# Основы Python. Часть 1

# Введение

**Java**

На java создают веб приложения, а точнее их Backend – т.е серверную часть.

1. Это безопасный язык;

2. Язык проверенный временем (более 20 лет).

**На java написаны все сайты сбера.**

**Переменная -** это специальное место в программе в которой хранятся какие-то данные

В java – принято слова после переменной int писать с маленькой буквы. Второе слово пишется слитно с первым, но с большой буквы.

Знак = - это символ присвоения значения;

В конце всех основных строк в java ставится «;»

If – это специальный оператор , который проверяет выполнение условия;

Чтобы написать что-то на языке java, мы должны создать проект. Проект это папка в которой размещен код.

Java – язык кроссплатфоменный, это значит, он будет работать на windows, MAC OS, linux/

**PYTHON**

1.В python текст пишется с маленькой буквы и слова разделяются нижним подчеркиванием \_.

2. После оператора if код пишется со сдвигом на одну табуляцию вправо.

**QA-тестировщик (инженеры по качеству или тестированию)**

*Функциональное тестирование* – тестирование направленное на проверку отдельных функций сайта.

*Нефункциональное тестирование* - тестирование юзабилити, т.е. удобства программы или нагрузочное тестирование – направлено на аспекты, влияющие на удовлетворенность клиента.

*Автоматизированное тестирование* – тестирование часто повторяющихся операций.

Ручное тестирование – тестирование неповторяющихся операций

**Кибербезопасность** включает в себя безопасность

- компьютерных сетей

- корпоративных инфраструктур

- веб-приложений

- мобильных приложений

- системного ПО

На калькуляторе возведите количество символов в степень, равную длине

пароля, и разделите на 86 400 (секунд в сутках) и на 365 (дней в году), а

затем на время в годах, выдаваемое сервисом.

86 400 (секунд в сутках) и на 365 (дней в году).

**Data Science**

Направления:

1. Компьютерное зрение (распознавание образов) – это набор технологий позволяющий анализировать изображение, делать по ним выводы, сравнивать их, находить похожие изображения и решать другие практические задачию

2. Технология машинного обучения – это технология обучения компьютера, способного заменить десятки, а то и сотни людей.

**Алгоритмы классификации.**

**Что значит программировать**

Рекомендованные книги для параллельного чтения с курсом:

Пол Бэрри — Изучаем программирование на Python

Рейтц Кеннет, Шлюссер Таня — Автостопом по Python

Марк Лутц — Learning Python / Изучаем Python. 5-е издание, 2019. Книга-справочник

Лутц Марк — Python. Карманный справочник

Онлайн-тренажер в игровой форме

# PYTHON-BASIC

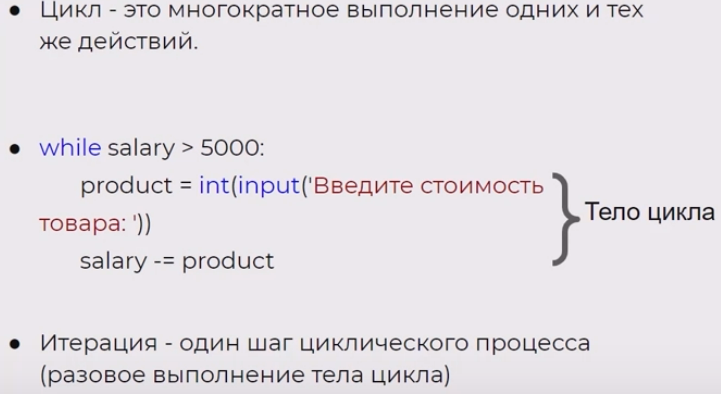
# Модуль\_5

## 5.5 Использование нескольких логических операторов.

Високосный год – это год если он одновременно кратен 4 **и не** кратен 100 **или** кратен 400.

Если при невыполнении первого условия if, нужно проверить второе, то записывается elif.

Цикл – это многократное выполнение одних и тех же действий.



**Пример кода с оператором while:**

money = int(input('Сколько у вас денег? '))

Условие цикла

while money < 500000:

saving = int(input('До целевой муммы положите еще. Сколько хотите отложить? '))

Тело цикла

money += saving

print('Вы отложили', money, 'руб.')

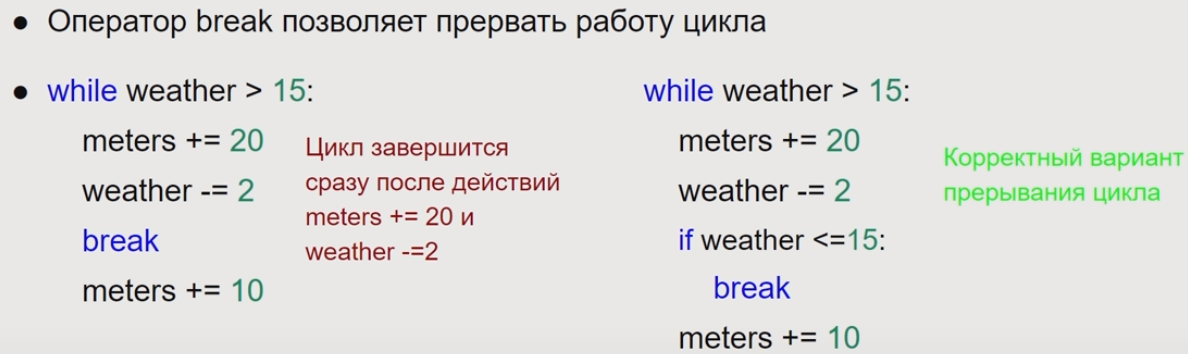
print('Достаточно, целевая сумма достигнута!')

**Оператор brake:**

Используется для резкого выхода из цикла, не дожидаясь конца его работы.

Конструкцию **brake** необходимо вызывать, только внутри конструкции **if**, т.е. должна выполнятся только при выполнении какого-то особенного условия.

А еще с помощью оператора **while,** можно зациклить запрос данных у пользователя.

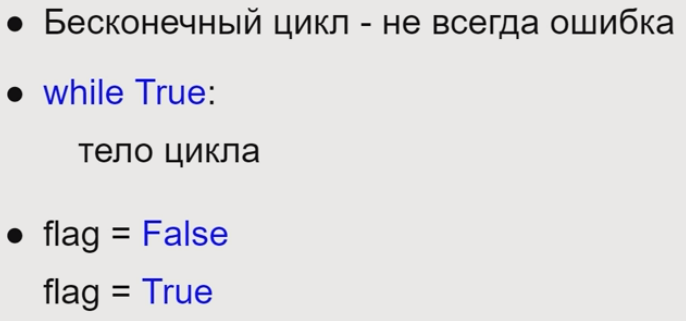


# Модуль\_6

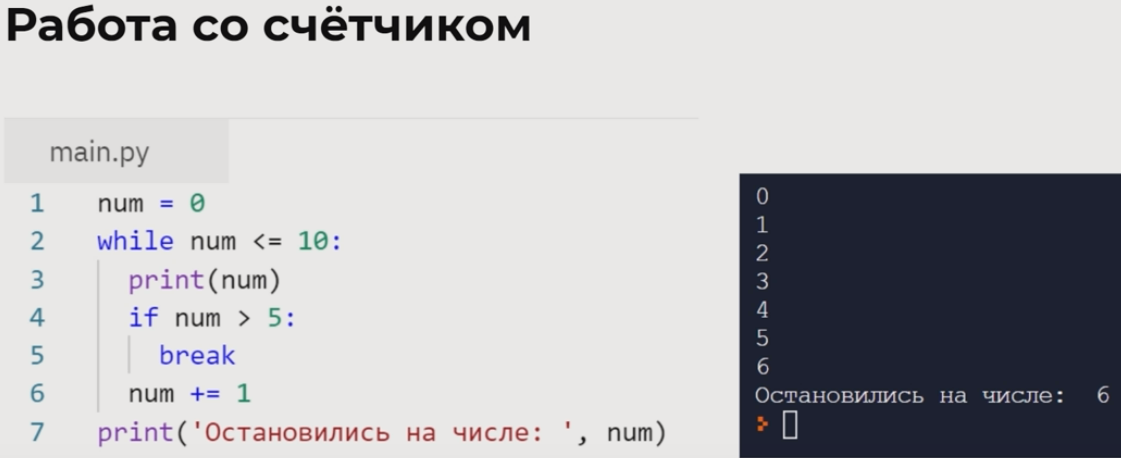
## 6.4 Бесконечный цикл. Логический тип данных.

