

Алгоритмы на Python



Содержание

№	ОРГАНИЗАЦИОННЫЕ ВОПРОСЫ	примеры	задания
A	Организация процесса учёбы (для очников)	ТУТ	
B	Какой редактор использовать	ТУТ	
C	Как добавить Python в пути	ТУТ	
D	Консольные приложения	ТУТ	
E	Шаблоны, примеры, функции...	ТУТ	
F	repl.it - как сложить все задания в одну папку	ТУТ	
G	Как построить график функции	ТУТ	
H	Примеры тестовых заданий		ТУТ
ПРАКТИКУМ			
1	Диалоги и ветвления		ТУТ
2	Циклические алгоритмы		ТУТ
3	Работа со строками		ТУТ
4	Работа с файлами		ТУТ
5	Алгоритмы анализа данных		ТУТ
6	Байки Солнечного города		ТУТ
7	Безумное чаепитие		ТУТ
8	Матрица		ТУТ
9	Черепашка	ТУТ	ТУТ
10	Объекты и сортировка		ТУТ
11	Область оптимальных решений		ТУТ
12	Парсер сайтов своими руками (requests) - YouTube	ТУТ	
13	Парсер на библиотеках (re, bs, lxml) - YouTube	ТУТ	



Организация процесса учёбы

Что важно

Основной упор будет сосредоточен на развитии **алгоритмического мышления** - как перевести из двоичной в десятичную систему счисления, как найти экстремум функции, как выбрать оптимальное решение, как найти путь в лабиринте, как отсортировать массив и другие классические алгоритмы - поэтому и название такое у этого учебника.

Как сдавать свои решения

Нужно сдавать лабораторные задания **своевременно**. На самой лабораторке я помогаю вам выполнять задания по текущей теме, если вам это необходимо, и вы обращаетесь за помощью. Можно пользоваться помощью не только преподавателя, но и интернета или товарищей в аудитории. Задачи можно решать совместно, но сдаёт каждый свои решения индивидуально.

Заданий не будет так много, чтобы вы не успели их завершить к концу лабораторки. В последние 20 минут занятия я обхожу аудиторию и опрашиваю вас по выполненным программам, а также, при необходимости (на моё усмотрение), даю маленькое задание для непосредственного выполнения в моём присутствии, чтобы вы показали, что можете сделать сами.

Задания можно выполнять заблаговременно (дома, в общежитии с друзьями), а на лабораторку приходить и сразу сдавать или уточнять у меня сложные вопросы, доделывать и потом сдавать.

Если вы отчитываетесь за лабораторку **несвоевременно**, то оценка будет **снижена на один балл**. Допускается сдавать задания на следующей лабораторке (например, вторую на третьей), если я не успел вас опросить на соответствующем занятии.

Дополнительные возможности реализации

Можно получить **дополнительные баллы**, значительно повышающие вашу итоговую оценку, если вы активно принимаете участие в некоторых мероприятиях: разрабатываете проект и выступаете с ним **на ИТ-Хакатоне** или участвуете в **команде по программированию** или получаете Сертификат в онлайн школах (SoloLearn, Stepik).



Какой редактор (IDE) использовать

Попробуйте несколько, посмотрите YouTube и сделайте выбор сами. В аудиториях пока установлен только Visual Studio Code.

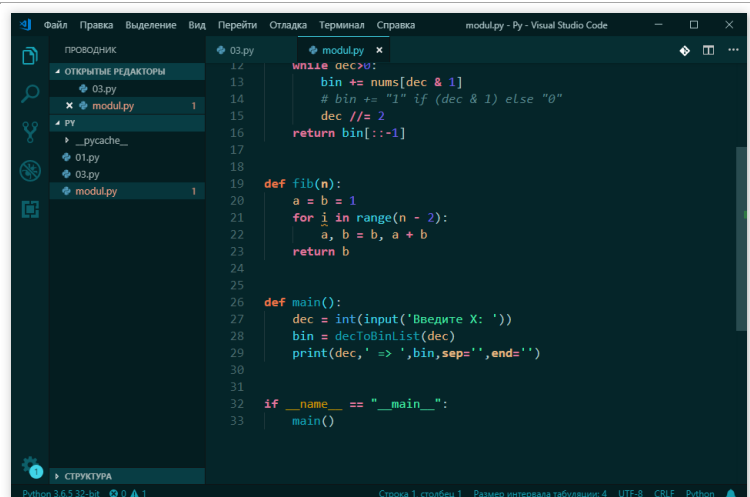
На том уровне сложности проектов, на котором будем работать мы, в принципе, хватило бы и Notepad++, однако, интегрированные среды разработки, при правильной настройке и некотором опыте работы с ними, дадут вам больший комфорт и удовольствие от работы, облегчат вам набор кода, уменьшат количество некорректных участков кода, помогут при поиске ошибок, ускорят вашу работу. В перечисленный ниже список я не включил, разве что, PyCharm – эта кроссплатформенная IDE создавалась специально под Python, включает большой функционал, в том числе и инструменты для веб-разработки с использованием фреймворка Django, и поддержку систем контроля версий Git и других. У PyCharm есть несколько вариантов лицензий, отличающихся функционалом, включая бесплатные варианты. На мой, субъективный взгляд, на первом курсе обучения можно ограничиться и менее навороченными возможностями, не отвлекаясь на пока незначимые детали. Если вы действительно займетесь разработкой программного обеспечения по обработке big data, или построением сайтов с использованием Django, или обработкой статистических данных, или анализом научной информации, то вы сможете сделать уже осознанный выбор необходимой именно для ваших требований среды разработки.

А пока вот вам небольшой список IDE для анализа и выбора:

Visual Studio Code

<https://code.visualstudio.com/>

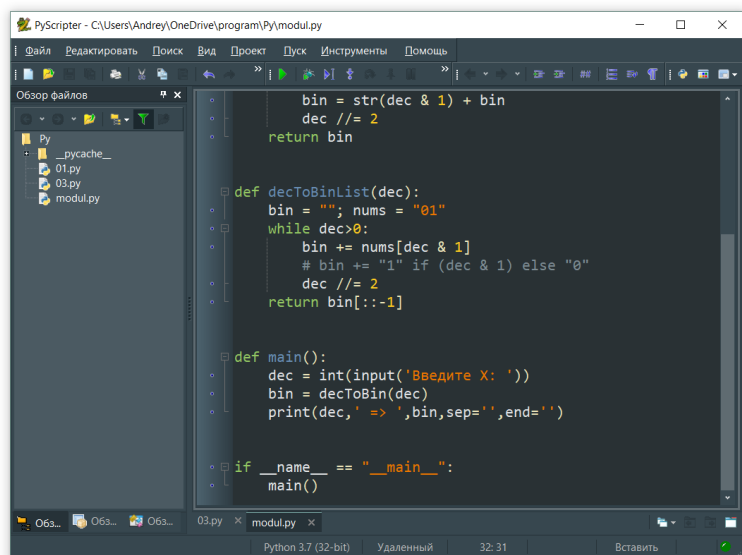
Есть всё, что необходимо: кроссплатформенность, подсветка синтаксиса, автодополнение кода, разные стили, отладка кода, запуск программы во внутреннем терминале, поддержка множества разных языков программирования, поддержка системы управления версиями Git, русификация, бесплатный, гибкие настройки под себя, включая и горячие клавиши, автоматические обновления.



PyScripter Python IDE

<https://sourceforge.net/projects/pyscripter/>

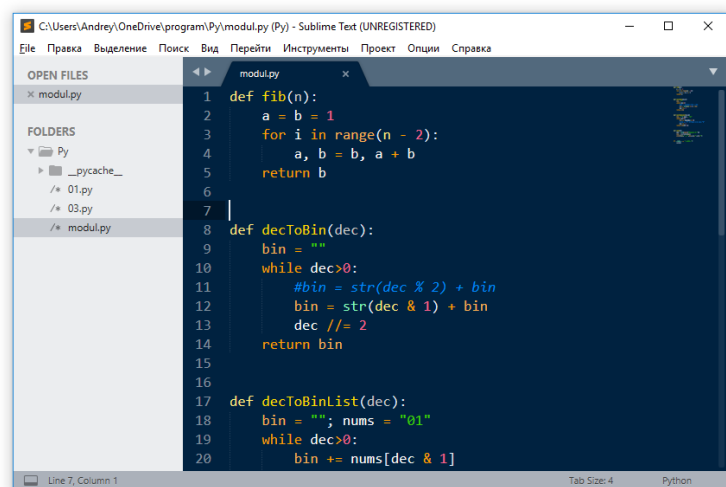
Бесплатный. Очень «лёгкий». Русифицирован. Подсказывает при написании кода. Очень много стилей оформления. Отдельные стили для оболочки программы и окна редактора. Программа запускается прямо из редактора. Есть функции для отладки кода программы. Можно управлять версиями Python, установленными на ваш компьютер.



Sublime

<https://www.sublimetext.com/>

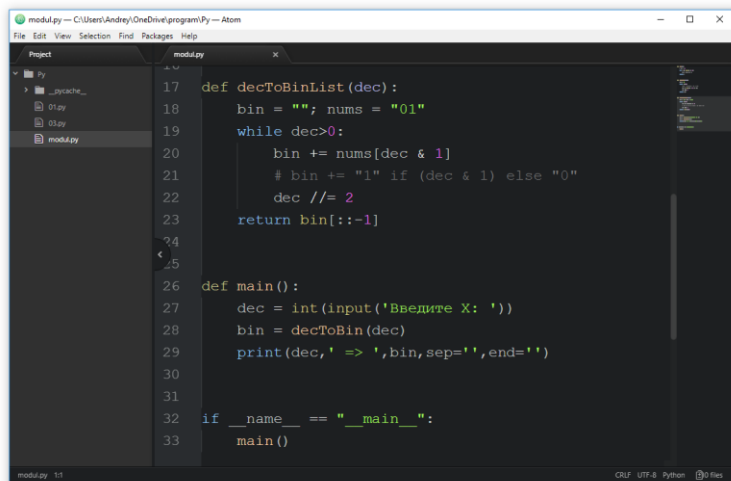
Это текстовый редактор с подсветкой синтаксиса. Условно бесплатный - нужна регистрация. Кроссплатформенный. С поддержкой плагинов - дополнительные модули для расширения функционала и сниппетов - фрагментов кода для повторного использования. Поддерживает несколько языков программирования.



Atom

<https://atom.io/>

Это текстовый редактор с подсветкой синтаксиса. Бесплатный. Кроссплатформенный. С поддержкой плагинов - дополнительные модули для расширения функционала. Поддерживает несколько языков программирования.

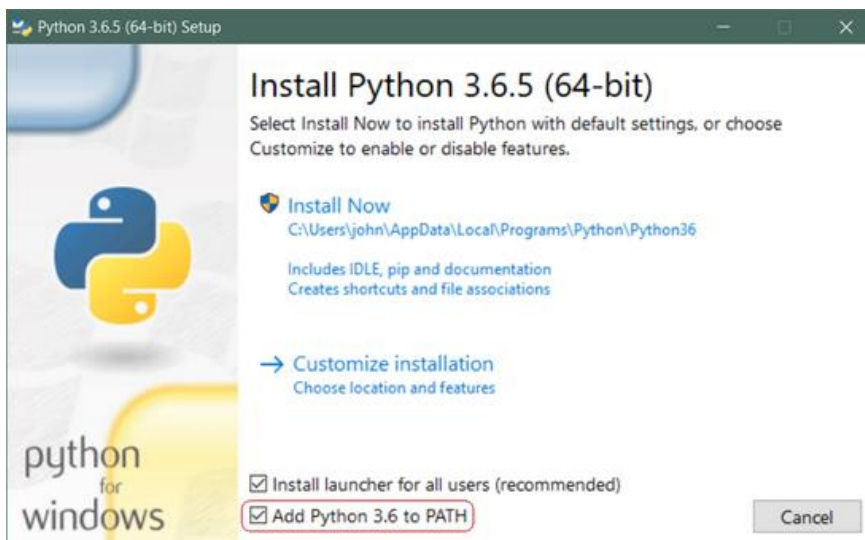


ВНИМАНИЕ!

Материал данного учебника не является исчерпывающим, а рассчитан на использование после прослушивания соответствующих лекций и основан на знаниях, полученных в предыдущем семестре. Всё, что непонятно, можно узнать у преподавателя непосредственно на занятии или самостоятельно в сети.

