекты типа *bytes*;

t - файл будет открыт в текстовом режиме (значение по умолчанию в *Windows*). Файловые методы принимают и возвращают объекты типа *str*. В этом режиме в *Windows* при чтении символ *\r* будет удален, а при записи, наоборот, добавлен.

Сохранить объекты в файл и в дальнейшем восстановить объекты из файла позволяет модуль *pickle*.

Модуль *pickle* предоставляет следующие функции:

– *dump*(<Объект>, <Файл>) - производит сериализацию объекта и записывает данные в указанный файл. В параметре <Файл> указывается файловый объект, открытый на запись в бинарном режиме.

– *load*(<Файл>) - читает данные из файла и преобразует их в объект. В параметре <Файл> указывается файловый объект, открытый на чтение в бинарном режиме.

```
with open("my.txt", "rt") as f:
    for line in f:
        print(type(line), end='')
        print(repr(line))
with open("my.txt", "rb") as f:
    for line in f:
        print(type(line), end='')
        print(repr(line))
```

```
<class 'str'>'s1\n'
<class 'str'>'s2\n'
<class 'bytes'>b's1\r\n'
<class 'bytes'>b's2\r\n'
```

Пример 7.8.

Записать данные в текстовый файл и считать эти данные, используя модуль pickle.

```
obj = ["Строка", (12, 3)] # объект для сохранения
import pickle # подключаем модуль pickle
with open('new_pickle.txt', 'wb') as f:
    pickle.dump(obj, f)
```

```
with open('new_pickle.txt', 'rb') as f:
    obj_l = pickle.load(f)
obj_l
```

```
['Строка', (12, 3)]
```

```
obj2 = (6, 7, 8, 9, 10)
with open('obj2.txt', 'wb') as f:
    pickle.dump(obj, f)
    pickle.dump(obj2, f)
with open('obj2.txt', 'rb') as f:
    obj_l2 = pickle.load(f)
    obj2_l = pickle.load(f)
obj_l2, obj2_l
```

```
(['Строка', (12, 3)], (6, 7, 8, 9, 10))
```

Пример 7.9.

Считать данные из текстового файла *myfile.txt*, в котором расположен массив. Отсортировать числа полученного массива по возрастанию, потом остальные элементы нечислового типа данных. Перезаписать результирующий массив в *myfile.txt*. Конечный десериализированный результат из файла выведите на экран. Стартовый файл содержит данные: [*'hbac'*, [5,6,7], 1, 4, 8, *True*, 1398, 61, {*'rrr'*:3}, -4, 24]

```
with open('myfile.txt', 'rb') as infile1:
    identical = pickle.load(infile1)
bufer=[]# массив для работы с числами

for i in range (len(identical)-1,-1,-1):
#обратный цикл для извлечения числовых элементов в отдельный массив bufer
    if type(identical[i])==int: #если переменная целочисленная
        bufer.append(identical[i])
        del identical[i]
bufer=sorted(bufer) #сортируем все числа по возрастанию
identical=bufer+identical #добавляем в начало основного массива отсартированные числа

with open('myfile.txt', 'wb') as outfile:
    pickle.dump(identical, outfile)

with open('myfile.txt', 'rb') as infile:
    identical = pickle.load(infile)
print(identical)
```

```
[-4, 1, 4, 8, 24, 61, 1398, 'hbac', [5, 6, 7], True, {'rrr': 3}]
```

Задачи по работе с модулем pickle.

1. Осуществить чтение/запись множества с разнотипными данными, используя модуль pickle.

2. Собрать статистику использования слов для текста revisor.txt и сохранить результаты в отдельных файлах. Написать функцию, которая на основе файлов статистики рассчитает статистку использования слов в тексте. Сохранить результат в виде отдельного текстового файла, в котором на каждой строке располагается информация по отдельному слову в формате: <слово> <сколько раз встретилось>. Список слов должен быть отсортирован по частоте слов (сначала идут наиболее распространенные слова).

Тесты по разделу 7.

1. *Выбери правильную запись указания пути к текстовому файлу:*

1) f = open("C:\\temp\\new\\file.txt")

2) f = open("C:\temp\new\file.txt")

3) f = open(r"C:\temp\new\file.txt")

4) f = open(r"C:\\temp\\new\\file.txt")

2. Поставить в соответствие методу связанное с ним действия:

1) file.close()	a) возвращает текущую позицию в файле
2) file.next()	b) добавляет последовательность строк в файл
3) file.read(n)	c) чтение первых n символов файла
4) file.readline()	d) добавляет строку str в файл
5) file.readlines()	e) читает и возвращает список всех строк в файле
6) file.tell()	f) закрывает открытый файл
7) file.write(str)	g) читает одну строчку строки или файла
8) file.writelines(sequence)	h) возвращает следующую строку файла

3. Текстовый файл new_file7.txt содержит данные:

Строка1
Строка2
Что выведет данный программный код:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    print(f.read(4))
    print(f.read(4))
```

4. Текстовый файл new_file7.txt содержит данные:

Строка1
Строка2
Что выведет данный программный код:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()
lines
```

5. Текстовый файл new_file7.txt содержит данные:

Строка1
Строка2
Что выведет данный программный код:

```
with open("new_file7.txt", "r", encoding="utf-8") as f:
    for l in f:
```

```
print(l)
```

6. *Как вывести заголовок файла формата *.csv*

```
import csv
with open('d:\ФинУнивер\ fail3.csv') as f:
    f_csv = csv.reader(f)
1) header = f_csv
2) header = next (next(f_csv))
3) header = next(f_csv)
4) header = next(csv.f)
```

7. *Напиши содержимое csv файла после выполнения программы:*

```
import csv
with open('d:\ФинУнивер\fail4.csv','w') as f:
    imya = ["Имя", "Класс", "Возраст"]
    f_csv = csv.DictWriter(f,delimiter=';',lineterminator="\r", fieldnames=imya)
    f_csv.writeheader()
    f_csv.writerow({"Имя": "Саша", "Возраст": "6"})
    f_csv.writerow({"Имя": "Маша", "Возраст": "15"})
    f_csv.writerow({"Имя": "Вова", "Возраст": "14"})
```

8. *При чтении csv файла с помощью DictReader данные записываются в виде:*

1)списка

2)словаря

3)кортежа

9. *Выбери правильные варианты. Модуль CSV имеет несколько функций и классов, доступных для чтения и записи CSV, и они включают в себя:*

1) csv.reader

2) csv.DictReader

3) csv.DictWrite

4) csv.writer

5) csv.DictRead

10. *Атрибут файлового объекта file.closed возвращает _____ если файл закрыт и _____ в противном случае.*

8. Модули (*numpy, matplotlib*). Собственные модули.

Модуль в языке *Python* - это обычный файл с расширением *.*py*. Импортирование модуля может выполняться разными способами для следующих случаев: импортирование одного модуля (*import math*); импортирование нескольких модулей (*import math, collections*); для импортируемого модуля задается произвольное имя (*import numpy as np*).

Основным объектом *NumPy* является однородный многомерный массив (в *numpy* называется *numpy.ndarray*). Это многомерный массив элементов (обычно чисел), одного типа.

Наиболее важные атрибуты объектов *ndarray*:

ndarray.ndim - число измерений (чаще их называют "оси") массива.

ndarray.shape - размеры массива, его форма. Это кортеж натуральных чисел, показывающий длину массива по каждой оси. Для матрицы из *n* строк и *m* столбов, *shape* будет (*n*,*m*). Число элементов кортежа *shape* равно *ndim*.

ndarray.size - количество элементов массива. Очевидно, равно произведению всех элементов атрибута *shape*.

ndarray.dtype - объект, описывающий тип элементов массива.

Пример 8.1.

С помощью модуля *numpy* создать массив *x* от -π до +π (с небольшим произвольным шагом). Для заданного списка рассчитать значения функций $y1 = 2sinx$ и $y2 = cos2x$. Построить на плоскости графики данных функций, задав необходимые атрибуты графиков, с помощью модуля *matplotlib*.

```
import numpy as np # импортируем модули
import matplotlib.pyplot as plt # импортируем модули
x = np.arange(-3.14, 3.14, 0.1)
y = 2*np.sin(x) # функция
e = np.cos(2*x)
fig = plt.subplots() # создаём рисунок и координатную плоскость
plt.plot(x, y,linewidth=2)
plt.plot(x, e,linewidth=4)# - отображается график
plt.grid()
plt.show()
```

Пример 8.2.

Получить индексы, для которых элементы массивов *a* и *b* совпадают. Например: *a* = *np.array*([1,2,3,2,3,4,3,4,5,6]); *b* = *np.array*([7,2,10,2,7,4,9,4,9,8]). Ожидается результат: *array*([1, 3, 5, 7])

```
import numpy as np
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
print(a==b)
print(np.array(len(a)))
c=np.arange(len(a))[a==b]
print(c)
print (a[a==b])
```

```
[False  True False  True False  True False  True False False]
10
[1 3 5 7]
[2 2 4 4]
```

Пример 8.3.

Создать матрицу 8 на 10 из случайных целых (используя модуль *numpy.random*) чисел из диапазона от 0 до 10 и найти в ней строку (ее индекс и вывести саму строку), в которой сумма значений минимальна.

```
import numpy as np
s=np.random.randint(11, size=(8,10))
print(s)
stroki=s.sum(axis=1)
print(stroki,stroki.shape)
minim=np.min(stroki)
print('Сумма значений минимальной строки ', minim)
c=np.arange(len(stroki))[stroki==minim]
print('Индекс строки с минимальной суммой  ', c)
print('Строка исходного массива с минимальной суммой  ',s[c])
```

Создание собственных модулей.