Для создания бесконечного цикла можно использовать оператор “while True” – «пока истина»

True, False - это логический тип данных, им также можно присваивать переменные, такие переменные называются флагами, их можно использовать для разного рода переключений и проверок. Они все пишутся с большой буквы и выделяются синим цветом.

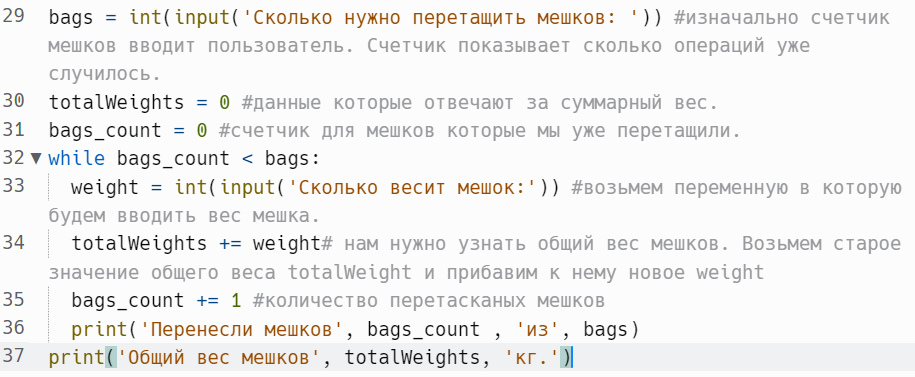


## 6.5 Цикл while со счётчиком



#Задача 3. Рыбалка

#Наши прекрасные родственники удачно сходили на рыбалку. Настолько, что ходили мешком перетаскивать рыбу с берега в машину целых n раз. Каждый мешок они взвешивали на электронных весах (все мешки весили по-разному). Напишите программу для весов, которая считает суммарный вес мешков и выводит его на экран.

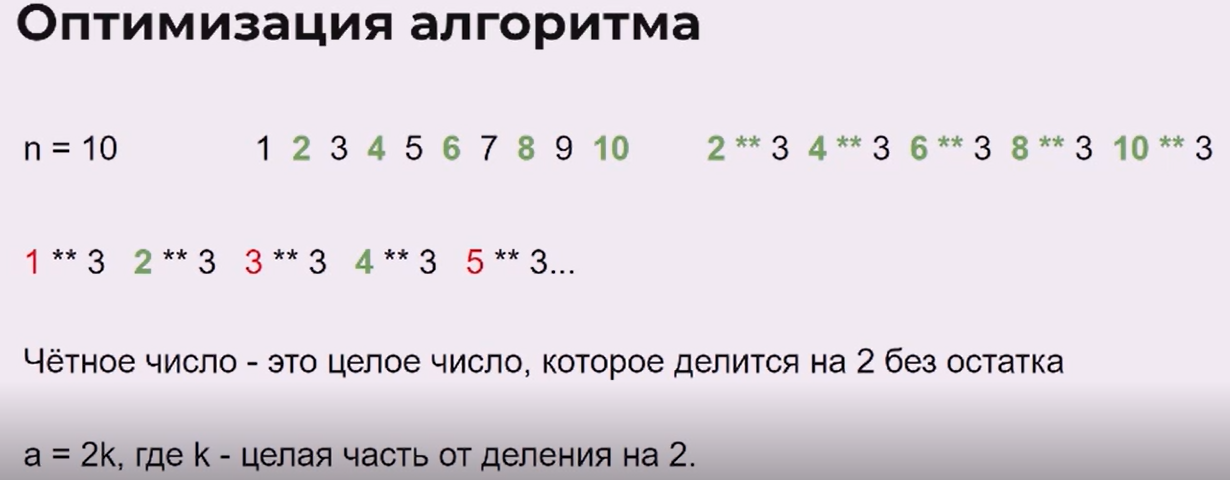


## 6.6 Оператор continue в цикле while.

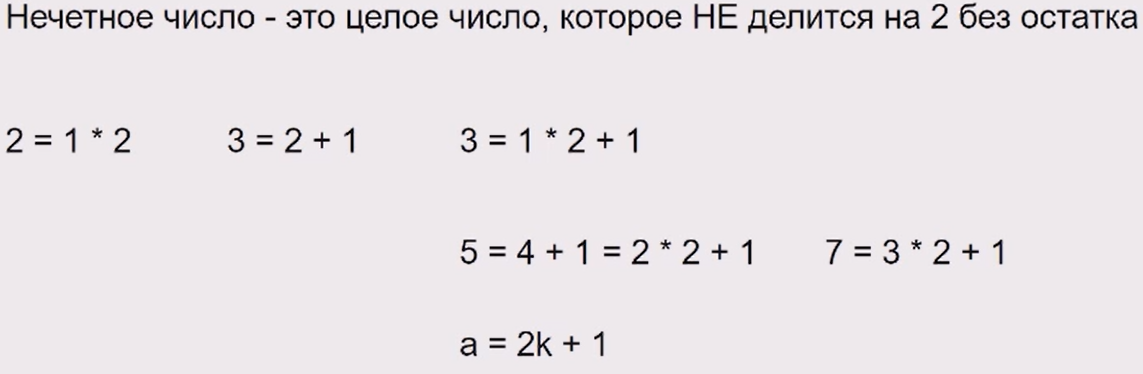
Continue – этот оператор позволяет перейти сразу же на следующую итерацию цикла, пропускает все что идет после него.