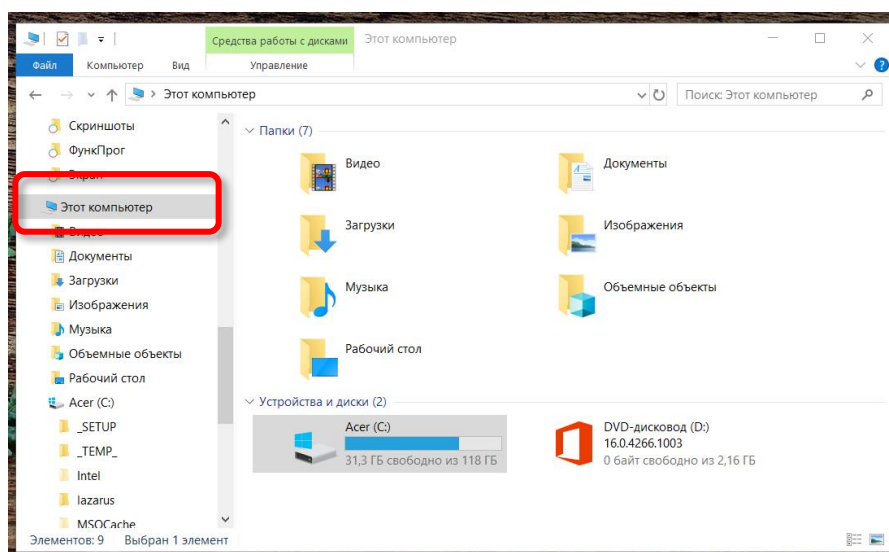


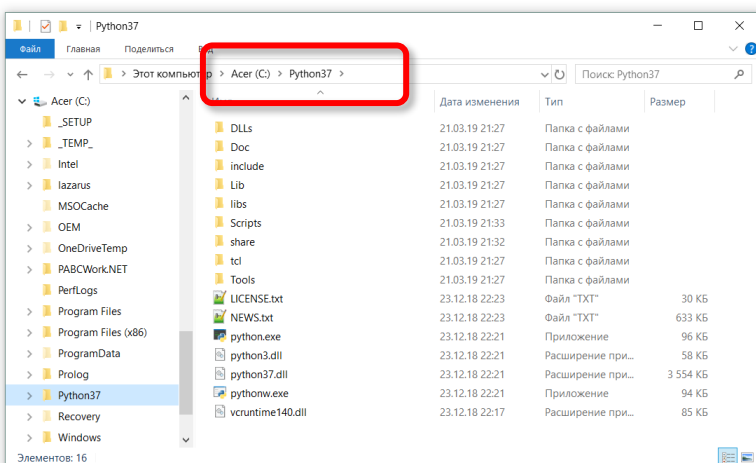
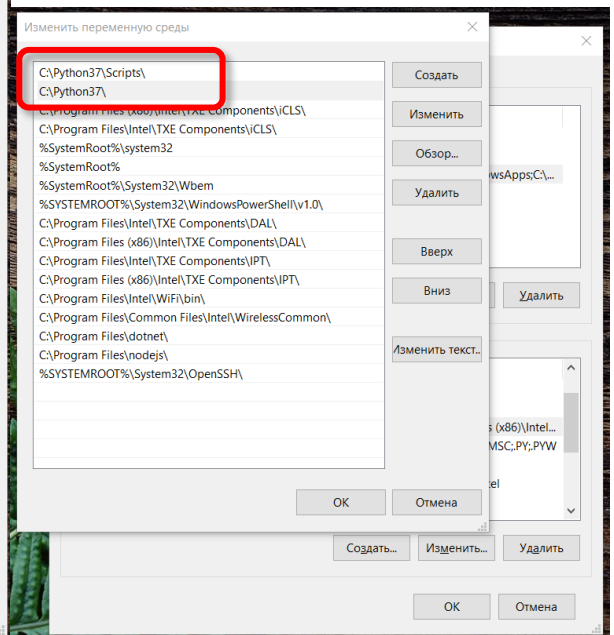
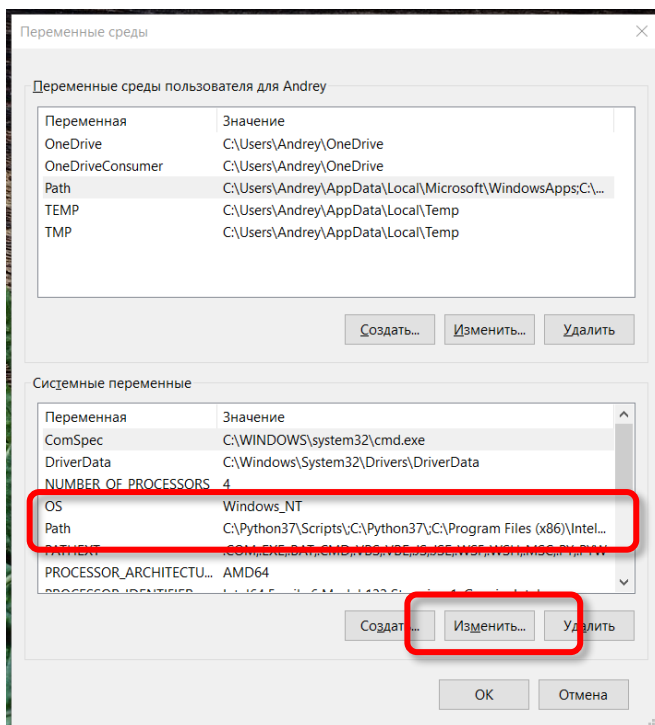
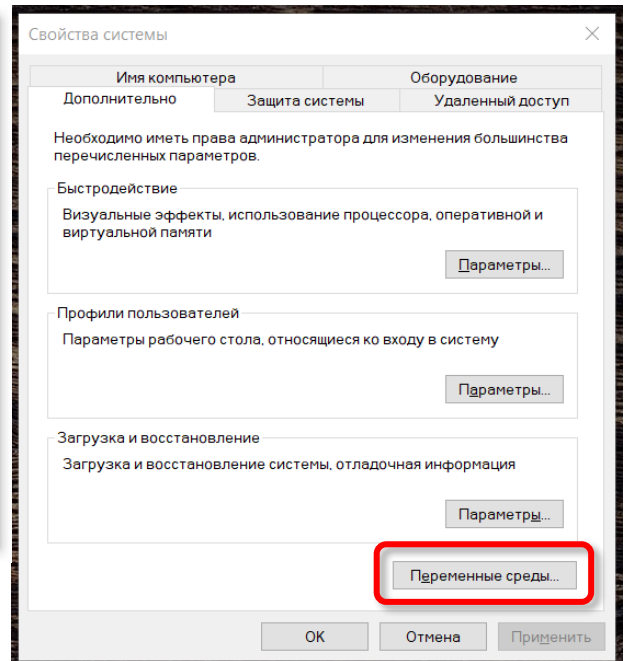
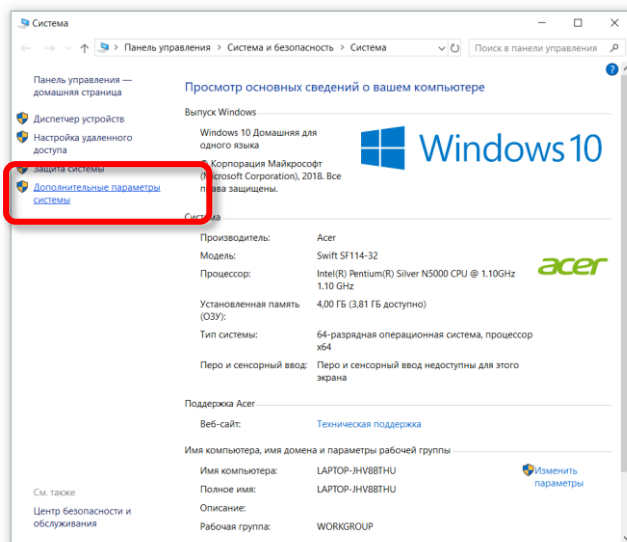
Add Python to PATH



Если, при установке Python, вы забыли поставить галочку «Добавить Python в пути», то это можно сделать самостоятельно потом.

Найдите в проводнике «Этот компьютер», кликните правой клавишей мыши и выберите опцию «Свойства», далее «Дополнительные параметры среды», далее «Переменные среды», там найдите строку Path в любом из окон (например, «Системные переменные») и выделите её мышкой, далее нажмите «Изменить» - в открывшемся окне нажмите клавишу «Создать» (или просто дважды кликните по свободной строке в списке путей) и туда впишите (или скопируйте путь к вашему Пайтону). Желательно прописать два пути - посмотрите, как это сделано у меня на предпоследнем скриншоте.







Шаблоны программ, примеры использования методов

1. Ввод строки с клавиатуры и сохранение её в переменной, вывод целого числа:

```
s = input()
s = input('Введите строку - ')

s = input('Введите целое число - ')
n = int(s)
n = int(input())
```

2. Ввод и форматирование вещественных чисел:

```
f = input('Введите вещественное число - ')
n = float(f)
print("n =", n)
print("n = {0}".format(n))
import math as m
Pi = m.pi
print("Pi = {0}".format(Pi))
print("Pi = {0}".format(round(Pi, 3)))
print("2*Pi = {0:.3f}".format(2*Pi))
```

Вывод:

```
n = 3.14
n = 3.14
Pi = 3.141592653589793
Pi = 3.142
2*Pi = 6.283
```

3. Есть строка, состоящая из цифр. Нужно вывести цифры этой строки через пробел. Ниже несколько способов реализации этой задачи:

```
s = "1234567890"

lst = list(s)
for elm in lst:
    print(elm, '', end='')
print()

line = ""
for elm in lst:
    line += str(elm) + " "
print(line)

line = ""
for elm in lst:
    line += "{0} ".format(elm)
```

```

print(line)

print(*lst)

line = ' '.join(lst)
print(line)

for smb in s:
    print(smb + " ", end='')

```

Вот такой будет вывод, все результаты идентичны, но способы реализации разные:

```

1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0

```

4. Рекурсивная функция перевода двоичного числа в десятичное.

Ввод как из командной строки, так и из консоли

```

import os
import sys

def binToDec(n):
    if not n: return 0
    if n[0] not in '01': raise ValueError
    return (2 ** (len(n)-1)) * (n[0] == '1') + binToDec(n[1:])

os.system('cls')

bin = sys.argv[1] if sys.argv[1:] else input('Введите двоичное число: ')
print('{0}(2) => {1}(10)'.format(bin, binToDec(bin)))

os.system('pause')

```

Варианты вывода:

Введите двоичное число: 1011

1011(2) => 11(10)

Для продолжения нажмите любую клавишу . . .

или так:

```

Traceback (most recent call last):
  File "binToDec.py", line 13, in <module>
    print('{0}(2) => {1}(10)'.format(bin, binToDec(bin)))
  File "binToDec.py", line 6, in binToDec
    if n[0] not in '01': raise ValueError
ValueError

```


5. Класс, конструктор и метод класса

```
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def getName(self):
        return self.name.split(' ')[1]

men = Person("Иванов Иван Иванович", 19)

print(men.age, '\t', men.name)
print(men.getName())
```

6. Из текстового файла в список объектов

```
file = open('рейтинг.txt', 'r')
lines = file.read().split('\n')
file.close()

class Person(object):
    def __init__(self, name, ball):
        self.name = name
        self.ball = ball

arr = []
for line in lines:
    name = line.split('\t')[1]
    ball = int(line.split('\t')[0])
    men = Person(name, ball)
    arr.append(men)
```

файл "рейтинг.txt"

22	Иж-Комби
-3	Унукулеле
77	Абракадабра
909	Гумус
666	Антананариву
0	Адольф
1	Ева
2	Петров
3	Васечкин

7. Сортировка объектов

так:

```
from operator import attrgetter

arr.sort(key = attrgetter('age'))

for men in arr:
    print(men.age, '\t', men.name)
```

или так:

```
def getAttr(men):
    return men.age

arr.sort(key = getAttr) # можно так arr.sort(key = getAttr, reverse=True)

for men in arr:
    print(men.age, '\t', men.name)
```

8. Генерация списка случайных чисел и способы вывода

```
from random import *

count = 10
num_min = 10
num_max = 99
lst = [randint(num_min, num_max) for i in range(count)]

for i, elm in enumerate(lst):
    print("{0}\t{1}".format(i+1, elm))

for i in range(len(lst)):
    print("{0}\t{1}".format(i+1, lst[i]))

i = 0
for elm in reversed(lst):
    print("{0}\t{1}".format(i+1, elm))
    i += 1
```

Функции библиотеки math

Для использования функций из библиотеки `math` необходимо подключить модуль командой `import math`. Например, пусть мы хотим округлять вещественные числа до ближайшего целого числа *вверх*. Соответствующая функция `ceil` от одного аргумента вызывается, например, так: `math.ceil(x)` (то есть явно указывается, что из модуля `math` используется функция `ceil`). Вместо числа `x` может быть любое число, переменная или выражение. Функция возвращает значение, которое можно вывести на экран, присвоить другой переменной или использовать в выражении:

```
import math
x = math.ceil(4.2)
y = math.ceil(4.8)
print(x, y)
```

Другой способ использовать функции из библиотеки `math`, при котором не нужно будет каждый раз указывать название модуля, выглядит так:

```
from math import ceil

x = 7 / 2
y = ceil(x)
print(y)
```

или так:

```
from math import *

x = 7 / 2
y = ceil(x)
print(y)
```

Некоторые из перечисленных функций (`int`, `round`, `abs`) являются стандартными и не требуют подключения модуля `math` для использования.