Для создания собственного модуля необходимо создать файл с расширением *.py*. Например, создадим файл *mod_test*1.*py* и в нем напишем следующий программный код:

```
def f1(x,y):
  c=x+y
  return(c)
```

Для подключения собственного модуля *mod_test*1.*py* необходимо сделать:

```
import mod_test1
mod_test1.f1(6,3)
```

Всякий раз, когда файл с расширением *.py* запускается как программа, интерпретатор *Python* создает в программе переменную с именем *_name_* и записывает в нее строку "__main__". Это можно использовать для того чтобы выполнять некоторый программный код только если файл запускается как программа.

Модуль *sys* позволяет также получить те слова, которые были введены в команде, запустившей сценарий на языке *Python*. Эти слова обычно называются аргументами командной строки и находятся во встроенном списке строк *sys.argv*.

Создаем собственный модуль *mod_test.py*:

```
def f1(lst):
```

```
    c=int(lst[1])+int(lst[2])
    return(c)
if __name__=='__main__':
    import sys
        print(f1(sys.argv))
```

Для запуска через *Jupiter Notebook* необходимо прописать, например:

%run mod_test 5 6

Пример 8.4.

Написать программу, вычисляющую разность температур разных шкал измерений. Температуры могут быть в Цельсия, фаренгейтах или кельвинах. Ответ должен быть рассчитан в соответствии со шкалой первой вводимой температурой. Для перевода шкал используйте пользовательский модуль. Например: вводится 78 k и 1 c. Следовательно, результат равен -196 k.

```
import модуль #подключаем пользовательский модуль
#считываем первое вводимое значение и шкалу измерения, введённые через пробел.
num1, sis1=input().split()
num2, sis2=input().split()
num1,num2 = int(num1), int(num2)
#для работы в первой шкале температур, переведём вторую температуру в нужную систему
if sis1=='k':
    num2 = модуль.kelv(num2,sis2)
elif sis1=='f':
    num2 = модуль.far(num2,sis2)
else:
    num2 = модуль.cels(num2,sis2)
res = num1 - num2
print(res, sis1)
```

Пользовательский модуль:

```
def cels(n,nach_sis):
    if nach_sis=='f':
        return ((n-32)*5)/9
    if nach_sis=='k':
        return (n-273)
def kelv(n,nach_sis):
    if nach_sis=='f':
        n=cels(n,'f')
    return (n+273)
def far():
    if nach_sis=='k':
#если мы переводим из кельвинов в фарингейты,
#тогда сначала кельвины переведём в цельсия
        n=cels(n,'k')
    return ((9*c)/5)+32
```

Задачи.

1. Написать функцию *Smooth*(*R*, *n*), заменяющую каждый случайный элемент вещественного массива *R* размера *n* на его среднее арифметическое со своими ближайшими соседями ("сглаживание массива"). Массив *R* – входной и выходной параметр, *n* – входной параметр. С помощью этой процедуры выполнить пятикратное сглаживание данного

массива *R* размера *n*, выводя на экран результаты каждого сглаживания в виде массива и графика.

2. Написать программу построения графика заданной функции на заданном отрезке [x_1; x_2]. График должен содержать оси, значения по осям. Единицы масштаба по осям *x*, *y* должны совпадать (для контроля вывести график функции *y=x*. График должен иметь сетку. Программа должна допускать построение графика функции с другими заданными коэффициентами (например, если основная функция $\sin(x)$, то программа должна допускать построение функции $a \cdot \sin(bx + c) + d$. График построить для функции $x + e^{\sin(x)}$ на заданном отрезке [x_1; x_2].

3. Написать функцию вычисления y:

$$y = \begin{cases} 3x^2 & x > 4{,}5 \\ e^{-x} & 1 \le x \le 4{,}5, \\ -cos^2(2x) & x < 1 \end{cases} \qquad x \in [-\frac{\pi}{2}; 2\pi]$$

и на заданном промежутке с шагом *h* (значение шага задаётся) построить график переменной *y*. Использовать модули *matplotlib* и *numpy*.

4. Для заданной математической функции y=f(x) разработать программу построения графика функции на интервале от 4 до 8,9. На графике предусмотреть наличие заголовка и подзаголовка, осей, подписей осей и линий сетки.

$$y = \frac{x^2 - 18x + 77.5}{x - 9}$$

5. Создать в цикле список значений *x* от -π до +π (с небольшим произвольным шагом). Для заданного списка рассчитать значения функций $y1 = 2sinx$ и $y2 = cos2x$. Построить на плоскости графики данных функций, задав необходимые атрибуты графика (заголовок, оси координат и их подписи, линии сетки, легенду и т.п.). Найти (приблизительно) координаты пересечения графиков данных функций.