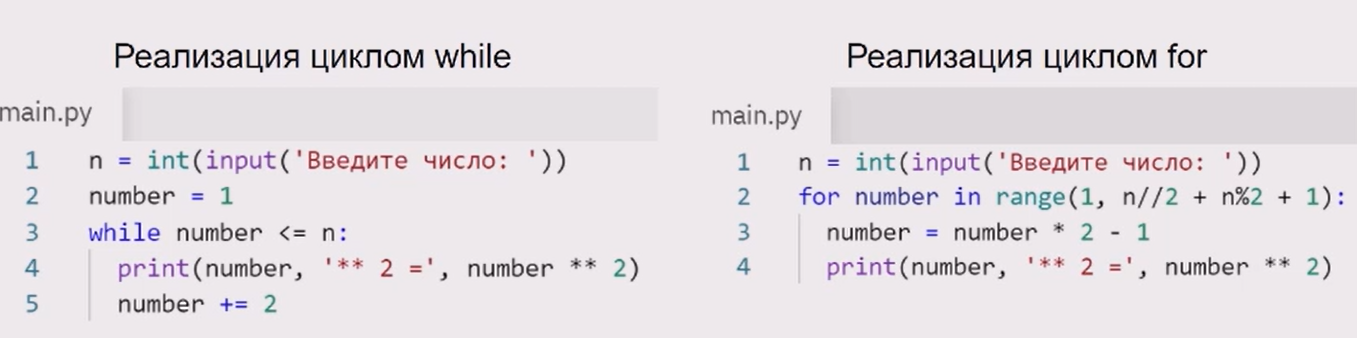
# Модуль\_8

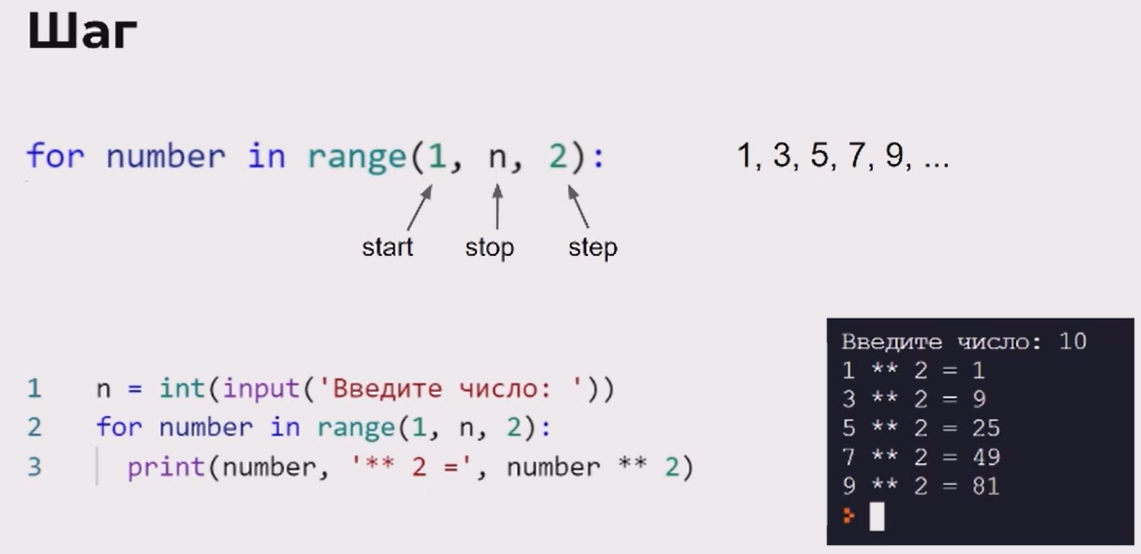
## 8.2



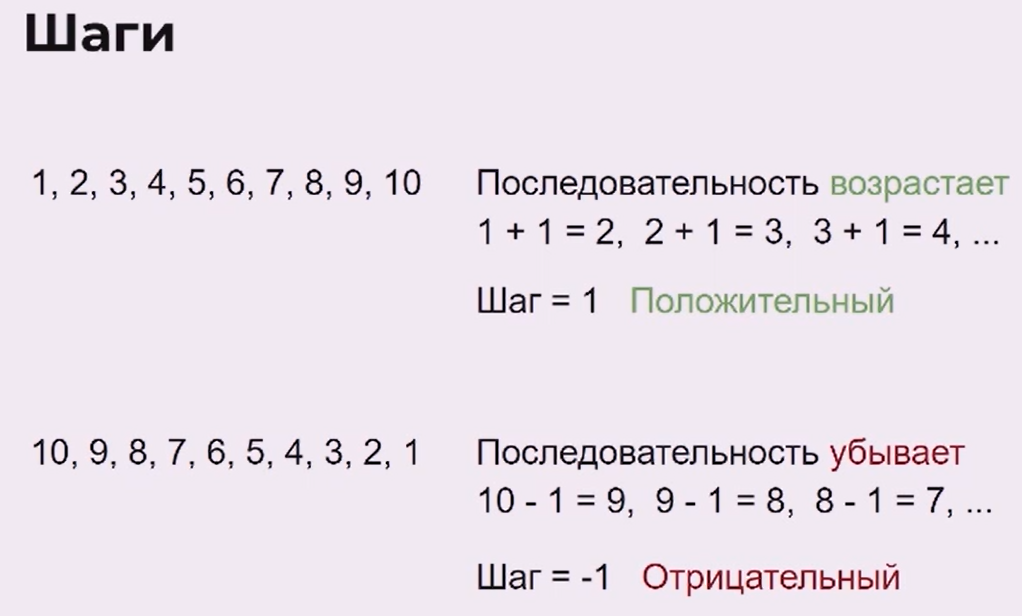
## 8.3 Функция range: start, stop, step







## 8.4 Отрицательный шаг в функции range



Итоги модуля № 8.

Пользовательский ввод start, stop, step

Чётные числа a = 2k

Нечётные числа a = 2k + 1

For number in range (10, 1, –1)

print(number)

for number in range (a, b, c) # c != 0

print(number)

range (start, stop, step)

step —шаг

# Модуль\_9

## 9.2 Сравнение строк

Хорошие уроки по патону есть на youtube.com от МФТИ. (Московского физико-технического института). Можете ввести в поисковую строку Алгоритмы на Python 3. Лекция №1 там цикл лекций на эту тему.

## 9.3 Цикл for: итерирование по строке

**# Задача 1. Python!**

# Напишите программу, которая выводит в отдельную строчку каждый символ фразы “Python!”.

**phrase = "Python!"**

**for symbol in phrase:**

**print(symbol)**

**# Задача 2. Вирус**

# Ваня экспериментирует с различного рода компьютерными вирусами, которые портят жизнь людям.

# На просторах Интернета он нашёл код довольно необычного вируса,

# который “поворачивает” весь текст в документе и повторяет каждый символ 3 раза.

# Пользователь вводит текст. Напишите программу, которая выводит каждый символ текста в отдельной строке и три раза.

# Пример:

# Введите текст: Привет!

# ППП

# ррр

# иии

# ввв

# еее

# ттт

# !!!

# Без разворота по условию задачи:

**phrase\_in = input("Введите текст: ")**

**for symbol in phrase\_in:**

**print(symbol \* 3)**

# С разворотом(с использованием не изученных пока срезов):

**phrase\_in = input("Введите текст: ")**

**for symbol in phrase\_in[::-1]:**

**print(sym**

**# Задача 3.**

# Мы входим в команду разработки нового текстового редактора и нам поручили разработать для него подсчёт нужного символа в тексте,

# а именно - буквы Ы. Причём отдельно с верхним регистром и отдельно с нижним.

# Напишите программу, которая считает количество больших и количество маленьких букв Ы в тексте и выводит ответ на экран.