Функция	Описание
Округление	
<code>int(x)</code>	Округляет число в сторону нуля. Это стандартная функция, для ее использования не нужно подключать модуль <code>math</code> .
<code>round(x)</code>	Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
<code>round(x, n)</code>	Округляет число <code>x</code> до <code>n</code> знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль <code>math</code> .
<code>floor(x)</code>	Округляет число вниз («пол»), при этом <code>floor(1.5) == 1</code> , <code>floor(-1.5) == -2</code>
<code>ceil(x)</code>	Округляет число вверх («потолок»), при этом <code>ceil(1.5) == 2</code> , <code>ceil(-1.5) == -1</code>
<code>abs(x)</code>	Модуль (абсолютная величина). Это – стандартная функция.
Корни, логарифмы	
<code>sqrt(x)</code>	Квадратный корень. Использование: <code>sqrt(x)</code>
<code>log(x)</code>	Натуральный логарифм. При вызове в виде <code>log(x, b)</code> возвращает логарифм по основанию <code>b</code> .
<code>e</code>	Основание натуральных логарифмов <code>e = 2,71828...</code>

Тригонометрия	
sin(x)	Синус угла, задаваемого в радианах
cos(x)	Косинус угла, задаваемого в радианах
tan(x)	Тангенс угла, задаваемого в радианах
asin(x)	Арксинус, возвращает значение в радианах
acos(x)	Арккосинус, возвращает значение в радианах
atan(x)	Арктангенс, возвращает значение в радианах
atan2(y, x)	Полярный угол (в радианах) точки с координатами (x, y).
degrees(x)	Преобразует угол, заданный в радианах, в градусы.
radians(x)	Преобразует угол, заданный в градусах, в радианы.
pi	Константа $\pi = 3.1415\dots$



Как построить график функции

Про библиотеку tkinter можно почитать [тут](#).

Пример_1. Оси координат, кружок и текст.

```
import tkinter as tk
import random

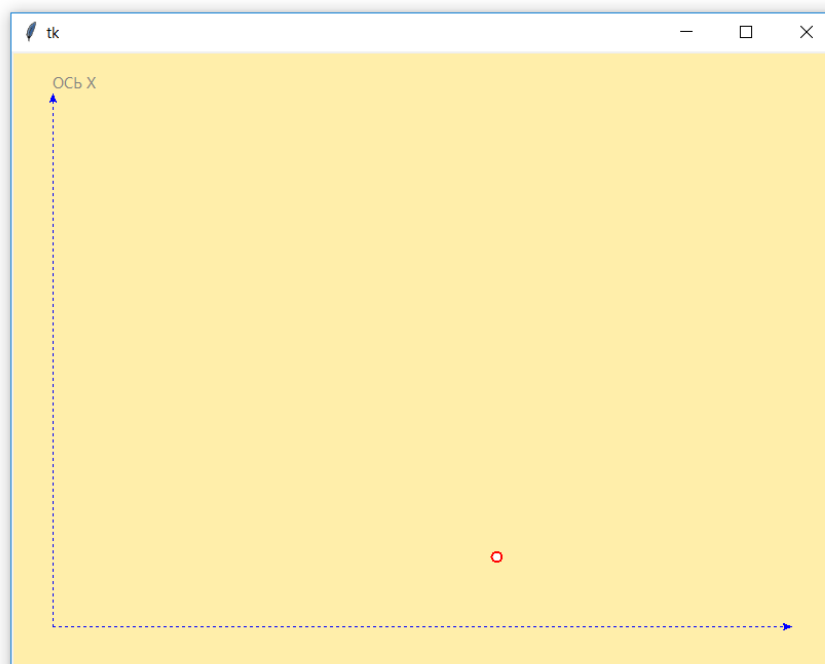
def axes(x0, y0, xm, ym, color):
    canvas.create_line(x0, y0, xm, y0, fill=color, arrow=tk.LAST, dash=(4, 2))
    canvas.create_line(x0, y0, x0, ym, fill=color, arrow=tk.LAST, dash=(4, 2))

w = 800
h = 600
pole = 40
x0 = pole
y0 = h-pole
xm = w-pole
ym = pole

window = tk.Tk()
canvas = tk.Canvas(window, width=w, height=h, bg='#fda')
canvas.pack()

r = 3
axes(x0, y0, xm, ym, 'blue')
x = random.randint(0, 800 - 2*pole)
y = random.randint(0, 600 - 2*pole)
canvas.create_oval(x-r, y+r, x+r, y-r, outline="red", fill="white", width=1)
title = 'Ось X'
canvas.create_text(pole, pole, text=title, anchor=tk.SW, fill="grey")

window.mainloop()
```



Задания по первому примеру.

1. Заполнить поле графика кружками со случайными координатами.

Реализация:

```
import tkinter as tk
import random as rnd

def axes(x0, y0, xm, ym, color):
    canvas.create_line(x0, y0, xm, y0, fill=color, arrow=tk.LAST, dash=(4, 2))
    canvas.create_line(x0, y0, x0, ym, fill=color, arrow=tk.LAST, dash=(4, 2))

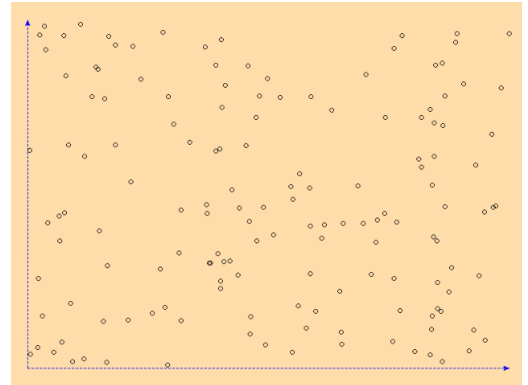
w = 800
h = 600
pole = 40
r = 3
n = 250
x0 = pole
y0 = h-pole
xm = w-pole
ym = pole

window = tk.Tk()
canvas = tk.Canvas(window, width=w, height=h, bg='#fda')
canvas.pack()

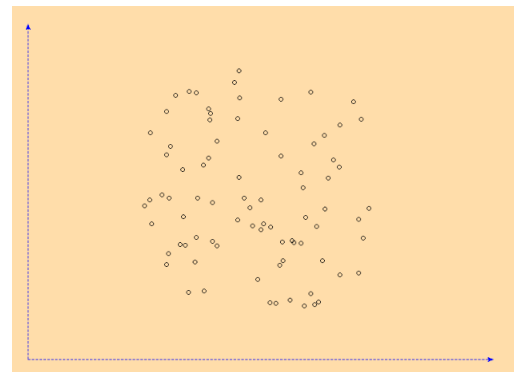
axes(x0, y0, xm, ym, 'blue')

for i in range(n):
    x = rnd.randint(pole, w-pole)
    y = rnd.randint(pole, h-pole)
    canvas.create_oval(x-r, y-r, x+r, y+r)

window.mainloop()
```



2. Заполнить поле графика кружками, расположенными в большом круге, центром которого является центр плоскости построения. Пусть rr - это размер радиуса большого круга в пикселях, в котором разрешены отображения пикселей. Такого рода задачи встречаются в практике, в том числе для некоторых сайтов, когда нужно отобразить дома в городе, которые попадают в определённый радиус.



Дописать эту реализацию:

```
import tkinter as tk
import random as rnd

def axes(x0, y0, xm, ym, color):
    canvas.create_line(x0, y0, xm, y0, fill=color, arrow=tk.LAST, dash=(4, 2))
    canvas.create_line(x0, y0, x0, ym, fill=color, arrow=tk.LAST, dash=(4, 2))
```

```

def check(x,y,rr):
    return True

w = 800
h = 600
pole = 40
r = 3
n = 250
rr = 200
x0 = pole
y0 = h-pole
xm = w-pole
ym = pole
xc = w//2
yc = h//2

window = tk.Tk()
canvas = tk.Canvas(window, width=w, height=h, bg='#fda')
canvas.pack()

axes(x0, y0, xm, ym, 'blue')

for i in range(n):
    x = rnd.randint(pole,w-pole)
    y = rnd.randint(pole,h-pole)
    if check(x,y,rr):
        canvas.create_oval(x-r,y-r,x+r,y+r)

window.mainloop()

```

3. Заполнить поле графика координатами, взятыми из текстового файла.

Данные в текстовый файл запишите самостоятельно в таком формате – в каждой строке файла координаты одной точки X и Y, записанные через табулятор, например:

```

12      3
12      4
222     100
333     300
600     600

```

3.А. Пусть X и Y – целые положительные числа, где X – лежит в диапазоне от 0 до $w-2*pole$, а $Y \in [0, h-2*pole]$.

3.Б. А как поступить, если диапазон X и Y выходит за рамки w и h (скажем, координаты точек будут лежать в диапазоне более 1000), или, наоборот, X и Y достаточно малые величины (скажем, это числа до 30)?

Пример_2:

```
import math
import tkinter as tk

def axes(x0, y0, xm, ym, fil_color):
    canvas.create_line(x0-10, y0, xm+10, y0, fill=fil_color, arrow=tk.LAST)
    canvas.create_line(x0, ym-10, x0, 2*y0-ym+10, fill=fil_color, arrow=tk.BOTH)

def graph(f, fm, tm, x0, y0, xm, ym, fil_color):
    xb = x0; t = 0; yb = y0+(ym-y0)*f(t)/fm
    canvas.create_line(xb, yb, xb, yb, fill=fil_color)
    for x in range(x0+2, xm, 2):
        t = tm*(x-x0)/(xm-x0)
        y = y0+(ym-y0)*f(t)/fm
        canvas.create_line(xb, yb, x, y, fill=fil_color)
        xb = x; yb = y

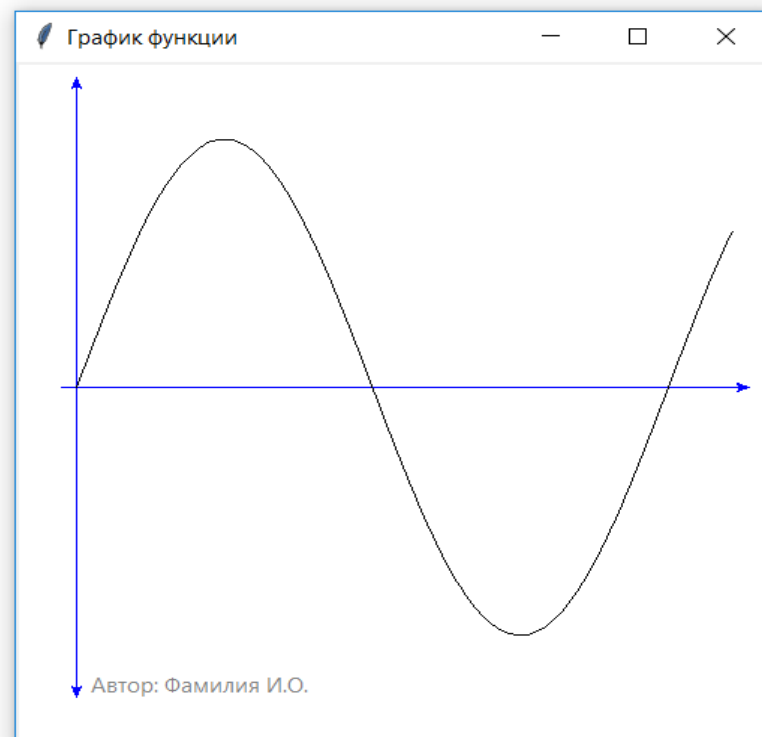
window = tk.Tk()
window.title("График функции")
canvas = tk.Canvas(window, width=500, height=500, bg='white', cursor="pencil")
canvas.pack()

x0 = 40; y0 = 240; xm = 480; ym = 20

axes(x0, y0, xm, ym, 'blue')
graph(math.sin, 1.2, 7, x0, y0, xm, ym, 'black')

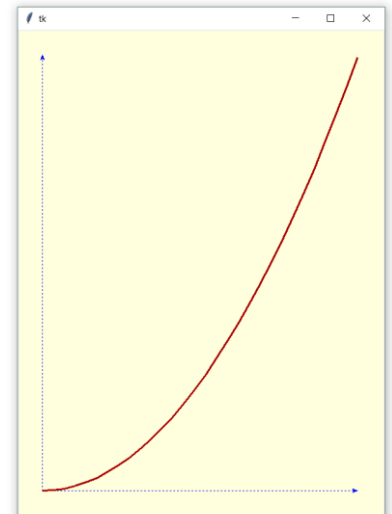
canvas.create_text(50, 450, text="Автор: ФИО", anchor=tk.NW, fill="grey")

window.mainloop()
```