6. Для заданной математической функции y=f(x) и ее производной y'=f'(x) разработать программу построения графиков функции и ее производной на интервале от -10 до +10. На графике предусмотреть наличие заголовка и подзаголовка, осей, подписей осей, линий сетки и легенды. Оценить, есть ли на указанном интервале локальные экстремумы заданной функции y=f(x).

$$y = x^3 - 3x^2 + x - 2$$

$$y' = 3x^2\text{-}6x\text{+}1$$

7. Для заданной математической функции *y=f(x)* разработать программу построения графика функции на интервалах от -1 до 1,9 и от 2,1 до 5. На графике предусмотреть наличие заголовка и подзаголовка, осей, подписей осей и линий сетки.

$$y = \frac{x^2 - 3x + 6}{x - 2}$$

8. С помощью генератора создать в цикле список значений *x* от -π до +π (с небольшим произвольным шагом). Для заданного списка рассчитать значения функций $y1 = 2sinx$ и $y2 = cos2x$. Построить на плоскости графики данных функций, задав необходимые атрибуты графика (заголовок, оси координат и их подписи, линии сетки, легенду и т.п.). Найти (приблизительно) координаты пересечения графиков данных функций.

9. Для заданной математической функции *y=f(x)* разработать программу построения графика функции на интервале от 3,1 до 6. На графике предусмотреть наличие заголовка и подзаголовка, осей, подписей осей и линий сетки.

$$y = \frac{x^2 - 9x + 7}{x - 3}$$

10. Из модулей math и numpy извлечь функции переводящие градусы в радианы. Соответственно извлеченным функциям перевести градусы в радианы двумя способами: перевести 75 градусов в радианы; перевести 180 градусов в радианы; перевести 90 градусов в радианы. Сравнить полученные два значения, переведенные из градусов в радианы между собой. Вывести результат в виде полученных значений в радианах и утверждение о равенстве или неравенстве полученных значений при переводе градусов в радианы разными способами.

11. Средствами numpy рассчитать произведения четных чисел от 2 до 20 на ближайшие к ним бОльшие нечетные числа.

12. Создать в numpy матрицу 11 на 7 вида: [[1, 2, 3, ..., 7], [11, 12, 13, ..., 17], [21, 22, 23, ..., 27], ..., [101, 102, 103, ..., 107]]

13. Сгенерировать двухмерный массив arr размерности (4, 7), состоящий из случайных действительных чисел, равномерно распределенных в дипазоне от 0 до 20. Нормализвать значения массива с помощью преобразования вида $ax+b$ax+b так, что после нормализации максимальный элемент масива будет равен 1.0, минимальный 0.0.

14. Создать матрицу 8 на 10 из случайных целых (используя модуль numpy.random) чисел из диапазона от 0 до 10 и найти в ней строку (ее индекс и вывести саму строку), в которой сумма значений минимальна.

15. Создать массив из 20 случайных целых чисел от 0 до 100. Обрезать значения массива (заменить значения выходящие за диапазон на крайние значения) снизу по значению 30, сверху по значению 70.

16. Создать две матрицы 30 на 3 из случайных целых чисел из диапазона от 0 до 10 и найти все значения первой матрицы, которые больше соответствующих (по расположению) значений второй матрицы. Подсчитать сумму этих значений.

17. Написать в отдельном модуле функцию *count_day* подсчета количества дней между двумя датами с использованием методов модуля *datetime*. Функция принимает два параметра в формате даты, например, *count_day*(12.01.2021, 15.02.2021) и выдает целое число.

18. Произвести обработку возможных исключений при импортировании модулей, функций из модулей. Обработку исключений рассмотреть на примере модулей и функций для работы с датами и временем.

19. Создайте собственный модуль, поместив в него ваши функции. Напишите любую программу с использованием функций этого модуля.

20. Создать скрипт для анализа тестовых файлов - сбора статистики упоминания слов. Аргументом командной строки является имя файла, который нужно проанализировать. Второй аргумент - количество слов, которые нужно выводить на экран (если аргумент не указан, то выводим статистику по 100 словам, статистика выводится в порядке убывания частоты слов, по каждому слову на экран выводится строка в формате "<слово> <частота>"). Если второй аргумент не целое число, то его рассматриваем как имя файла, в который в формате *CSV* с заголовком сохраняем статистику слов (Первый столбец - слово, второй столбце - частота).

21. Создать три модуля. В первом расположена функция от трех параметров длин сторон параллелепипеда, она вычисляет объем параллелепипеда. Во втором модуле функция вычисляет площадь параллелепипеда. В третьем модуле функция вычисляет периметр сторон параллелепипеда. В основном файле вызывать функции из каждого модуля.

Например, для значений

10, 20, 30

Результаты вызова функций будет

Объем = 6 000

Периметр = 200

Площадь = 2200

Тесты по разделу 8.