# Пример:

# Введите текст: Прыг скок

# Больших букв Ы: 0

# Маленьких букв Ы: 1

**upper\_letter = "Ы"**

**lower\_letter = "ы"**

**upper\_count = 0**

**lower\_count = 0**

**phrase\_in = input("Введите текст: ")**

**for symbol in phrase\_in:**

**if symbol == upper\_letter:**

**upper\_count += 1**

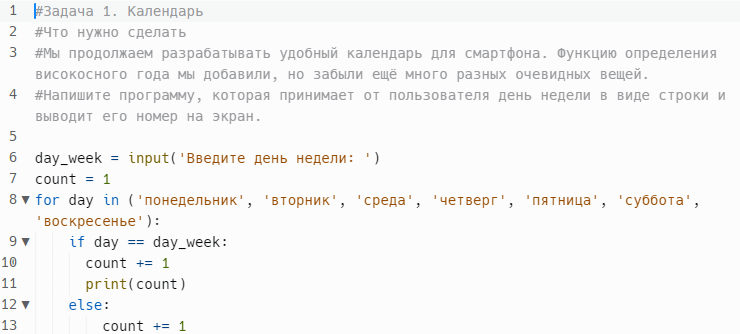
**elif symbol == lower\_letter:**

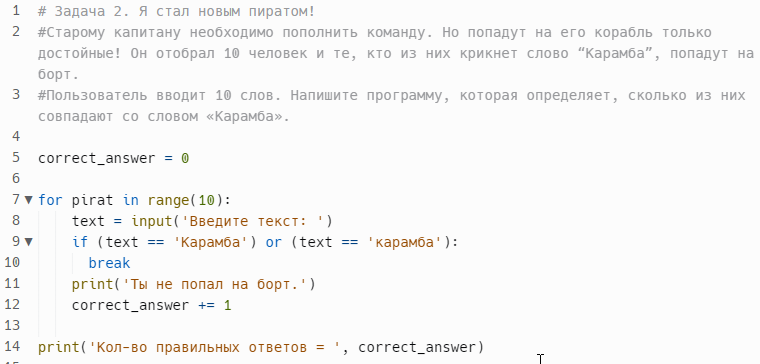
**lower\_count += 1**

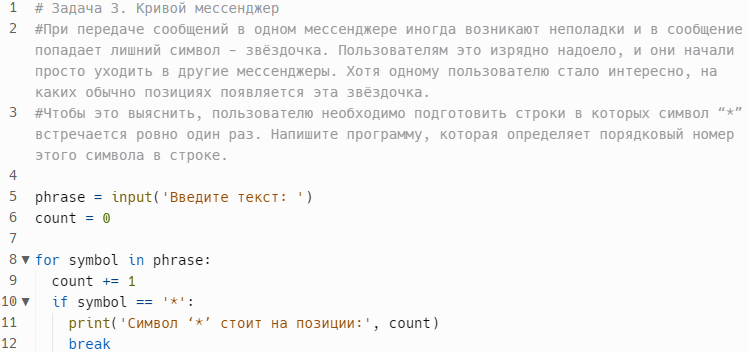
**print("Больших букв Ы: ", upper\_count)**

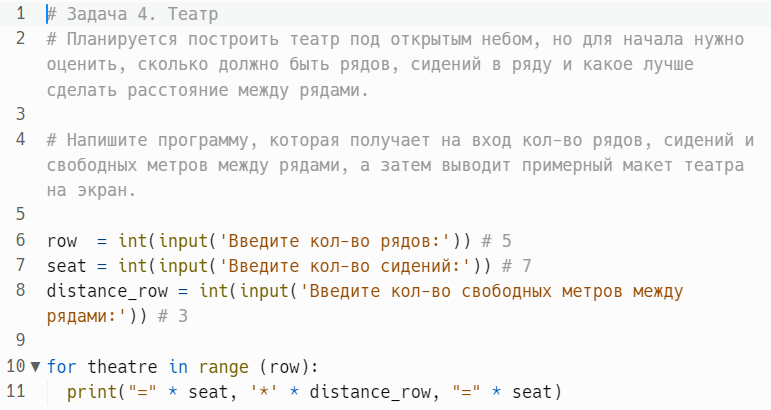
**print("Маленьких букв Ы: ", lower\_count)**

## Практическая работа к модулю №9

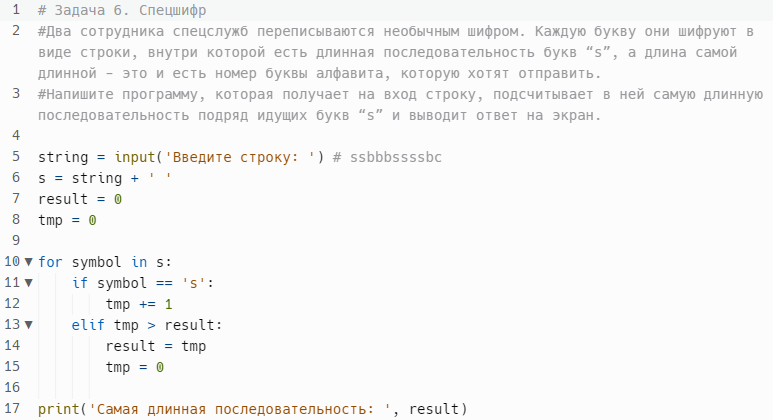












# Задача 9. Коровы

# Что нужно сделать

# Для коров есть 10 стойл. В каждом стойле разные условия для животных, поэтому и молока они дают по-разному. В первом стойле производят 2 литра в день, во втором 4, в третьем - 6, потом 8, 10, 12, 14, 16, 18, 20. Но коровы стоят не во всех стойлах. Свободные и занятые обозначаются строкой из букв a и b, где a - свободное стойло, b - занятое.

# Напишите программу для подсчета получаемого молока в коровнике, с учетом следующего взаимодействия пользователя с программой: пользователь вводит строку из 10 символов a и b. Необходимо определить, сколько в итоге будет произведено молока за день.

a = 0

b = 0

summ = 0

count = 1

summ = 0

cow = input('введите последовательность расстановки коров в стойлах (а = незанятое коровой стойло, b - занятое коровой стойло): ') # abbbbaaaba

for stall in cow:

stall == 'b'

count += 1

# if stall == 'b':

summ = (count-1) \* 2

print('сумма надоенного молока в', count-1, 'стойле', (count-1) \* 2, 'л.')

# summ = (count - 1) \* 2

# print('За день будет произведено', summ, 'литров молока.')

# Модуль\_10

Для получения матрицы не забываем в конце писать print(), во вложенном цикле.

## 10.1 Работа со вложенными циклами

**# Задача 1. Таблица умножения**

#

# Математик Паша недавно заметил, что у него уже есть куча разных таблиц степеней, но нет самого основного — таблицы умножения.

# Пора бы это исправить.

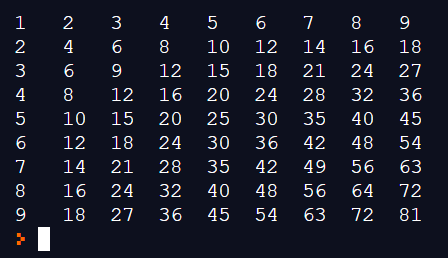
# Напишите программу, которая выводит таблицу умножения для чисел от 1 до 9.

# Для этого используйте конструкцию вложенного цикла: внешний отвечает за первый множитель, а внутренний — за второй.

# Дополнение: выведите настоящую таблицу умножения, без всяких знаков, только числа. Чтобы она получилась красивой и ровной,

# используйте литерал \t внутри оператора end. \t — это литерал табуляции, благодаря ему все числа выстраиваются в виде таблицы.