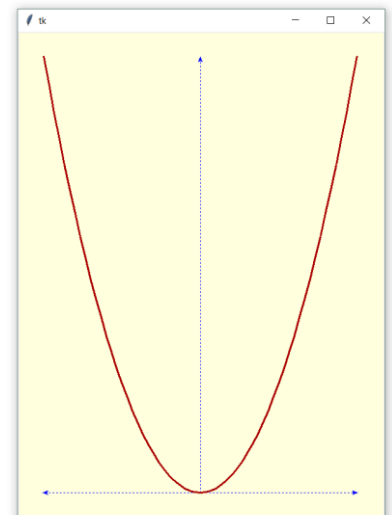
Пример_3. Построение графика функции $y = x^2$.

В этом примере сначала определяются размеры экрана вывода и поля от краёв экрана до графика, затем строятся оси координат, потом вычисляются точки графика и максимальные значения по осям X и Y , максимальные значения используются далее для вычисления масштаба построения графика (чтобы привести его к размерам экрана), полученные точки в виде списка объектов отправляются в функцию построения графика (строится в виде отдельных отрезков между точками).



В программе есть недостатки и недоработки.

Программа рассчитана только на правый верхний квадрант плоскости координат. **Попробуйте самостоятельно** внести изменения в программу, чтобы построить график функции $y=x^2$ для области определения функции $x \in [-3; +3]$.



```
import tkinter as tk

class Point(object): # декларируем класс для хранения координат точек
    def __init__(self, x, y):
        self.x = float(x)
        self.y = y

def my_func(x):
    return x**2 # тут пишем свою функцию

def getPoints(func, x_left, x_right, step):
    points = []
    max_x = 0; max_y = 0
    x = x_left
    while x <= x_right:
        y = func(x) # вычисляем значение функции
        points.append(Point(x,y))
        if x > max_x:
            max_x = x
        if y > max_y:
            max_y = y
        x += step
    return [points, max_x, max_y] # возвращаем точки и максимумы по осям
```

```

# рисуем оси
def axes(x0, y0, xm, ym, color):
    canvas.create_line(x0, y0, xm, y0, fill=color, arrow=tk.LAST, dash=(4, 2))
    canvas.create_line(x0, y0, x0, ym, fill=color, arrow=tk.LAST, dash=(4, 2))

# рисуем линии графика с учётом масштаба
def graph(points, mx, my, color):
    for i in range(len(points) - 1):
        x0 = points[i].x * mx; y0 = points[i].y * my
        x1 = points[i+1].x * mx; y1 = points[i + 1].y * my
        canvas.create_line(x0 + pole, h - pole - y0, x1 + pole, h - pole - y1,
            fill=color, width=3)

window = tk.Tk()
w = 600; h = 800; pole = 40 # размеры окна и полей
canvas = tk.Canvas(window, width=w, height=h, bg='#ffd')
canvas.pack()

axes(pole, h-pole, w-pole, pole, 'blue') # рисуем оси

# область определения функции и шаг функции
x_left = 0; x_right = +3; step = 0.1
tmp = getPoints(my_func, x_left, x_right, step)
points = tmp[0] # получаем точки графика
mx = (w-2*pole) // tmp[1] # вычисляем масштаб по X
my = (h-2*pole) // tmp[2] # вычисляем масштаб по Y
graph(points, mx, my, '#a00') # рисуем линию графика функции

window.mainloop()

```

Как вы поняли библиотека Tkinter не имеет автоматизации для построения графиков функций, нам всё приходилось делать «вручную». Можно использовать и иные, специально подготовленные, библиотеки для построения графиков функций. Смотрите следующие примеры.

Пример_4.

График функции с использованием библиотеки matplotlib

Кроме библиотеки Tkinter существует несколько специализированных библиотек для работы с математикой и функциями. Приведу пример построения графика функции $y=x^2$ с использованием matplotlib.

Чтобы программа работала на вашем компьютере, сначала нужно установить библиотеку введя команду в консоли:

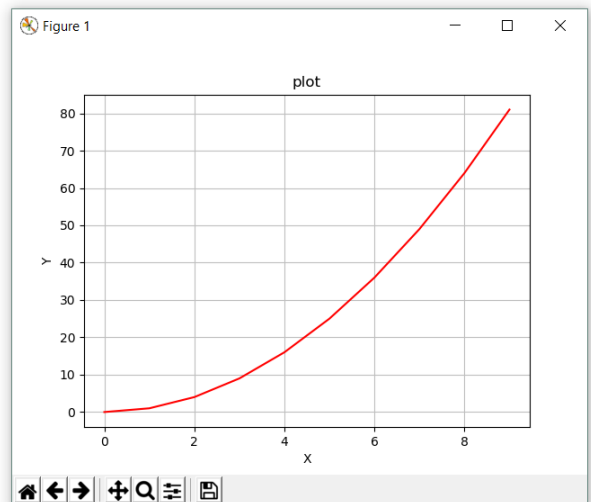
pip install matplotlib

- там много вспомогательных модулей, нужно будет подождать минуту, пока они все установятся.

Сама программа построена так:

- сначала формируются все значения по оси X;
- потом все значения по оси Y - с использованием заранее определённой функции;
- потом немного оформления;
- в самом конце все точки передаются для построения графика в специальную функцию.

Обратите внимание, что тут вы получаете несколько дополнительных возможностей, включая масштабирование мышкой и сохранение построенного графика в графический файл в самых распространённых форматах (нажмите на экранную клавишу сохранения).



```
import matplotlib.pyplot as plt

def my_func(x):
    return x**2

# Значения по X
data_x = [x for x in range(10)]
# Значения по Y
data_y = [my_func(x) for x in data_x]

plt.title("plot")
plt.xlabel("X")
plt.ylabel("Y")

plt.plot(data_x, data_y, 'r')

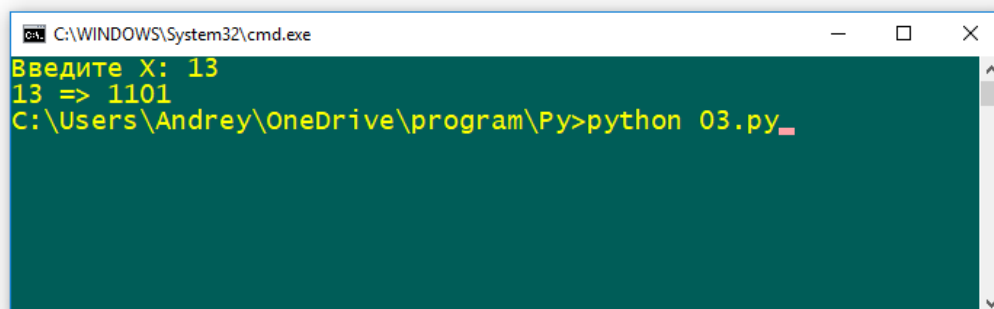
# Сетка на плоскости координат
plt.grid(True, linestyle='-', color='0.75')

plt.show()
```



Консольные приложения

Консольные приложения можно запустить из консоли, и они не имеют графического интерфейса пользователя – управлять вводом данных можно с использованием клавиатуры. Вот так могут выглядеть запуск программы в консоли, ввод данных вручную и результаты работы программы:



```
C:\WINDOWS\System32\cmd.exe
Введите X: 13
13 => 1101
C:\Users\Andrey\OneDrive\program\Py>python 03.py_
```

Консольную программу можно запустить как из внутреннего терминала редактора, так и из командной строки операционной системы.

Теперь перейдём собственно к программированию. Сначала оцените программу возведения числа в степень, параметры в которую пользователь вводит сам с клавиатуры:

```
x = int(input("Введите X = "))
y = int(input("Введите Y = "))
result = x**y

print('x^y =', result)
```

Как видите, в Python, стиль работы с данными очень похож на JavaScript, не правда ли:

```
x = Number(prompt("Введите X = "));
```

Теперь рассмотрим программу, в которой предусмотрены меры приёма параметров не в диалоге с пользователем (когда программа уже запущена), а через строку запуска программы из консоли (до работы программы, в момент её запуска):

```
import sys
import os

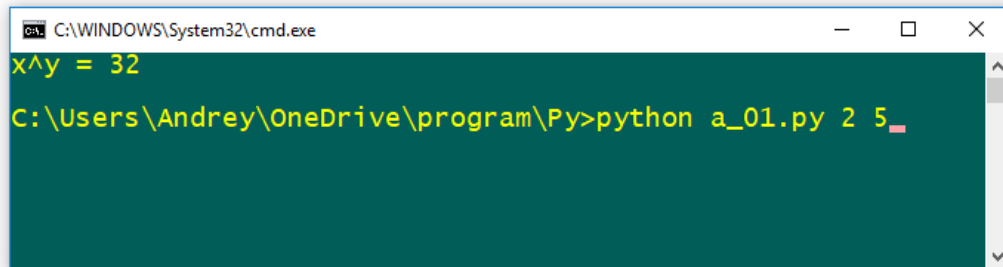
os.system("cls") # очистка экрана

arr = sys.argv # параметры из командной строки
x = int(arr[1])
y = int(arr[2])
```

```
result = x**y
```

```
print('x^y =', result)
```

Как видите, в начале программы мы подключаем необходимые модули, потом в тексте программы мы используем функции из этих модулей: для очистки экрана и для получения списка параметров, введённых в командной строке через пробел:

A screenshot of a Windows command prompt window titled "C:\WINDOWS\System32\cmd.exe". The window has a dark green background. The first line shows "x^y = 32" in yellow text. The second line shows the command "C:\Users\Andrey\OneDrive\program\Py>python a_01.py 2 5_" in yellow text, with the cursor at the end of the command.

Проведите эксперимент и узнайте, что же храниться в списке параметров под индексом 0.