1. Установить соответствие между функциями модуля math языка Python и их описанием:

Функция	Описание
math.fabs()	Вычисление факториала

math.fmod()	Возведение числа в указанную степень
math.factorial()	Модуль числа
math.pow()	Остаток от деления X на У

2. *Установить соответствие между функциями модуля math языка Python и их описанием:*

Функция	Описание
math.degress()	Преобразование градусов в радианы
math.radians()	Преобразование радиан в градусы
math.ceil()	Округление до ближайшего меньшего целого числа
math.floor()	Округление до ближайшего большего целого числа

3. *Список ключевых слов в Python можно получить, подключив модуль* _____ .

4. *Какими операторами можно импортировать модуль?*

1) import
2) load
3) include и import
4) from и import
5) include
6) include и load
7) from и load
8) import и from

5. *Установите соответствие между наименованием и содержанием модулей языка Python:*

Наименование	Содержание
os	Содержит библиотеку математических функций и констант, используемых при обработке вещественных чисел
math	Содержит библиотеку функций математической и статистической обработки многомерных массивов и матриц
random	Содержит библиотеку функций для работы с объектами файловой системы
numpy	Содержит библиотеку функций для генерации и обработки случайных чисел
sympy	Содержит библиотеку функций визуализации данных и построения графиков
matplotlib	Содержит библиотеку функций реализации символьных вычислений

6. *Из какого модуля нужно импортировать функцию reduce?*

```
from            import reduce
lst=[2,4,6,8,10,8,6,4,-22]
cumulat=reduce(lambda x,y: x+y, lst)
print(cumulat)
```

7. *Выбери правильные варианты. Модуль CSV имеет несколько функций и классов, доступных для чтения и записи CSV, и они включают в себя:*

1) csv.reader
2) csv.DictReader
3) csv.DictWrite
4) csv.writer
5) csv.DictRead

8. *Для того чтобы отделить исполнение всей программы, размещенной в модуле от его импортирования в языке Python имеется специальная переменная*

1) __name__
2) name
3) main
4) __main__
5) module
6) __module__

9. Объектно-ориентированное программирование.

Класс описывает модель объекта, его свойства и поведение. Класс содержит набор переменных и функций. Переменные называются атрибутами. В них хранятся характеристики объекта (название, размер, цвет, количество и т.д.). Функции называются методами. Метод определяет действие, которое объект может выполнять над самим собой или другими объектами. Синтаксис класса выглядит следующим образом:

class *имя_класса:*
  переменная=значение
  def *имя_метода(self,...,...):*
    self.переменная=значение

Объект – экземпляр класса. Объект хранит конкретные значения атрибутов и информацию о принадлежности к классу, может выполнять методы класса. Создать объект можно так:

имя_объекта=имя_класса([параметры конструктора])

Методы экземпляра класса имеют обязательный первый параметр (*self*), который содержит ссылку на объект, для которого вызван метод.

Пример 9.1.

Создать класс *Car*(), содержащий информацию о машине (марка, модель и номер). Определить несколько методов этого класса.

```
class Car:
    #создаем атрибуты класса
    name='e684ex'
    mark='ford'
    model=2007
    #создаем методы
    def start(self):
        print('Заводи мотор')
    def stop(self):
        print('Заглуши мотор')
#создаем объекты класса
car1=Car()
car2=Car()
print(car1.name)
car1.name="12345" #изменили значение атрибута
print(car1.name)
car2.start()
car2.stop()
```

Существуют методы со специальными именами, предназначенные для выполнения определенных действий. Одним из таких методов является метод __*init*() (в начале и конце имени здесь стоят по два символа подчеркивания), который по аналогии с другими языками программирования называют конструктором. Конструктор автоматически вызывается при создании экземпляра класса. Обычно используется для задания начальных значений атрибутов. Этот метод, как любая функция, может иметь параметры (помимо *self*), которые указываются при создании объекта.

Метод __*str*()__представляет объект в виде строки. Метод вызывается при выводе с помощью функции *print*(), а также при использовании функции *str*().

Пример 9.2.

Создать класс *Bank*(), моделирующий работу банковского счета. Возможные методы класса: изменение суммы банковского счета, вывод информации о количестве счетов владельца и суммах счетов, а также методы *__init__* и *__str__*.

```
class Bank():
   count = 0
   def __init__ (self,number):
      self.number = number
      self.balance = 0
      Bank.count = Bank.count+1
   def __str__(self):
      r = 'Номер счета равен'+str(self.number)
      return r
   def changer(self, balance):
      self.balance = balance
   def display_count (self):
      print('Номер',self.number, 'баланс', self.balance)
      print('Сколько счетов', Bank.count)
```

Пример 9.3.