# Результат должен получиться таким:



for i in range(1, 10):

for j in range(1, 10):

print(i \* j, end='\t')

print()

**# Задача 2. Таблица суммы**

#

# Напишите программу, которая запрашивает у пользователя число N и выводит таблицу суммы для чисел от 1 до N.

#

# Пример:



n = int(input("Введите число: "))

for i in range(n + 1):

for j in range(n + 1):

print(i+j, end='\t')

print()

**# Задача 3. Анализ данных**

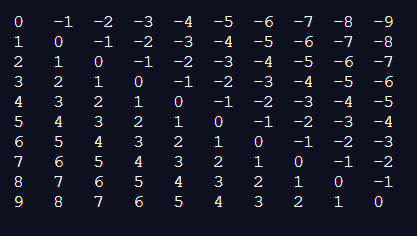
#

# Отдел анализа данных сегодня получил вот такую интересную штуку:

#

# Вам, как работнику этого отдела, дали задание понять, как и почему такое произошло.

# Напишите программу, которая выводит на экран такую таблицу.



for i in range(10):

for j in range(10):

if j <= i:

print(i-j, end='\t')

else:

print(i-j, end='\t')

print()

## 10.2 Использование if во вложенных циклах

**# Задача 1. Матрица**

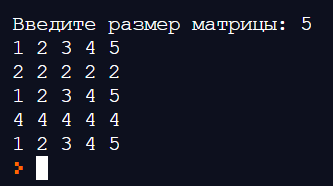
#

# Напишите программу, которая выводит квадратную матрицу размера N на N.

# В каждой нечётной строке матрицы идут числа от 1 до N, а в каждой чётной — просто числа, равные номеру этой строки.

#

# Пример:



n = int(input("Введите число: "))

for i in range(1, n + 1):

for j in range(1, n + 1):

if i % 2 == 0:

print(i, end='\t')

else:

print(j, end='\t')

print()

**# Задача 2. Чёрный ящик**

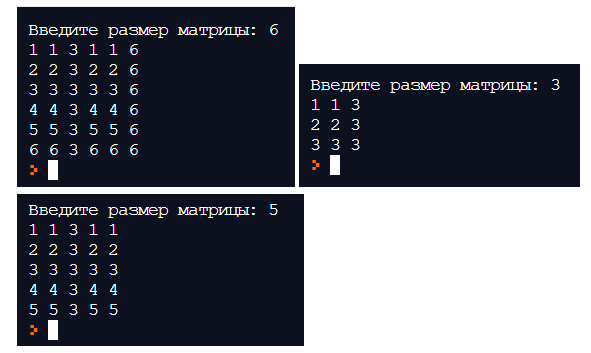
#

# Преподаватель показал студентам несколько результатов программы и сказал: «Кто догадается,

# что делает программа и как она это делает, получит зачёт автоматом».

# Студент Миша намерен получить этот зачёт. Он заметил, что в результатах программы есть определённая закономерность.

# Вот некоторые результаты:



#

# Напишите программу, которая запрашивает у пользователя размер матрицы и выводит соответствующие результаты.

#

# Подсказка: столбцы.

#

n = int(input("Введите число: "))

for i in range(1, n + 1):

for j in range(1, n + 1):

if j % 3 == 0:

print(j, end='\t')

else:

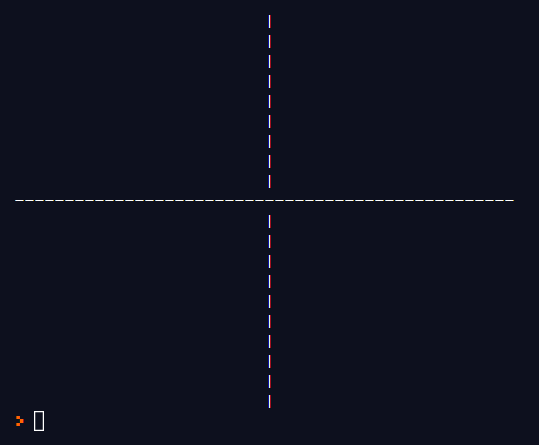
print(i, end='\t')

print()

**# Задача 3. Координатные оси**

#

# Напишите программу, которая рисует координатные оси на поле 20×50. Результат должен получиться таким:



# Что нужно поменять в коде, что в середине был не дефис, а вертикальная палочка?

x\_lim = 50

y\_lim = 20

for y in range(y\_lim):

for x in range(x\_lim):

if y == y\_lim // 2:

print('-', end='')

elif x == x\_lim // 2:

print('|', end='')

else:

print(' ', end='')

print()

# Вариант с вертикальной палочкой в середине:

for y in range(y\_lim):

for x in range(x\_lim):

if x == x\_lim // 2: # Ключевое отличие в порядке проверок. Если сперва выполняется проверка на дефис, то именно дефис будет выбран.

print('|', end='')

elif y == y\_lim // 2:

print('-', end='')

else:

print(' ', end='')

print()

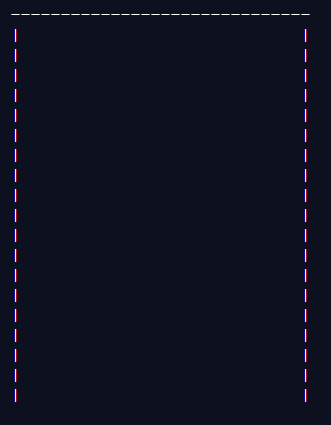
## 10.3 Работа с двумя счетчиками в условном операторе

**# Задача 1. Врата**

#

# Напишите программу, которая выводит в консоль «врата», состоящие из тире, вертикальных линий и пробелов.

# Поле состоит из 20 строк и 30 столбцов (но не стесняйтесь пробовать и другие размеры).



x\_lim = 30

y\_lim = 20

for y in range(y\_lim):

for x in range(x\_lim):

if y == 0:

print('-', end='')

elif x == 0 or x == x\_lim - 1:

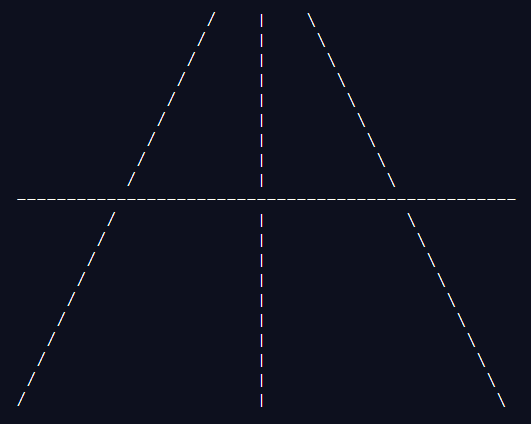
print('|', end='')

else:

print(' ', end='')

print()

**# Задача 2. Дорога**



Мы делаем текстовую игру — гонку, и нам нужно вывести на экран что-то вроде трассы, где будут соревноваться наши машины.

# Напишите программу, которая выводит такую дорогу на экран (поле 20×50).

#

# Что нужно сделать, чтобы обочины рисовались поверх горизонтальной линии?

#

x\_lim = 50

y\_lim = 20

delta = 3 # число 3 выбрано "на глаз", в зависимости от него дорога будет шире/уже

for y in range(y\_lim):

for x in range(x\_lim):

if y == y\_lim // 2:

print('-', end='')

elif x == x\_lim // 2:

print('|', end='')

elif x == x\_lim // 2 - (delta + y):

print('/', end='')

elif x == x\_lim // 2 + (delta + y):

print('\\', end='') # \ - символ который используется для экранирования, поэтому его пришлось дублировать

else:

print(' ', end='')

print()

# Вариант с рисованием обочины поверх горизонтальной линии:

for y in range(y\_lim):

for x in range(x\_lim):

if x == x\_lim // 2 - (delta + y):

print('/', end='')

elif x == x\_lim // 2 + (delta + y):

print('\\', end='') # \ - символ который используется для экранирования, поэтому его пришлось дублировать

elif y == y\_lim // 2:

print('-', end='')

elif x == x\_lim // 2:

print('|', end='')

else:

print(' ', end='')

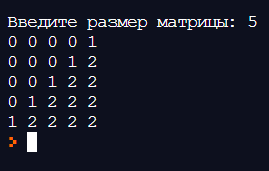
print()

**# Задача 3. Диагональная матрица**

# Напишите программу, которая получает на вход размер квадратной матрицы и выводит на экран по такому принципу

# диагональ из единиц с левого нижнего угла до верхнего правого, ниже диагонали — двойки, выше — нули.

# Пример:



n = int(input("Введите размер матрицы: "))

for y in range(n):

for x in range(n):

buf\_x = (n - 1) - y # вспомогательное число, которое будет уменьшаться от n-1 до 0

if buf\_x > x:

print(0, end='\t')

elif buf\_x == x:

print(1, end='\t')

else:

print(2, end='\t')

print()

## 10.4 Решение задач с помощью вложенных циклов

**# Задача 1. Электронная очередь**

#

# Нам дали заказ написать программу для электронной очереди. У каждого человека в очереди есть номер:

# нулевой, первый, второй, третий и так далее. Каждый час очередь уменьшается на одного человека,

# то есть уходит сначала нулевой номер, затем первый, второй и так далее. Наша программа получает

# на вход одно число — количество людей в очереди — и выводит на экран историю обслуживания очереди: какие номера в какой час оставались.