Но что будет, если пользователь не введёт два параметра? Попробуйте сами, и вы увидите примерно такую реакцию:

IndexError: list index out of range

Произошла попытка обращения к несуществующему элементу списка, и программа «поломалась». Предпримем попытку предотвращения такого исхода:

```
import sys

if len(sys.argv) > 2:
    x = int(sys.argv[1])
    y = int(sys.argv[2])
    result = 'x^y = ' + str(x**y)
else:
    result = 'недостаточное кол-во параметров'

print(result)
```

Но такая попытка не работает при вводе некорректного значения, например, не числового - проведите эксперимент и посмотрите реакцию. Немного доработаем программу:

```
import sys

if len(sys.argv) > 2:
    try:
```

```

x = int(sys.argv[1])
y = int(sys.argv[2])
result = 'x^y = ' + str(x ** y)
print(result)
except:
    print('ошибка ввода данных')
else:
    print('недостаточное кол-во параметров')
И проведите её апробацию:

```

```

C:\WINDOWS\System32\cmd.exe

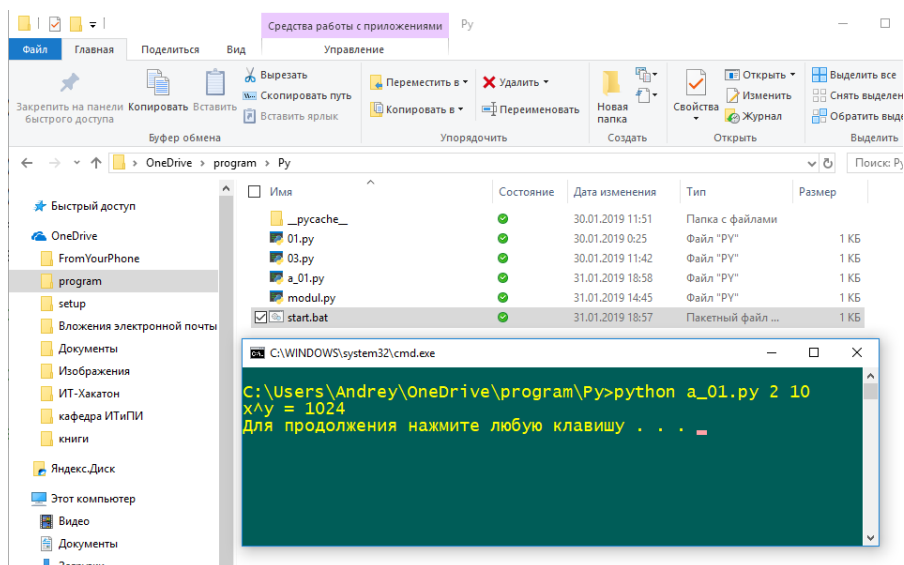
C:\Users\Andrey\OneDrive\program\Py>python a_01.py 2 5
x^y = 32

C:\Users\Andrey\OneDrive\program\Py>python a_01.py 2 5y
ошибка ввода данных

C:\Users\Andrey\OneDrive\program\Py>

```

Однако, каждый раз запускать консоль и там вводить вручную необходимые параметры, особенно, если их много, неудобно. Можно организовать специальный командный файл, так называемый батник, через который и запускать нашу программу прямо из проводника двойным кликом мышки:



Содержимое программы:

```

import sys
import os

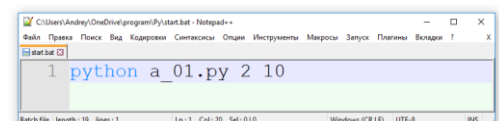
if len(sys.argv) > 2:
    try:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
        result = 'x^y = ' + str(x ** y)
        print(result)
    except:
        print('ошибка ввода данных')
else:
    print('недостаточное кол-во параметров')

os.system("pause")

```

Содержимое файла start.bat:

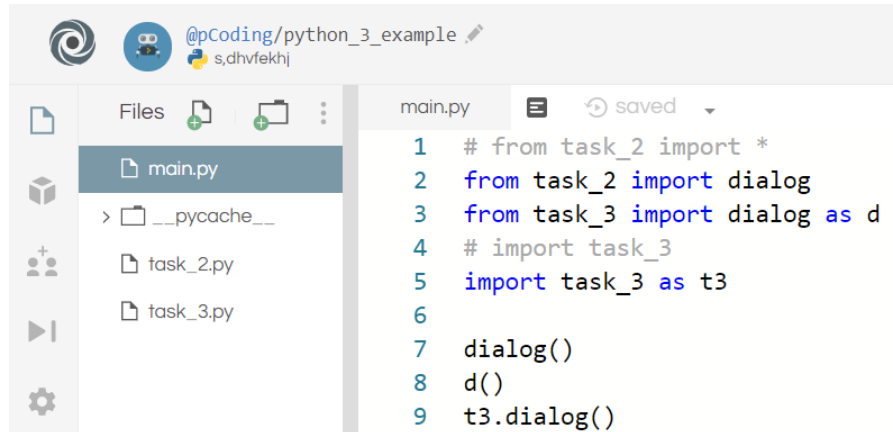
```
python a_01.py 2 10
```





Как сложить все задания в одну папку - repl.it

Если вы делаете Контрольную работу или Задания лабораторной работы на сервисе repl.it, то, возможно, вам хотелось бы разместить все файлы *.py в одной папке:



К сожалению, в сервисе repl.it не удаётся воспользоваться таким способами запуска приложений:

```
os.system('part_2.py')  
os.startfile('part_2.py')
```

Поэтому можно использовать следующий подход: вы создаёте для каждого задания свой файл *.py и пишете там соответствующий заданию код, размещая его в функциях, а из основного файла main.py вызываете эти функции. Для вызова функций из разных файлов можно воспользоваться несколькими способами импорта модулей (посмотрите начало файла main.py, кое-что оставил в комментариях как альтернативу):

Исследуйте следующий пример (ссылка: <https://repl.it/@pCoding/severaltasks>):

Этот код представлен далее в трёх файлах. В файле main.py - он основной и запускаемый - вы можете перечислить функции из тех заданий, которые расположены в других файлах.

Файл main.py

```
# from task_2 import *  
from task_2 import dialog  
from task_3 import dialog as d  
# import task_3  
import task_3 as t3  
  
dialog()  
d()  
t3.dialog()
```

Файл task_2.py

```
def my_sqrt(num):  
    """  
    Вычисляет квадратный корень числа  
    """  
    return num**.5  
  
def dialog():  
    """  
    Организация диалога с пользователем  
    """  
    n = int(input('Введите число - '))  
    print('корень числа {0} = {1}'.format(n, my_sqrt(n)))
```

Файл task_3.py:

```
def is_prime(num):  
    """  
    Определяет простое число или нет  
    """  
    result = True  
    for i in range(2, num):  
        if num%i==0:  
            result = False; break  
    return result  
  
def dialog():  
    """  
    Организация диалога с пользователем  
    """  
    n = int(input('Введите число - '))  
    result = 'простое' if is_prime(n) else 'сложное'  
    print('число {0} = {1}'.format(n, result))
```

ПРАКТИКУМ





Лабораторная работа 1.

Диалоги и ветвления.

Задания для самостоятельного исполнения.

1. Пользователь вводит число. Программа определяет его чётность/нечётность и выводит ответ в виде слова: «чётное» или «нечётное».
2. Пользователь вводит два целых числа. Программа определяет: кратно ли большее из них по отношению к меньшему. Например, ввели два числа 11 и 33 – ответ: «33 кратно 11».
3. Пусть даны три числа. Программа выбирает меньшее из трёх.
4. Пусть даны длины трёх отрезков. Программа определяет можно ли из них построить треугольник.
5. Пользователь вводит радиус круга. Программа в ответ выводит два ответа в столбик: длина окружности и площадь круга.
6. Пользователь вводит коэффициенты (a, b, c) квадратного уравнения. Программа ищет корни уравнения и выводит на экран.

ВНИМАНИЕ!

Нужно уметь запускать программу и из редактора, и из консоли.

Нужно уметь передавать параметры для программы и в процессе диалога и из командной строки консоли.

Нужно уметь запускать программу с помощью bat-файла.

Нужно уметь выводить данные как в столбик, так и в одну строку.

Для справки

Для использования математических констант и функций можете в вашу программу импортировать математический модуль:

```
import math
```

Если потребуется вывести результат с заданным количеством знаков после запятой, то можно использовать форматированный вывод:

```
print('%0.2f' % (result))
```

Если необходимо произвести вычисления с округлением до заданного количества знаков после запятой, то следует употреблять функцию round:

```
result = round(x, 3)
```



Лабораторная работа 2.

Циклические алгоритмы.

Задания для самостоятельного исполнения.

1. Пользователь вводит положительное целое число. Программа выводит на экран сумму цифр этого числа.
2. Пользователь вводит положительное целое число. Программа проверяет, является ли оно степенью двойки. Если является, то вывести YES, иначе - NO.
3. Напишите программу перевода десятичного числа в восьмеричное.
4. Напишите программу вычисления суммы **n** **первых** членов ряда:

$$2/1 + 2/3 + 4/3 + 4/5 + 6/5 + 6/7 + \dots$$

Пользователь указывает величину **n**.

5. Напишите программу вычисления значения тригонометрического ряда с заданной точностью **q**:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Пользователь указывает необходимую точность **q** (например, 0.01).

6. Это задание из книги "ООП в C++". Р. Лафоре. стр. 139. Напишите код, который строит пирамиду из символов "X". На вход подаётся одно целое число - это высота пирамиды, на выходе - сама пирамида.

```
C:\WINDOWS\py.exe
Введите высоту пирамиды: 7
  X
 XXX
XXXXX
XXXXXXX
XXXXXXXXX
XXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXXXX
Для продолжения нажмите любую клавишу . . .
```



Лабораторная работа 3.

Работа со строками.

Задания для самостоятельного исполнения.

Пользователь вводит целое число n , а программа в ответ выводит на экран некоторую фигуру из символов - далее приведены примеры выполнения заданий для $n=5$.

Напишите функции, которые будут делать так же для любого n в диапазоне от 1 до 10:

1) XXXXXX	2) XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX	3) XXXXXX ----- XXXXXX ----- XXXXXX	4) X-X-X X-X-X X-X-X X-X-X X-X-X
5) X----- -X----- --X--- ---X- ----X	6) ----X ---X- --X-- -X--- X----	7) X-X-X -X-X- X-X-X -X-X- X-X-X	8) 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5
9) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25		10) 1 2 3 4 5 10 9 8 7 6 11 12 13 14 15 20 19 18 17 16 21 22 23 24 25	

Бонусное задание: вывести на экран таблицу умножения...



Лабораторная работа 4.

Работа с файлами.

Рассмотрим базовые способы работы с текстовыми файлами.

Пусть есть текстовый файл `input.txt`:

```
1 2 3
4 5 6
7 8 9
```

Рассмотренный далее код расположен по этому адресу:

<https://repl.it/@pCoding/ReadWrite>

Прочитаем содержимое текстового файла и выведем на экран:

```
f1 = open('input.txt', 'r') # открываем файл для чтения
text = f1.read() # читаем все символы
print(text) # вывести все символы на экран
f1.close() # закрываем файл
```

Можно поступить иначе, считать файл в строковую переменную, разбить её на список по сепаратору «перенос строки»:

```
f1 = open('input.txt', 'r') # открываем файл для чтения
text = f1.read() # читаем все символы
lines = text.split('\n') # разбиваем на массив строк
f1.close() # закрываем файл
```

и вывести все элементы списка на экран с помощью цикла по коллекции:

```
for line in lines:
    print(line) # выводим на экран все строки
```

или вывести строки на экран с помощью цикла по индексам с проверкой условия:

```
n = len(lines) # узнаём количество строк
for i in range(n):
    if i>0: # выводим часть списка
        print(lines[i])
```

или вывести строки в обратном порядке в другой текстовый файл:


```
f2 = open('output.txt', 'w')
for i in range(n-1, -1, -1): # строки в обратном порядке
    line = lines[i] + '\n'
    f2.write(line) # выводим строку в файл
f2.close()
```

или подсчитать сумму чисел в определённом столбце:

```
acc = 0
for line in lines:
    lst = line.split(' ')
    acc += int(lst[0]) # сумма чисел в левом столбце
print("acc =", acc)
```

или создать из списка строк двумерный список:

```
# создать двумерный массив
tab = [] # создали пустой список
for line in lines:
    tmp = line.split(' ') # делаем из строки список
    tab.append(tmp) # добавляем его в tab
```

и найти в нём сумму чисел в левом столбце:

```
acc = 0
for row in range(len(tab)):
    acc += int(tab[row][0]) # строка row и столбец 0
print("acc =", acc)
```

или просто вывести все элементы двумерного списка на экран:

```
for t in tab:
    print(*t)
```

или поменять знаки для всех чисел, лежащих на главной диагонали:

```
for row in range(len(tab)):
    t = tab[row]
    for col in range(len(t)):
        if row==col:
            tab[row][col] = -int(tab[row][col])
for t in tab:
    print(*t)
```

Задания для самостоятельного исполнения.

На вход в программу подаётся текстовый файл `input.txt` с квадратной матрицей целых чисел:

```
1 2 0 3 4
9 9 -4 8 0
1 6 5 7 0
9 9 -4 8 0
1 6 5 7 0
```

1. вывести на экран все строки этого файла.
2. вывести на экран только нечётные строки этого файла.
3. вывести в файл `output.txt` все строки исходного файла, только пронумерованные, то есть перед каждой строкой стоит порядковый номер (начать с 1) и пробел, потом уже сама строка.
4. вывести на экран сумму всех чисел этого файла.
5. вывести на экран строку с максимальной суммой чисел в строке.
6. вывести на экран два числа: сумму чисел на главной и второстепенной диагоналях матрицы.
7. транспонировать содержимое матрицы и вывести матрицу на экран.