Создать класс *Raboch* (Рабочие), у которого имеются 2 атрибута *name* и *vozrast* (имя и возраст) и методы *__init__*(), *display_count*(выводит количество рабочих) и *display_raboch* (выводит данные о рабочих) и класс *Deti* (Дети), у которого есть: два атрибута *name* и *vozrast* (имя и возраст); методы *__init__*() и *school*(), возвращающий номер школы ребенка.

```
class Raboch(object):
    raboch_count=0
    def __init__(self,name,vozrast):
        self.name=name
        self.vozrast=vozrast
        Raboch.raboch_count=Raboch.raboch_count+1
    def display_count(self):
        print('Всего сотрудников ',Raboch.raboch_count)
    def display_raboch(self):
        print('Имя ',self.name,'Возраст ',self.vozrast)
rabochii1=Raboch('Вася',20)
rabochii2=Raboch('Маша',25)
rabochii1.display_raboch()
rabochii2.display_raboch()
rabochii1.display_count()
print('Всего сотрудников ',Raboch.raboch_count)
```

```
Имя  Вася Возраст  20
Имя  Маша Возраст  25
Всего сотрудников  2
Всего сотрудников  2
```

```
rabochii1.zarplata=22
print(rabochii1.__dict__)
rabochii1.zarplata=28
print(rabochii1.__dict__)
del rabochii1.zarplata
print(rabochii1.__dict__)
```

```
{'name': 'Вася', 'vozrast': 20, 'zarplata': 22}
{'name': 'Вася', 'vozrast': 20, 'zarplata': 28}
{'name': 'Вася', 'vozrast': 20}
```

```
class Deti(Raboch):
    deti_count=0
    def __init__(self,name,vozrast):
        self.name=name
        self.vozrast=vozrast
        Deti.deti_count=Deti.deti_count+1
    def school(self,number):
        print('Школа номер ',number)
rebenok1=Deti('Даша',4)
rebenok2=Deti('Миша',7)
rebenok3=Deti('Ксюша',10)
rebenok1.display_raboch()
rebenok2.display_raboch()
rebenok3.display_raboch()
rebenok1.school(245)
rebenok1.display_count()
print('Всего детей ',Deti.deti_count)
```

```
Имя  Даша Возраст  4
Имя  Миша Возраст  7
Имя  Ксюша Возраст  10
Школа номер  245
Всего сотрудников  2
Всего детей  3
```

Задачи.

1. Создайте класс *Point* (точка), у которого имеются 2 атрибута x и y (координаты) и методы *__init__*() и *__str__*(), и класс *Rect* (прямоугольник), у которого есть: • два атрибута (верхний левый угол и правый нижний угол прямоугольника). Значениями атрибутов являются объекты класса *Point*; • методы *__init__*() и *__str__*(); метод *sides*(), возвращающий длины сторон прямоугольника; метод *perim*(), вычисляющий периметр прямоугольника. Продемонстрируйте работу с классами, создав необходимые объекты и вызвав все их методы.

2. Создайте класс *Point* (точка), у которого имеются 2 атрибута *x* и *y* (координаты) и методы *__init__*() и *__str__*(), и класс *Treyg* (треугольник), у которого есть: • три атрибута (верхушки треугольника). Значениями атрибутов являются объекты класса *Point*; • методы *__init__*() и *__str__*(); • метод *sides*(), возвращающий длины сторон треугольника; • метод *perim*(), вычисляющий периметр треугольника. Продемонстрируйте работу с классами, создав необходимые объекты и вызвав все их методы.

3. Создайте класс Ведомость, имеющий атрибут класса: список_дисциплин (значением является список с названиями дисциплин); дисциплина (при задании значения проверять наличие дисциплины в атрибуте список_дисциплин), группа; методы: *put* – добавляет в ведомость информацию об оценке студента (фамилия, оценка – параметры метода). Для хранения данных внутри класса используйте словарь, в котором ключом является фамилия студента. Возможные оценки – «отлично», «хорошо», «удовл.», «неудовл.», «н/я»; *get* – возвращает оценку, полученную студентом (фамилия студента – параметр метода); *change* – изменяет оценку, полученную студентом (фамилия студента и новая оценка – параметры метода); *del* – удаляет информацию о студенте из ведомости (фамилия студента – параметр

метода); *result* – возвращает кортеж из 5 чисел (количество каждого вида оценок в ведомости); *__init__* – конструктор; *__str__* – возвращает строку, содержащую заголовки (название экзамена, группа) и результаты экзамена в виде таблицы; *count* – возвращает количество студентов в ведомости; *names* – возвращает список фамилий, имеющихся в ведомости. Продемонстрируйте работу с классами, создав необходимые объекты и вызвав все их методы.

4. Используя класс *People* в качестве базового, создайте класс Сотрудник (*Worker*), имеющий атрибуты: должность (*post*); зарплата (*salary*) и методы: *__init__* – конструктор; *__str__* – для вывода строковой информации. Создать два метода для класса Сотрудник и один метод для класса *People*. Продемонстрируйте работу с классами, создав необходимые объекты и вызвав все их методы.