#

human\_count = int(input("Введите количество людей в очередь: "))

for human\_start in range(human\_count + 1):

print("Час:", human\_start)

for human\_number in range(human\_start, human\_count):

print(human\_number, end='\t')

print()

**# Задача 2. Цифры больше пяти**

#

# Пользователь вводит последовательность из N чисел. Напишите программу,

# которая подсчитывает общее количество цифр больше пяти во всей последовательности.

n = int(input("Количество чисел в последовательности: "))

cipher\_count = 0

for \_ in range(n):

new\_number = int(input("Введите число: "))

while new\_number:

if new\_number % 10 > 5:

cipher\_count += 1

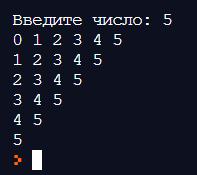
new\_number //= 10

print(cipher\_count)

**# Задача 3. Лестница чисел**

#

# Пользователь вводит число N. Напишите программу, которая по этому числу выводит вот такую лестницу из чисел:



n = int(input("Введите число: "))

for start in range(n + 1):

for number in range(start, n + 1):

print(number, end='\t')

print()

## 10.5 Блок else для цикла. Бесконечный внешний цикл

**Самое важное**

**Вложенные циклы**

Важно понять, в какой последовательности Python выполняет код.

Если мы укажем:

for i in range(10):

<тело цикла> — тело цикла повторится 10 раз.

При этом Python’у на самом деле не так важно, что будет телом цикла.

Это может быть одна операция print(i):

for i in range(10):

print(i)

Но это может быть и вложенный цикл:

for i in range(10):

for i in range(10):

print(i)

Для лучшего понимания последовательности выполнения кода используйте

дополнительные принты!

for i in range(2):

print(‘start’)

for j in range(5):

print(i, j)

print(‘end’)

Такие принты позволят явно увидеть, в каком порядке выполняются операции.

**Пример**

Итерация первого внешнего цикла

start

пары чисел — тут начинается вложенный цикл (его первый запуск)

0 0

0 1

0 2

0 3

0 4

end — вложенный цикл закончился, и после него выполнился print(‘end’)

Итерация второго внешнего цикла

start

пары чисел — тут начинается вложенный цикл (его первый запуск)

1 0 (обратите внимание, что первое число теперь равно 1, так как это вторая итерация

внешнего цикла)

1 1

1 2

1 3

1 4

end — вложенный цикл закончился, и после него выполнился print(‘end’)

Таким образом, нам видно, что вложенный цикл отработал полностью два раза (по

одному разу на каждую итерацию внешнего цикла).

**Условные блоки во вложенных циклах**

Выше указано, что вложенный цикл полностью отрабатывает на одной итерации

внешнего цикла. Это так же значит, что всё это время, пока работает вложенный цикл,

переменная внешнего цикла не изменяется.

Это позволяет выполнять во вложенном цикле разные действия в зависимости от

переменной внешнего цикла.

for i in range(4):

print(‘start’)

for j in range(2):

if i < 2:

print(‘hello’)

else:

print(‘bye’)

print(‘end’)

На первых двух итерациях вложенный цикл будет два раза повторять hello, на

последующих — по два раза повторять bye.

start

hello

hello

end

start

hello

hello

end

start

bye

bye

end

start

bye

bye

end

**Изменение количества итераций во вложенном цикле**

Работая со вложенными циклами, программист получает доступ к нескольким

переменным цикла. Важно при этом понимать, чему равна каждая из них на

протяжении всего времени работы циклов.

Особенно это важно, если нужно связать переменные циклов между собой.

Например, можно влиять на количество итераций вложенного цикла в зависимости от

переменной цикла внешнего цикла.

for i in range(3):

print(‘start’)

for j in range(i):

print(‘hello’)

print(‘end’)

Мы увидим:

start — внешняя переменная цикла равна 0, значит, вложенный цикл просто не будет

запущен (0 итераций)

end

start — тут внешняя переменная уже равна 1, значит, одна итерация вложенного цикла

будет выполнена

hello

end

start

hello — теперь переменная внешнего цикла равна 2, и у нас появилось две надписи

hello

hello

end

**Оператор else для цикла FOR**

Оператор else может использоваться не только для условного блока if/else. Он также

может быть использован для циклов, в этом случае код внутри блока else сработает,

если цикл был завершен самостоятельно, то есть без применения break.

**Синтаксис**

for <переменная цикла> in <последовательность>:

<тело цикла>

else:

<код>

Обращайте внимание на отступы — именно они позволят Python понять, что else вы

применяете к циклу for, так как они находятся на одном уровне отступа.

**Не допускай следующих ошибок!**

Не забывайте, что вложенный цикл — это просто блок кода, который выполняется, как

и весь остальной код, внутри цикла. Помните о том, что код выполняется сверху вниз

(строчка за строчкой).

Это значит, что на каждую итерацию внешнего цикла будет полностью выполнен код

внутри, в том числе полностью отработает вложенный цикл от начала до конца!\_\_

## Практическая работа к модулю №10

**print('Задача 1. Тестовое задание')**

# Степан пришёл устраиваться на работу, где ему дали тестовое задание:

# проанализировать такую таблицу,

# понять как она строится и написать программу для вывода её на экран.

# 0 2 4 6 8 10

# 1 3 5 7 9 11

# 2 4 6 8 10 12

# 3 5 7 9 11 13

# 4 6 8 10 12 14

# 5 7 9 11 13 15

# Помогите Степану реализовать такую программу.

# Подсказка: номера столбцов. А ещё не забудьте про литерал \t для табуляции

for y in range(6):

for x in range(0, 11, 2):

print(y + x, end = '\t')

print()

**print('Задача 2. Лестница')**

# Пользователь вводит число N.

# Напишите программу, которая выводит такую “лесенку” из чисел:

# Введите число: 5

# 1

# 2 2

# 3 3 3

# 4 4 4 4

# 5 5 5 5 5

n = int(input('Введите число: ')) #5

for row in range(1, n + 1):

for col in range(1 - row, 1):

print(row, end='\t')

print()

**print('Задача 3. Рамка')**

# Напишите программу,

# которая рисует с помощью символьной графики прямоугольную рамку.

# Для вертикальных линий используйте символ вертикального штриха “|”,

# а для горизонтальных - дефис “-”. Пусть пользователь вводит ширину и высоту рамки.

# \_ \_ \_ \_ \_ \_ \_ \_ \_

# | |

# | |

# | |

# | |

# | |

# | |

# |\_ \_ \_ \_ \_ \_ \_ \_ \_|

x\_line = int(input('Введите высоту рамки: ')) #9

y\_line = int(input('Введите ширину рамки: ')) #8

for row in range(x\_line + 1):

for col in range(y\_line + 1):

if row == 0 or row == x\_line:

print('-', end = ' ')

elif col == 0 or col == y\_line:

print('|', end = ' ')

else:

print(' ', end = ' ')

print()