Бонусные задачи.

8. На вход в программу подаётся целое число n . Задача: вывести в текстовый файл `output.txt` таблицу умножения размерностью $n \times n$ (строки и столбцы от 1 до n включительно).
9. На вход в программу подаются целые числа n и m . Задача: вывести в текстовый файл `output.txt` квадратную матрицу размерностью $n \times n$, содержащую целые случайные числа в диапазоне от 0 до m включительно.



Лабораторная работа 5.

Алгоритмы анализа данных.

Задания для самостоятельного исполнения.

1. Свой-Чужой

Жители племени «Тумба-Юмба» единственные в долине, которые умеют правильно отвечать на один особенный вопрос. По правильному ответу их можно отличить от жителей других племён.

Вот этот вопрос: назовите сумму целых чисел от 1 (включительно) до N (включительно). Не стоит недооценивать задачу, ведь N может быть отрицательным.

Напишите программу для проверки жителей. На вход в консоли с клавиатуры вводится одно целое число N (по модулю не более 10^4), а на выход – на экран выводится искомая сумма.

2. Дороги

В племени «Тумба-Юмба» есть N точек для охоты, некоторые из которых соединены тропами. Вождь племени решил провести инвентаризацию троп. Но, он не силен в математике, поэтому он просит вас сосчитать количество дорог. Требуется написать программу, помогающую сосчитать количество дорог по заданной матрице связности точек.

Входные данные

В первой строке входного файла INPUT.TXT записано число N ($0 \leq N \leq 100$). В следующих N строках записано по N чисел, каждое из которых является единицей или нуликом. Причем, если в позиции (i, j) квадратной матрицы стоит единица, то i -ый и j -ый точки охоты соединены тропами, а если нулик, то не соединены.

Выходные данные

На экран необходимо вывести число, определяющее количество троп. Пример.

Если входной файл такой:

```
5
0 1 0 0 0
1 0 1 1 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
```

то правильный ответ = 3.

3. Прямоугольники

Дети племени «Тумба-Юмба» любят играть в логические игры. Однажды вождь племени придумал детям задачу на построение прямоугольников одинаковой площади. Пусть дана площадь прямоугольника, тогда нужно найти количество различных прямоугольников с целочисленными длинами сторон заданной площади. Например, для площади 20 можно построить три различающихся прямоугольника с такой же площадью.

Итак, пользователь с клавиатуры вводит одно целое положительное число, не превосходящее 10^9 – это заданная площадь прямоугольника. А на экран нужно вывести искомое количество прямоугольников.

4. Результаты охоты

Вождь племени «Тумба-Юмба» очень аккуратен в подсчёте успехов на охоте, но он особенным образом записывает результаты. В местных лесах водятся 9 разных видов животных – они обозначаются цифрами от 1 до 9. После охоты вождь раскладывает трофеи по видам животных подряд, например, так:

334555

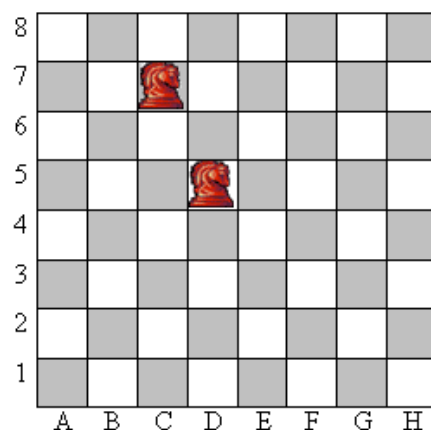
то есть пойманы две «тройки», одна «четвёрка», три «пятёрки», и, затем, так же как мы только что назвали результаты, так и записывает их: 231435

Вождь таким образом шифрует результаты охоты. Помогите вождю – напишите программу, которая из входной строки будет делать зашифрованную. Длина входной строки не превышает 100 символов.

5. Ход конём

Интеллектуалы племени «Тумба-Юмба» после охоты учатся играть в шахматы. На этой неделе они осваивают ход Конём. Напишите программу, которая будет проверять правильность хода.

На вход подаётся строка – это шахматная запись начальной позиции фигуры и конечной позиции, например, так: D5–C7



На выход нужно вывести результат анализа записи хода.

YES

- если указанный ход верный,

NO

- если такой ход не по правилам, указанным для Коня.

Гарантируется, что запись хода будет указана корректно, то есть будет состоять из 5-ти символов, в середине «дефис», буквы и цифры на правильных местах и в разрешённом диапазоне для шахматной доски.

6. Шамбала

Каждый нечётный день года шаман племени «Тумба-Юмба» должен выкладывать из камешков особенную фигуру, называемую Шамбалой, для того, чтобы боги до следующего нечётного дня были благосклонны к жителям племени. Как строится Шамбала посмотрите из примеров:

1-й день	3-й день	5-й день																																																																																																											
<table><tr><td>х</td></tr></table>	х	<table><tr><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td></tr><tr><td>х</td><td></td><td></td><td></td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td></td><td>х</td></tr><tr><td>х</td><td></td><td></td><td></td><td>х</td></tr><tr><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td></tr></table>	х	х	х	х	х	х				х	х		х		х	х				х	х	х	х	х	х	<table><tr><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td></tr><tr><td>х</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td></td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td></td><td></td><td></td><td></td><td>х</td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td></td><td>х</td><td></td><td>х</td><td></td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td></td><td></td><td></td><td>х</td><td></td><td>х</td></tr><tr><td>х</td><td></td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td></td><td>х</td></tr><tr><td>х</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>х</td></tr><tr><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td><td>х</td></tr></table>	х	х	х	х	х	х	х	х	х	х								х	х		х	х	х	х	х		х	х		х					х	х	х		х		х		х		х	х		х				х		х	х		х	х	х	х	х		х	х								х	х	х	х	х	х	х	х	х	х
х																																																																																																													
х	х	х	х	х																																																																																																									
х				х																																																																																																									
х		х		х																																																																																																									
х				х																																																																																																									
х	х	х	х	х																																																																																																									
х	х	х	х	х	х	х	х	х																																																																																																					
х								х																																																																																																					
х		х	х	х	х	х		х																																																																																																					
х		х					х	х																																																																																																					
х		х		х		х		х																																																																																																					
х		х				х		х																																																																																																					
х		х	х	х	х	х		х																																																																																																					
х								х																																																																																																					
х	х	х	х	х	х	х	х	х																																																																																																					

То есть на экран, в символическом виде, нужно вывести концентрические круги символами 'X' и пробелами, количество которых зависят от номера N текущего дня. N вводит пользователь.



Лабораторная работа 6.

Байки Солнечного города.

Задания для самостоятельного исполнения.

1. Тот-кого-нельзя-называть

Чтобы настроение жителей Солнечного города находилось по «Шкале Счастья» в рамках от Хорошего до Прекрасного достаточно обеспечить такое распределение зарплат, чтобы коротышка с самой низкой зарплатой получал не менее 10% от самой большой зарплаты в городе. Самый активный житель города – Тот-кого-нельзя-называть, его все так называли, даже мэр города – Тот-кого-нельзя-оскорблять, решил отслеживать настроение жителей. Помогите ему и напишите программу.

На вход подаётся последовательность целых чисел (это зарплаты горожан), например, такая:

600 100 500 1000

на экран нужно вывести отношение самой низкой зарплаты к самой высокой, выраженное в процентах с округлением до целого.

2. Уравнение Незнайки

Уравнение для Незнайки представляет собой строку длиной 5 символов, например, такую: $x+5=7$.

Второй символ строки является либо знаком '+' (плюс) либо '-' (минус), четвёртый символ — знак '=' (равно). Из первого, третьего и пятого символов ровно два являются цифрами из диапазона от 0 до 9, и один — буквой x , обозначающей неизвестное.

Требуется написать программу, которая позволит решить данное уравнение, то есть найти величину x .

На вход подаётся одна строка, например:

$3-x=9$.

На выход значение x :

-6

3. Нумерология Кнопочки

Как и многие другие коротышки, малышка Кнопочка верит во всякие чудеса, обожает разные гадания и нумерологию. Кнопочка руководит солнечногорской гостиницей и считает, что некоторым особенным

клиентам нужно делать значительную скидку, чтобы коротышечьи боги были благосклонны к её бизнесу. Особенность клиентов определяется по их порядковому номеру заселения следующим образом: пусть n – номер заселения, нужно подсчитать сумму всех целых чисел (от 1 до n), на которые n делится без остатка и, если эта сумма окажется нечётной, то данный гость получает скидку (если $n=3$, то $1+3=4$ – не особенный гость; если $n=4$, то $1+2+4=7$ – особенный). Кнопочка не успевает выполнять все обязанности по гостинице и ей очень нужна программа, которая автоматизирует процесс вычисления особенных клиентов.

На вход подаётся текстовый файл формата – порядковый номер клиента, символ табуляции, имя клиента, например:

- 1 Стекляшкин
- 2 Пилюлькин
- 3 Смекайло
- 4 Растеряйка
- 5 Молчун
- 6 Сахариныч
- 7 Винтик
- 8 Шпунтик
- 9 Пачкуля Пёстренький
- 10 Гусля
- 11 Винтик
- 12 Шпунтик

Нужно вывести на экран список Фамилий особенных клиентов.

4. Шифрование Пилюлькина

Пилюлькин решил производить сладкую газировку в промышленных масштабах, но монополии запрещены в Солнечном городе. Однако он решил работать через подставные компании, а рецепт и ингредиенты каждый раз оставлял в одном из номеров солнечногорской гостиницы. Своим подельникам, Жулио и Нига, из Лос-Поганоса он передавал лишь зашифрованную запись, публикуя её в «Газете дураков». Запись состоит из последовательности нулей и единиц без пробелов общей длиной не более 255 символов. Самая длинная непрерывная цепочка нулей в шифровке Пилюлькина и обозначала номер, которые должны были снять подельники в гостинице и там забрать искомые компоненты.

Например, если дана такая последовательность – 00101110000110, то зашифрован номер 4.

Продажные полицейские солнечногорска не в состоянии разгадать такой шифр, а, возможно и не хотят, так как они продажные.

Помогите детективу Биглю, не являющемуся полицейским, - напишите программу, которая по шифру Тилюлькина будет восстанавливать зашифрованный гостиничный номер.

5. Лекции Незнайки

Незнайка регулярно посещает лекции Знайки по истории, но он так неаккуратно ведёт записи, что это каждый раз становится проблемой при подготовке к экзаменам. Незнайка решил исправляться - ему нужно вести статистику своих ошибок, чтобы отслеживать улучшения.

Ошибки Незнайки состоят в том, что он в каждом записываемом имени допускает ровно одну орфографическую ошибку, то есть заменяет ровно одну из букв имени какой-то другой буквой.

Пусть дан список правильных написаний имён, и список имён или других слов из записей Незнайки. Сопоставлять два списка сам Незнайка не в состоянии. Напишите программу, которая поможет вести статистику ошибок Незнайки.

Входные данные - это текстовый файл «input.txt».

В первой строке файла находится целое число N - это количество правильных имён из лекций Знайки. Следующие N строк содержат правильные имена. Далее идет строка, содержащая целое число M - количество «подозрительных» имен и слов из записей Незнайки. Следующие M строк - это те самые имена с ошибкой и слова. Каждое из имен - последовательность из K заглавных букв английского алфавита ($1 \leq N, M, K \leq 30$).

По результатам работы программы на экран нужно вывести одну строку, состоящую из N чисел - для каждого правильного имени выводится количество «подозрительных слов» из записей Незнайки, то есть тех, которые отличаются строго на одну букву.

Пример входного файла:

```
3
ZEUS
POSEIDON
AFINA
4
ZEVS
POSEYDON
AVYNA
ZERS
```

Вывод на экран:

```
2 1 0
```

6. Равнобедренные треугольники

Знайка иногда ведёт частные уроки по геометрии и ему нужно рисовать равнобедренные треугольники. Но это рутинная операция и Знайке совсем не хочется тратить на это своё время. Помогите ему и напишите соответствующую программу.

На вход подаётся целое число N в диапазоне от 1 до 30.