5. Используя класс Сотрудник в качестве базового создайте класс Преподаватель (*Teacher*), имеющий: атрибут дисциплины (*disciplines*), в котором хранятся названия дисциплин, которые ведет преподаватель; методы *__init__* и *__str__*;методы добавить_дисциплину (*add_dis*) и удалить_дисциплину (*delete_dis*), которые позволяют изменять список дисциплин. Продемонстрируйте работу с классами, создав необходимые объекты и вызвав все их методы.

6. Создать базовый класс по следующей предметной области. Известны оклад (зарплата) и ставка процента подоходного налога. Определить размер подоходного налога и сумму, получаемую на руки. Исходными данными являются величина оклада (переменная *oklad*, выражаемая числом) и ставка подоходного налога (переменная *procent*, выражаемая числом). Размер налога (переменная *nalog*) определяется как *oklad∗procent*/100, а сумма, получаемая на руки (переменная *summa*) — как *oklad-nalog*

7. Построить базовый класс с указанными в таблице полями и методами: конструктор; функция, которая определяет «качество» объекта – *Q* по заданной формуле; метод вывода информации об объекте. Построить дочерний класс (класс-потомок), который содержит: дополнительное поле P; функцию, которая определяет «качество» объекта дочернего класса – *Qp* и перегружает функцию качества родительского класса (*Q*), выполняя вычисление по новой формуле. Создать проект для демонстрации работы: ввод и вывод информации об объектах классов.

Поля и методы базового класса	Поля и методы дочернего класса
Автомобиль: - марка автомобиля; - мощность двигателя (кВт); - число мест; - *Q* = 0,1*мощность *число мест	P: год выпуска - *Qp* = *Q* - 1,5·(*T*-*P*), где *T* – текущий год.

8. Создать класс Профиль местности, который хранит последовательность высот, вычисленных через равные промежутки по горизонтали. Методы: наибольшая высота, наименьшая высота, перепад высот (наибольший, суммарный), крутизна (тангенс угла наклона; наибольшая, средняя), сравнение двух профилей одинаковой длины (по перепаду, по крутизне).

9. Задание: построить базовый класс с указанными в таблице полями и методами: конструктор; функция, которая определяет «качество» объекта – *Q* по заданной формуле; метод вывода информации об объекте. Построить дочерний класс (класс-потомок), который содержит: дополнительное поле *P*; функцию, которая определяет «качество» объекта дочернего класса – *Qp* и перегружает функцию качества родительского класса (*Q*), выполняя вычисление по новой формуле. Создать проект для демонстрации работы: ввод и вывод информации об объектах классов.

Поля и методы базового класса	Поля и методы дочернего класса
Компьютер: - наименование процессора; - тактовая частота процессора (МГц); - объем оперативной памяти (Мб); - *Q* = (0,1*частота) + память	*P*: объем накопителя SSD (Гб) *Qp* = *Q* +0,5**P*

10. Создать класс Интервалы, который хранит левую и правую границы интервала. Методы: длина интервала, смещение интервала (влево, вправо), сжатие (растяжение) интервала на заданный коэффициент, сравнение двух интервалов, сумма, разность двух интервалов. Рассматривать конечные вещественные интервалы [a, b]. Сумму и разность интервалов определять следующим образом:

сложение: [a, b] + [c, d] = [a + c, b + d]

вычитание: [a, b] − [c, d] = [a − d, b − c]

11. Создайте класс Студент, имеющий: атрибут Имя – строка, содержащая фамилию; метод *__init__*. При создании объекта указывается имя, список Дисциплины пустой; атрибут Дисциплины – словарь сданных дисциплин. Ключом является название дисциплины, значением – оценка. Метод *put* добавляет новую дисциплину в атрибут Дисциплины. Параметрами метода являются название дисциплины и оценка; свойство Сдано возвращает список названий сданных дисциплин. Создайте экземпляр класса, продемонстрируйте работу с атрибутами, методами и свойствами.

12. Создайте класс Заказ(*Order*), у которого есть атрибуты код_товара(*code*), цена(*price*), количество(*count*) и методы *__init__* и *__str__*. Создайте 2 класса-потомка: Опт(*Opt*) и Розница(*Retail*). В этих классах создайте методы *__init__*, *__str__* и сумма_заказа(*summa*), позволяющий узнать стоимость заказа. Для опта стоимость единицы товара составляет 95% от цены, а при покупке более 500 штук – 90% цены. В розницу

стоимость единицы товара составляет 100% цены. Стоимость заказа равна произведению цены на количество. Продемонстрируйте работу с классами, создав необходимые объекты и методам.

13. Создайте класс *Length* (Длина), имеющий атрибуты: *value* (значение), *unit* (единица измерения). При изменении единицы измерения значение должно соответственно меняться. Например, при переходе от сантиметров к метрам значение должно уменьшаться в 100 раз. Допустимые значения атрибуты *unit*: ‘см’, ‘м’, ‘км’. Организуйте эту проверку. Продемонстрируйте работу с классом.

Тесты по разделу 9.