**print('Задача 4. Крест')**

# Напишите программу,

# которая выводит на экран крест из символов “^”.

# (Символы выводятся по диагоналям воображаемого квадрата.)

# ^ ^

# ^ ^

# ^ ^

# ^ ^

# ^^

# ^^

# ^ ^

# ^ ^

# ^ ^

# ^ ^

x = 50

y = 30

delta = 3

for row in range(y):

for col in range(x):

if x == x // 2 - (delta - y):

print('/', end'')

print()

**print('Задача 5. Простые числа')**

#Напишите программу,

#которая считает количество простых чисел в заданной последовательности

#и выводит ответ на экран.

seq\_num = int(input('Сколько чисел будет всего: '))

print()

count = 0

prime = 0

for num in range(1, seq\_num + 1):

print('Введите', num, '-е число: ', end = '')

number = int(input(''))

prime = True

for i in range(num):

if num % 10 == i:

prime = False

if prime:

count += 1

print('Это простое число:', number, end = '\n')

print()

print('Простых чисел было:', count)

**print('Задача 6. Сумма факториалов')**

# Напишите программу,

# которая запрашивает у пользователя число N

# и находит сумму факториалов 1! + 2! + 3! +... + N!

n = int(input('Введите число N: '))

summ = 0

for i in range(1, n + 1):

summ += i \* n

print('Сумма факториалов равна:', summ, '.')

**print('Задача 7. Наибольшая сумма цифр')**

# Вводится N чисел.

# Среди натуральных чисел, которые были введены,

# найдите наибольшее по сумме цифр. Выведите на экран это число и сумму его цифр.

seq\_num = int(input('Введите число N чисел: '))

summ = 0

max\_num = 0

max\_sum = 0

for num in range(1, seq\_num + 1):

print('Введите', num, '-е число :', end = '')

number = int(input(''))

backup\_num = number

while number > 0:

summ += number % 10

number //= 10

if summ > max\_sum:

max\_sum = summ

max\_num = backup\_num

summ = 0

print('Число', max\_num, 'имеет наибольшую сумму цифр:', max\_sum)

**print('Задача 8. Пирамидка')**

# Напишите программу,

# которая выводит на экран равнобедренный треугольник (пирамидку),

# заполненный символами хэштега "#". Пусть высота пирамиды вводится пользователем.

# Подсказка: вспомните, как выводился колонтитул вида -----!!!!!!-----

#

###

#####

#######

symbol = int(input('Введите высоту пирамидки: '))

aggregate = 1

for i in range(symbol):

print(' ' \* (symbol - 1 - i), '#' \* aggregate)

aggregate += 2

**print('Задача 9. Пирамидка 2')**

# Напишите программу,

# которая получает на вход количество уровней пирамиды и выводит их на экран,

# Пример:

#

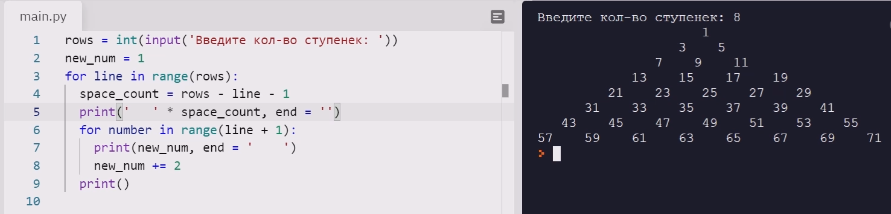
# 1

# 3 5

# 7 9 11

# 13 15 17 19

# 21 23 25 27 29

level = int(input('Введите количество уровней пирамиды: '))

number = 1

for row in range(1, level + 1):

print('\t' \* (level - row), end = '')

for col in range(row):

print(number, end = '')

number += 2

print('\t' \* 2, end = '')

print()

**print('Задача 10. Яма ')**

# В одной компьютерной текстовой игре рисуются всяческие элементы ландшафта.

#

# Напишите программу,

# которая получает на вход число N и выводит на экран числа в виде “ямы”:

# Введите число: 5

#

# 5........5

# 54......45

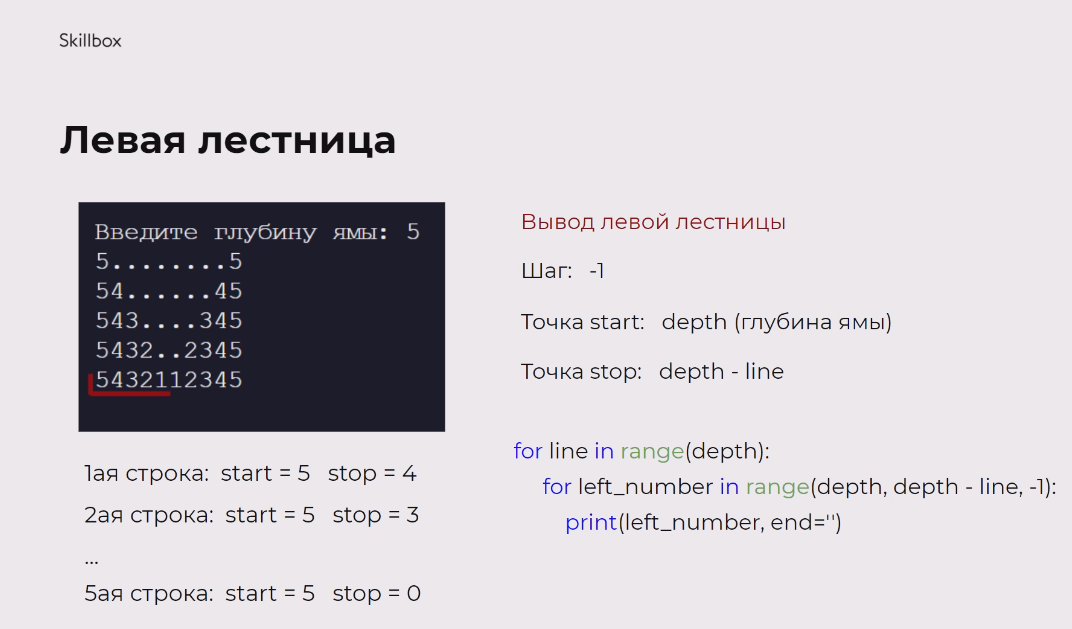
# 543....345

# 5432..2345

# 5432112345

n = int(input('Введите число: ')) #5

print()

for row in range(n, 0, -1):

for col in range(n, 0, -1):

if row <= col:

print(col, end=' ')

else:

print('.', end=' ')

for col in range(1, n + 1):