По результатам работы программы на экран выводится равнобедренный треугольник по такому формату: N – это высота треугольника, основание треугольника параллельно вертикальной стороне экрана, остальные параметры смотрите по примерам:

Ввод N:	1	2	3	4	5
Вывод:	*	* ** *	* ** * * ** *	* ** * * * * * * ** *	* ** * * * * * * * * ** *



Безумное чаепитие.

Задания для самостоятельного исполнения.

В этой работе нужно написать функции и разместить их в модуле `utils.py`.
Основная программа должна обращаться к функциям из модуля `utils.py`.

1. Точное время.

Около дома под деревом стоял накрытый стол, а за столом пили чай Мартовский Заяц и Безумный Шляпник.

Стол был большой, хорошо сервированный, готовый к чаепитию. Однако чаёвники сидели с пустыми чашками. Это ничуть не смутило Алису - она выбрала место

сама и, не спрашивая, уселась в удобное большое кресло.

- Приступим, - дерзко сказала Алиса.

- Мы не можем, - нервно выкрикнул Мартовский Заяц.

- Отчего же? - не унималась Алиса.

- От того, что мы не знаем, когда заварится чай... - с горечью вымолвил Безумный Шляпник.

Тягостная пауза. Алиса с недоумением смотрит на присутствующих.

- Ну, вот смотри, - нехотя продолжил Шляпник, вынимая из кармана часы. - Сколько сейчас времени?

- Полпервого, - уверенно заявила Алиса.

- Что ты, что ты, - засуетился Мартовский Заяц, - твоя самоуверенность тебя погубит, подытожил он.

Шляпник решил прояснить ситуацию:

- У нас принято всё делать точно. "Сейчас не полпервого"...

Мартовский кот пренебрежительно хмыкнул.

- Сейчас 12.28.05, - уточнил Шляпник.

- А зачем говорить секунды? - продолжала вредничать Алиса.



- А затем, глупое создание, что наш чай нужно заваривать ровно 333 секунды и не секундой дольше.



Помогите Алисе, напишите функцию, в которую в качестве аргументов подаются начальное время и количество секунд, а функция возвращает конечное время.

Начальное время всегда задаётся в формате: XX:XX:XX, то есть, например: 00:01:02 или 14:14:14 - всегда по два разряда на часы, минуты и секунды.

Нужно найти сколько будет времени, если прибавить Y секунд.

Примеры. В первой строчке подаётся начальное время, во второй количество секунд для заваривания чая.

Вход: 00:00:00 1	Вход: 00:00:58 3	Вход: 09:54:28 333
Выход: 00:00:01	Выход: 00:01:01	Выход: 10:00:01

Гарантируется, что итоговое время не превысит 23:59:59

2. Палиндромы.

- А что будет, если мы не уложимся в точное время заваривания чая? - никак не унималась Алиса.

- Ничего хорошего не будет, - буркнул в ответ Заяц.

- Он не далёк от истины, последствия непредсказуемы, - пояснил Шляпник.

- В прошлый раз время пошло вспять, - продолжил он после паузы, занятой отхлёбыванием свежесваренного ароматного чая.

- Как же вы выкрутились? - удивилась Алиса.

- А также, ответил Шляпник, - мы знали заранее и подготовились, мы пользовались только числами, словами и фразами, которые и в прямом и в обратном направлении читаются одинаково - палиндромами (данный термин имеет греческое происхождение и обозначает "движущийся обратно").



Например, число «1231» и слово «рочот» - не палиндромы, они нам не нужны. А вот, число "12321" - палиндром, слово "топот" и фраза "Аргентина манит негра!" - палиндромы. Их можно использовать при любом течении времени...

Помогите Алисе и напишите функцию, которая будет проверять вводимую строку на палиндромность.

При проверке на палиндромность регистр буквы не имеет значение, знаки препинания и знаки пунктуации не учитываются.

Примеры палиндромов:

На в лоб, болван.

Я иду с мечем судия. Автор Гавриил Державин.

А роза упала на лапу Азора. Автор Афанасий Афанасьевич Фет.

Есть даже стихи:

Перевертень. Автор Велимир Хлебников

Кони, топот, инок,

Но не речь, а черен он.

Идем, молод, долом меди.

Чин зван мечем навзничь.

Голод, чем меч долот?

Пал, а норы худ и дух ворона лап.

А что? Я лав? Воля отча!

3. Игра.

- О чём вы общаетесь во время чаепития? - спросила Алиса всех присутствующих.

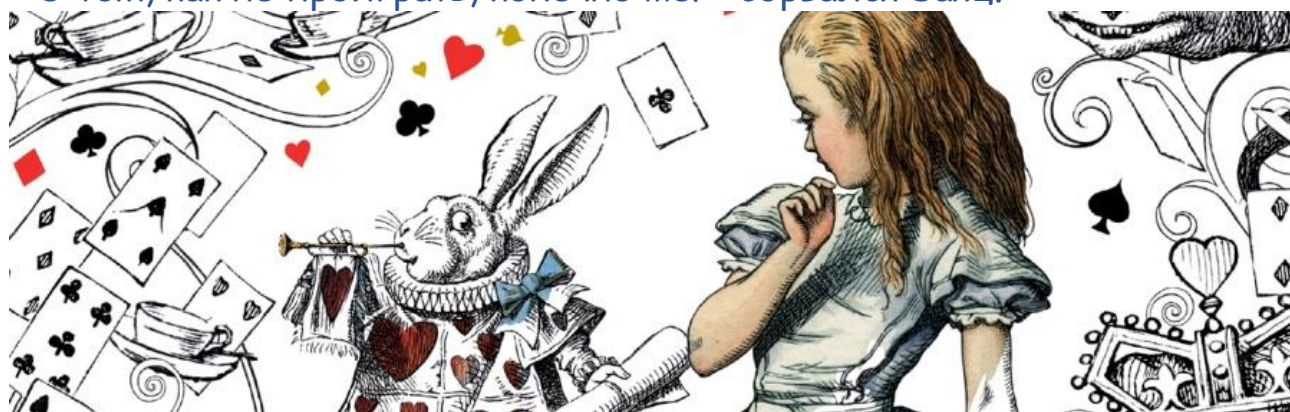
- Ни о чём! - фыркнул Мартовский Заяц.

- Как так? - озадачилась Алиса.

- Мы не занимаемся пустой болтовнёй, мы думаем... - многозначительно добавил Шляпник.

- О чём же вы думаете? - с ехидством поинтересовалась Алиса.

- О том, как не проиграть, конечно же! - сорвался Заяц.



- Или о том, как победить, - уточнил Шляпник, ну, вот смотри: загадаю я, предположим, целое число, а ты должна угадать его за минимальное количество попыток. В попытке ты говоришь мне предполагаемое число. После каждой твоей попытки я тебе отвечаю либо "угадала", либо "больше", либо "меньше". В начале игры я выдаю тебе 10 печенюшек к чаю, за каждую ошибочную попытку ты мне возвращаешь одну печенюшку. Все печенюшки, которые останутся после угадывания моего числа, ты можешь использовать при чаепитии - понятно?
- А если с первой угадаю? - пыталась съязвить Алиса.
- Угадала бы, если бы я загадывал число от 0 до 1, - ухмыльнулся Шляпник, - я же загадываю трёхзначное число.

Напишите программу для игры в угадывание трёхзначного числа.

Программа сначала сама загадывает трёхзначное число. Далее пользователь пытается его угадать. Сначала у пользователя 10 печенюшек. С каждой неудачной попыткой у него изымается 1 печенюшка.

Попытка - это ход числом. Например, загадано $x=666$. Пользователь ходит 400, программа должна ответить из списка вариантов (ровно, больше, меньше). Таким образом, пользователь, делая ходы, последовательно приближается к загаданному числу. Игра заканчивается, когда программа ответит на ход "ровно" или печенюшки закончатся. Нужно вывести на экран число оставшихся печенюшек.

4. Шаги.

- Ну, ладно, засиделась я с вами тут, - вздохнула Алиса, - пойду я.

- Не спеши, - резко выпалил Мартовский Заяц.

- А то что? - с угрозой парировала Алиса.

- А то никогда не вернёшься домой, - обречённо ответил Шляпник.

На Алису от неожиданности накатила волна страха, но она и виду не подала. Она уже поняла, что с чаёвниками можно обсудить любую тему, главное не молчать и соображать.

- Но я же знаю обратную дорогу, - с осторожностью произнесла Алиса.

- В том то и дело, что чай наш заваривался на грибах и на обратном пути твой левый шаг не будет равен правому, будет меняться.

Твой путь можно будет разбить на отдельные отрезки по X шагов. Участков будет несколько - болото, тропа, пустырь, лес и т.д., а последний участок поляна со входом в кроличью нору всегда находится в тумане, поэтому нужно будет вслепую подходить ко входу в нору.

Так вот, на каждом из участков мы определим длину шага левой ногой и правой ногой в сантиметрах, а также количество шагов на участке, получится примерно вот такая таблица:

55	45	1000
60	62	800
58	47	1100
52	59	500

Если в конце пути знать насколько больше ты прошагала одной ногой по сравнению к другой, то можно будет гарантированно вернуться ко входу в нору, и ты успешно вернёшься домой!



Помогите Алисе и напишите программу для вычисления разницы, полученной в конце пути. Входные данные брать из текстового файла "**steps.txt**". Например, для случая, указанного выше можно получить такой результат:

Левой ногой => $55 \cdot 1000 + 60 \cdot 800 + 58 \cdot 1100 + 52 \cdot 500 = 192800$ сантиметров или 1928 метров.

Правой ногой => $45 \cdot 1000 + 62 \cdot 800 + 47 \cdot 1100 + 59 \cdot 500 = 175800$ сантиметров или 1758 метров.

Таким образом, мы получаем разницу равную 170 метров.

Ответ давать в метрах без знака, то есть по абсолютной величине.

Матрица.

Задания для самостоятельного исполнения.

В этой работе нужно написать функции и разместить их в модуле `utils.py`. Основная программа должна обращаться к функциям из модуля `utils.py`.

1. Код Пифии.

Хакер Нео, не понимая своей миссии в Матрице, ищет встречи с Пифией. Личная встреча опасна, как и любое электронное сообщение, так как Нео уже давно попал под наблюдение Матрицы, и за ним охотится Агент Смит. Чтобы передавать сообщения, Пифия размещает их в сети, но адреса документов шифрует.



Описание кода Пифии. Вы знаете, что каждый символ имеет свой порядковый номер в таблице символов, таким образом буквы можно заменить на их номера, а слова на последовательности целых чисел. Например, рассмотрим кодировку для имени Нео. Первая буква - N в таблице символов стоит на 78-ой позиции - это и есть её номер. Но Пифия пошла дальше и стала кодировать эти номера в шестнадцатеричной системе счисления, то есть нужно заменить 78_{10} на $4E_{16}$, так как

$$4E_{16} \Rightarrow 4 \cdot 16^1 + 14 \cdot 16^0 = 64 + 14 = 78_{10}$$

Итак, если зашифровать имя Нео кодом Пифии, то получится такая последовательность: 4E656F, где каждые два последовательно записанных символа кодируют одну букву.



Нео проводит дни в ожидании послания от Пифии - это будет адрес некоторого файла в сети. Однажды сообщение придёт и нужно будет оперативно его декодировать.

Помогите Избранному - напишите программу для декодирования послания от Пифии и попробуйте декодировать это сообщение:

68747470733A2F2F70636F64696E672E72752F7478742F776F7264732E747874

Послание содержит адрес необходимого файла в сети. После декодирования адреса скачайте файл, он понадобится в следующем задании.

2. Бинарный поиск.

Скачанный на предыдущем этапе файл, содержит важную информацию – 100, а может даже 200 тысяч ключевых слов, расположенных в алфавитном порядке. Нео должен хранить этот файл и, при получении ключевого слова от Тринити, найти его позицию в списке слов.



Помогите Избранному и напишите программу для бинарного поиска позиции искомого слова в файле. В файле в каждой новой строчке находится новое слово. Программа должна гарантировать, что слово найдётся не более, чем за 20 итераций цикла. Линейный поиск недопустим из-за его продолжительности, что в условиях постоянной слежки недопустимая роскошь.

Проведите испытание своей программы со словами ‘ячейка’, ‘матрица’, ‘избранный’.

3. Звонок Морфеусу.

После того, как позиция слова ‘матрица’ будет найдена, её нужно будет передать Морфеусу.