1. Установите соответствие между понятиями объектно-ориентированного программирования и их содержанием:

Наименование	Содержание
Инкапсуляция	возможность расширения (наследования) ранее написанного программного кода класса с целью дополнения, усовершенствования или привязки под новые требования
Полиморфизм	размещение в одном компоненте данных и методов, которые с ними работают
Наследование	способность выполнять действие над объектом независимо от его типа на основе создания базового класса

2. Отметьте правильные утверждения принципов объектно-ориентированной парадигмы:

a) Данные структурируются в виде объектов, каждый из которых имеет определенный тип, то есть принадлежит к какому-либо классу

b) Классы – результат формализации решаемой задачи, выделения главных ее аспектов

c) Внутри объекта инкапсулируется логика работы с относящейся к нему информацией

d) Объекты в программе взаимодействуют друг с другом, обмениваются запросами и ответами

e) Объекты одного типа различным образом отвечают на одни и те же запросы

f) Объекты могут организовываться в более сложные структуры

g) Вычисление генератором следующего значения происходит при выполнении метода next(), при этом предыдущее значение сохраняется

3. Правильные утверждения относительно понятий объектно-ориентированного программирования:

a) Наследование позволяет создавать новый класс на основе уже существующего класса

b) Ключевыми понятиями наследования являются подкласс (производный класс, дочерний класс) и суперкласс (базовый класс, родительский класс)

c) Подкласс наследует от суперкласса все публичные атрибуты и методы

d) Полиморфизм предполагает способность к изменению функционала, унаследованного от базового класса

e) Инкапсуляция предотвращает прямой доступ к атрибутам объекта из вызывающего кода

f) Инкапсуляция обеспечивает прямой доступ к атрибутам объекта из вызывающего кода

4. *Как создать конструктор класса A?*

a) A(параметры конструктора)

b) def __init__(параметры конструктора)

c) def __A__(параметры конструктора)

d) def init(параметры конструктора)

5. *Как много конструкторов в классе может иметь Python:*

a) 0 (в них нет необходимости)

b) 1 (в Python можно создать лишь один конструктор)

c) 2 (для чисел и других объектов)

d) бесконечно много

6. *Результатом работы программы:*

```
class Slov(dict):
    pass
    def getDict(self,key, default=0):
        return dict.getDict(self,key,default)
b1=Slov(Иванов=5,Петров=4)
print(b1)
```

будет _____.

7. *Результатом работы программы:*

```
class Student:
    def setName(self,newName):
        self.name=newName
    def getName(self):
        return print(self.name)
s1=Student()
s2=Student()
s1.setName('Alex')
s1.getName()
Student.getName(s2)
```

будет …

Литература

1. Программирование на языке Python: Учебно-методическое пособие по дисциплине «Компьютерный практикум» для студентов, обучающихся по направлению подготовки 38.03.05 "Бизнес-информатика" (очная и заочная формы обучения). - М.: Финансовый университет, департамент анализа данных, принятия решений и финансовых технологий, 2018. -68 с.
2. Программирование на языке Python. Часть 2: Учебное пособие по дисциплинам «Компьютерный практикум» и «Алгоритмы и структуры данных в языке Python» для студентов, обучающихся по направлениям подготовки "Бизнесинформатика" и «Прикладная информатика» профиль «Высокопроизводительные вычисления в цифровой экономике». - М.: Финансовый университет, департамент анализа данных, принятия решений и финансовых технологий, 2019. -51 с.
3. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: СимволПлюс, 2009. – 608 с., ил.
4. Доусон М. Программируем на Python. - СПб.: Питер, 2014 -416 с.: ил.
5. Сузи Р.А. Язык программирования Python [Электронный ресурс]: курс / Р.А. Сузи. - 2-е изд., испр. - Москва: Интернет-Университет Информационных Технологий, 2007. - 327 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=233288.
6. Северенс Ч. Введение в программирование на Python [Электронный ресурс] / Ч. Северенс. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с. [Электронный ресурс]. - Режим доступа: http://biblioclub.ru/index.php?page=book&id=429184.

Учебное издание

Догадина Елена Петровна
кандидат технических наук, доцент Департамента анализа данных и машинного обучения факультета информационных технологий и анализа больших данных Финансового университета при Правительстве Российской Федерации

СБОРНИК ЗАДАЧ ПО ДИСЦИПЛИНЕ «КОМПЬЮТЕРНЫЙ ПРАКТИКУМ»
1, 2 семестр

Для студентов,
обучающихся по направлениям
10.03.01 «Информационная безопасность»,
38.03.05 «Бизнес-информатика»
(программа подготовки бакалавра)

Компьютерный набор, верстка: Е.П. Догадина

Вычитка и корректура выполнены авторами

Формат 60x90/16. Гарнитура *Times New Roman.*
Усл. п.л. 6,375. Изд. № - 2021. Заказ № ____
Электронное издание