Чтобы Агент Смит не догадался по какому номеру отслеживать звонок – Морфеус кодирует свой номер с помощью графических матриц, где для каждой цифры выделяется прямоугольное поле размером 2 по горизонтали и 4 по вертикали:



0	1	2	3	4	5	6	7	8	9
**	*	**	**	**	**	*	**	**	**
**	**	*	*	**	*	*	*		**
**	*	*	*	*	*	**	*	**	*
**	*	**	**	*	**	**	*	**	*

Закодированный таким образом номер телефона может выглядеть как некий узор:

```
***** *
***** * * * *
* *** * * * *
* * * * * * *
```

В этом узоре матрицы отдельных цифр размещены подряд без дополнительных пробелов. Возьмите этот узор, скопируйте и поместите в текстовом файле number.txt. Напишите программу, которая читает текстовый файл с узором, распознаёт там образы цифр и выводит на экран зашифрованный номер телефона.



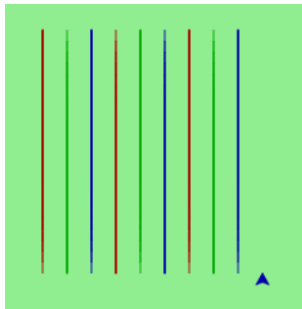
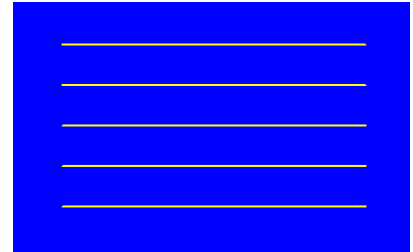
Лабораторная работа 9.

Черепашка.

Задания для самостоятельного исполнения.

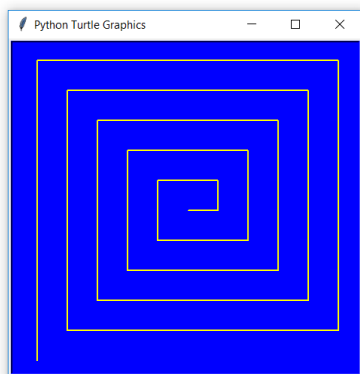
1. Простые движения.

Вывести черепашкой n горизонтальных линий. Размеры линий выбирайте сами. Пользователь вводит с клавиатуры n .



2. Цветные движения.

Вывести черепашкой n вертикальных линий. Размеры линий выбирайте сами. Пользователь вводит с клавиатуры n . Цвета чередуются в таком порядке: красный, зелёный, синий.

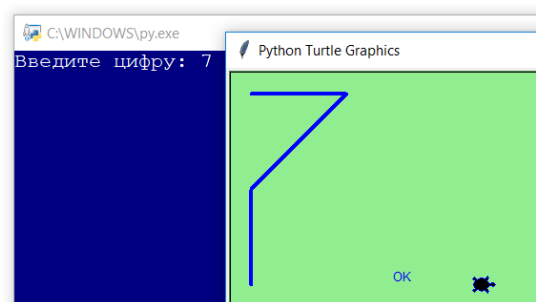
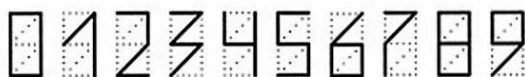


3. Спираль.

Пользователь задаёт количество петель спирали. Одна петля – это четыре стороны. На рисунке приведён пример, когда количество петель задано 5. Вывести черепашкой спираль. Начало рисования из центра экрана.

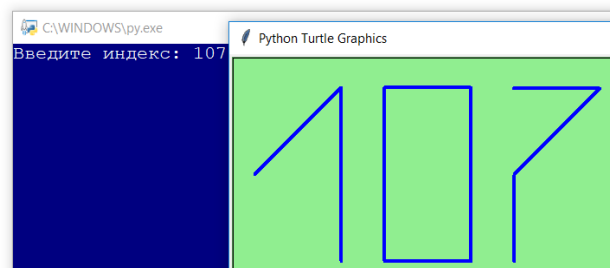
4. Цифра.

Пользователь задаёт одну десятичную цифру. Вывести черепашкой цифру по формату цифр индекса:



5. Индекс.

Пользователь задаёт индекс – шестизначное десятичное число. Вывести черепашкой индекс. Описание цифр разместить в отдельном модуле numbers.py, а в основной программе подключить модуль директивой `import`.





Черепашка. Примеры

Библиотека подробно разобрана тут: <https://docs.python.org/3/library/turtle.html>

Начертить хвостом квадрат

```
from turtle import *

win = Screen()
win.bgcolor("lightgreen")

alex = Turtle()
alex.shape("turtle")
alex.color("#4400AA")
alex.speed(1)
alex.pendown()
alex.xcor = 50
alex.ycor = 50
alex.setheading(0)

alex.forward(50) # вперёд
alex.right(90)   # повернуть
alex.forward(50) # вперёд
alex.right(90)   # повернуть
alex.forward(50) # вперёд
alex.right(90)   # повернуть
alex.forward(50) # вперёд

win.mainloop()
```

Ходим по спирали

```
import turtle

w = turtle.Screen()
w.bgcolor("lightgreen")

alex = turtle.Turtle()

alex.shape("turtle")
alex.color("#4400AA")
alex.speed(0)
alex.penup()

step = 15 # длина шага
count = 35 # кол-во шагов
for i in range(count):
    alex.stamp() # оставить отпечаток
    step += 4 # увеличить шаг
    alex.forward(step) # вперёд
    alex.right(33) # повернуть

w.mainloop()
```

Реагируем на события

```
import turtle

def goU():
    turtle.setheading(90)
    turtle.forward(step)
def goR():
    turtle.setheading(0)
    turtle.forward(step)
def goD():
    turtle.setheading(270)
    turtle.forward(step)
def goL():
    turtle.setheading(180)
    turtle.forward(step)

turtle.reset()
turtle.shape('turtle')
turtle.bgcolor('#009944')
turtle.turtlesize(2)
turtle.pencolor('yellow')

sp = int(turtle.numinput('Вопрос', 'Введите скорость: ', default=2))
step = 50
turtle.speed(sp)

turtle.pendown() # Опускаем перо перо (начало рисования)
turtle.onkey(goU, "Up")
turtle.onkey(goR, "Right")
turtle.onkey(goD, "Down")
turtle.onkey(goL, "Left")
turtle.listen() # Включить прослушивание событий
turtle.penup() # Поднять перо (закончить рисовать)

turtle.mainloop() # Задержать окно на экране
```



Объекты и сортировка.

Задания для самостоятельного исполнения.

Вспомогательные материалы
смотрите в Презентациях Лекций...

Пусть дан текстовый файл:

№	Фамилия	Имя	Отчество	Возраст	Средний балл
1	Абрамович	Петя	Иванович	24	4.5
2	Кокорин	Петро	Петрович	18	3.8
3	Саакашвили	Адольф	Семёнович	33	4.1
4	Папов	Коля	Олич	19	4.4
5	Некто	Оле	Лукойе	14	3.3

тут пробел

тут табуляция

Первая строка служебная и не участвует в обработке информации, но присутствует в файле и нужна только для наглядности представления информации. Во время чтения данных из файла учтите это и первую строку не включайте в список для обработки данных. Разделителем для столбцов Номер, «Фамилия Имя Отчество», Возраст, «Средний балл» является символ **табуляции**, а Фамилия Имя Отчество разделены **пробелом**.

Во время выполнения заданий данные из файла сначала считывайте в **список объектов**, потом уже ведите обработку этого массива в разных заданиях по-разному. Файл с указанными параметрами структуры создайте самостоятельно и выполните для него с использованием объектно-ориентированного стиля следующие **задания**:

1. Вывести в столбик на экран **Фамилии и Инициалы** студентов в виде:

Абрамович П.И.

Кокорин П.П.

Саакашвили А.С.

Папов К.О.

Некто О.Л.

2. Вывести на экран в столбик Возраст и, через пробел, Фамилии и Инициалы **совершеннолетних** студентов:

24 Абрамович П.И.

18 Кокорин П.П.

33 Саакашвили А.С.

19 Папов К.О.

3. Вывести на экран в столбик Возраст, Средний балл и Фамилии и Инициалы студентов, у которых Средний балл выше, чем средний балл в группе, **отсортировав** (можно использовать любой метод сортировки) их по возрасту:

19 4.4 Папов К.О.

24 4.5 Абрамович П.И.

33 4.1 Саакашвили А.С.

4. Напишите модуль modul.py в котором разместите функцию генерации списка случайных целых чисел и функцию сортировки методом выбора. Объясните алгоритм сортировки «**выбором**». Напишите программу, которая будет пользоваться этим модулем – в программе интерфейс пользователя, где, при старте программы, можно ввести количество и диапазон случайных чисел.

5. Добавьте в модуль функцию сортировки методом пузырька. Объясните алгоритм сортировки «**пузырьком**».

6. Добавьте в программу возможность измерения времени исполнения функции сортировки. Вы знаете, что измеряемое время изменяется от запуска к запуску. Сделайте так, чтобы проверка запускалась 10 раз подряд и только затем выводилось на экран **среднее время** выполнения функции сортировки.



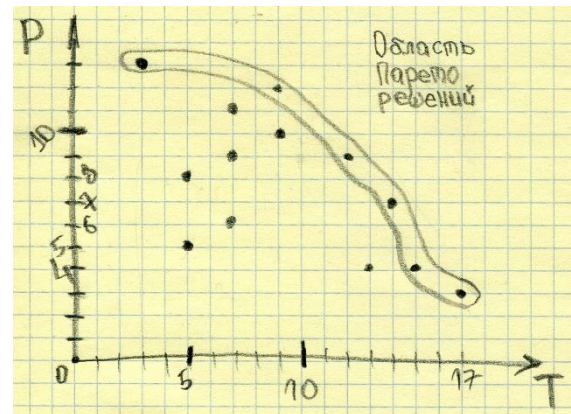
Область оптимальных решений.

Задания для самостоятельного исполнения.

Одно комплексное задание.

Дан входной текстовый файл `smart.txt`:

1	samsung	a1	17	3
2	samsung	a2	15	4
3	samsung	a3	13	4
4	samsung	a4	7	6
5	samsung	a5	5	5
6	samsung	a6	5	8
7	samsung	a7	7	9
8	samsung	a8	9	10
9	samsung	a9	7	11
10	iphone	x10	3	13
11	iphone	7 9	12	
12	samsung	8	12	9
13	samsung	se	14	7



Каждая строка файла - запись про один телефон. Запись состоит из четырёх полей: `id`, `name`, `time`, `power`. Поля разделены символом табуляции (можно использовать другой символ). Наименования и величины условны.

Техническое задание.

Разработать программу со следующим функционалом:

- 1) разработать класс `Smartphone` для хранения данных об объектах, объекты - это телефоны;
- 2) считать данные из файла `smart.txt` и поместить их в список объектов;
- 3) отсортировать список по убыванию по производительности смартфонов (поле `power`) и вывести три лучших;
- 4) отсортировать список по убыванию по продолжительности работы (поле `time`) и вывести три лучших;
- 5) выбрать объекты, которые находятся в области Pareto (см. рисунок) и вывести данные о них в текстовый файл `output.txt`. Выводить данные в том же формате, как они были во входном файле.

Декларацию класса и все функции обработки данных разместить в модуле `utils.py`. Основная программа должна обращаться к функциям из этого модуля.

Примеры тестовых заданий.

ТЕСТ_1 Python Структурные операторы

Какое значение будет в переменной x после выполнения кода:

```
x = 0
for i in range(1,5):
    x = x + i
print(x)
```

Какое значение будет в переменной x после выполнения кода:

```
x = 0
for i in range(9):
    x += i % 3
print(x)
```

Какое значение будет в переменной x после выполнения кода:

```
x = 0
for i in range(9,0,-1):
    if i % 2 == 0:
        x = x + 1
print(x)
```

ТЕСТ_2 Python Списки, строки, функции, классы

Что будет выведено на экран после выполнения кода:

```
nums = [1,2,3,4,5]
print(3 in nums)
```

Что будет выведено на экран после выполнения кода:

```
def func(x,y):
    return x if y%x==0 else y
print(func(5,6))
```

Сколько строчек будет выведено на экран в результате выполнения этого кода:

```
for i in range(1,6):
    if i<3 == 0:
        continue
    print(i)
```

Что будет выведено на экран в результате выполнения этого кода:

```
a = '22222'
print(a[2:])
```