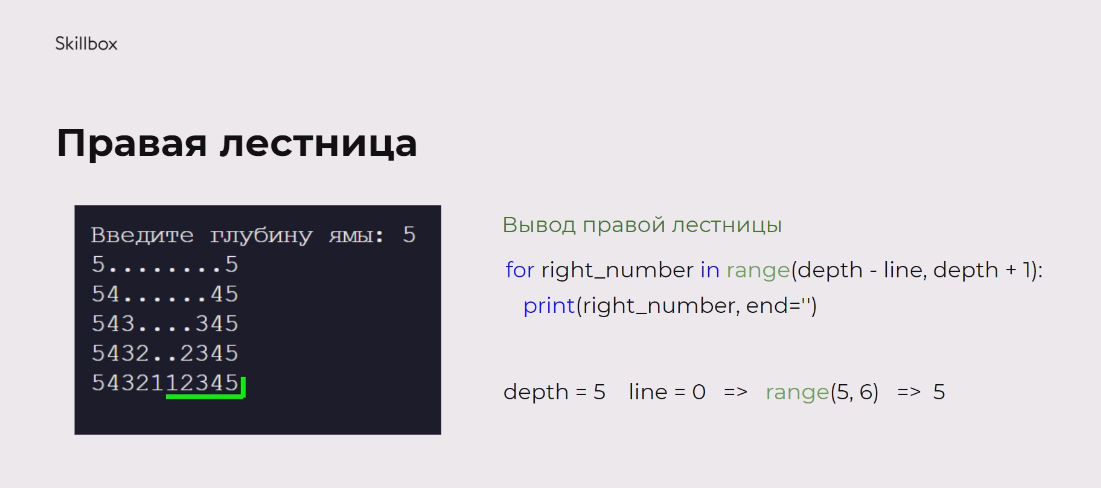
if row <= col:

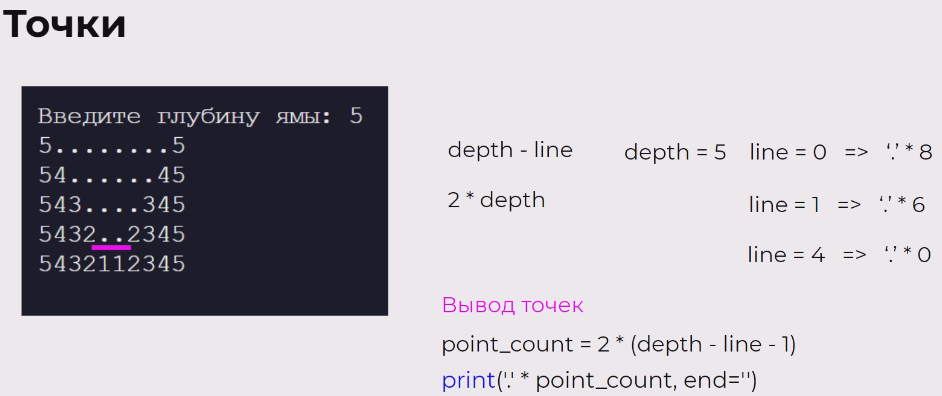
print(col, end=' ')

else:

print('.', end=' ')

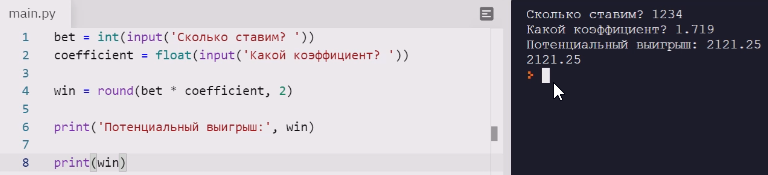
print()





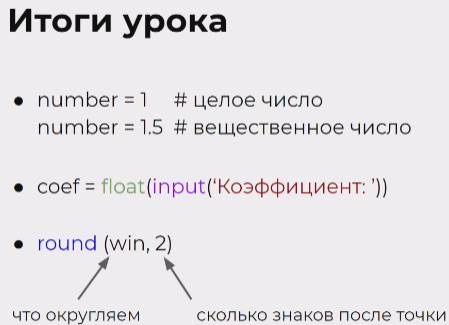
# Модуль\_11

## 11.2 Ввод вещественного числа. Функции float и round



Индекс массы тела.





#### **Практика**

##### **Задача 1. Ставки на спорт**

Нас наняла букмекерская контора, где проводятся ставки на конный спорт. Напишите программу расчёта потенциального выигрыша игрока. Для этого вводится его ставка в рублях и коэффициент (вещественное число), а выводится их произведение в качестве потенциального выигрыша.

Пример:

Сколько ставим? 1234  
Какой коэффициент? 1.716  
Потенциальный выигрыш: 2117.544

Усложнение задачи: сделайте так, чтобы после точки выводилось не больше двух цифр.

bet = float(input('Сколько ставим? ')) #1234

ratio = float(input('Какой коэффициент? ')) #1.716

win = bet \* ratio

print(round(win, 2))

##### **Задача 2. День рождения**

В честь дня рождения сына отец не придумал ничего лучше, кроме как подарить денег. Зато он придумал хоть и странную, но интересную формулу, по которой высчитывается сколько он подарит: возраст сына умножается на 1.5 и на его температуру тела. Остаётся только позавидовать такой фантазии.

Напишите программу, которая запрашивает у пользователя возраст (целое число) и температуру тела (вещественное число) и в результате выводит сколько отец подарит сыну денег на день рождения.

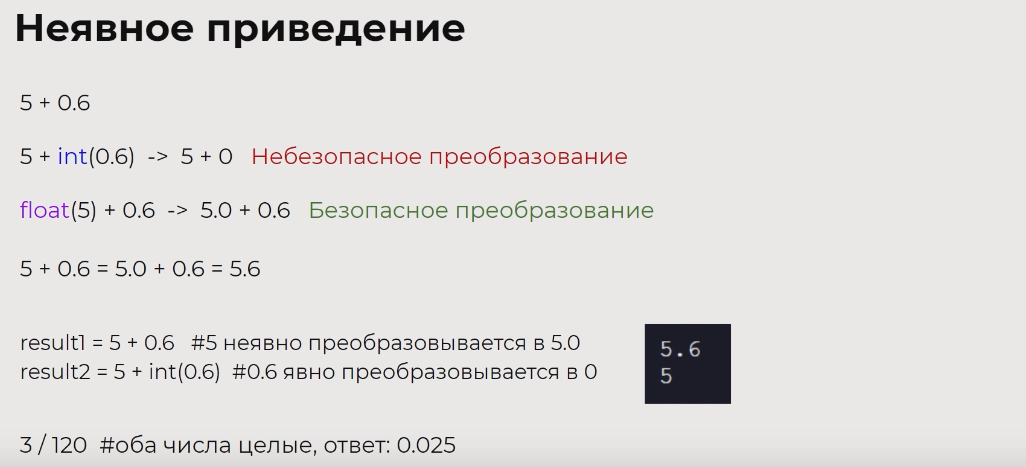
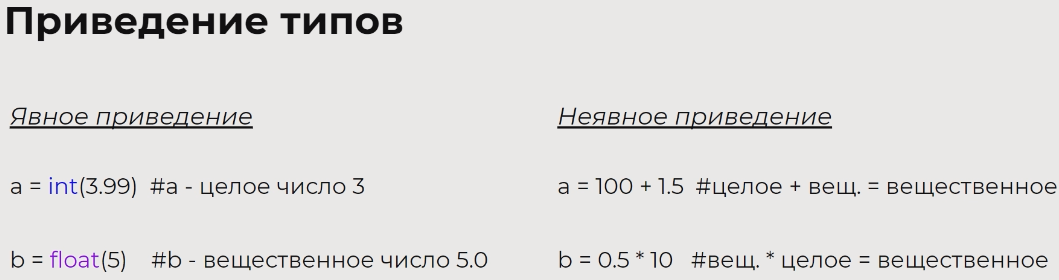
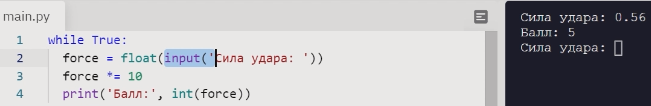
age = float(input('Какой возраст? ')) #5  
body\_temp = float(input('Какая температура? ')) #36.6  
  
money = round(age \* body\_temp, 2)  
  
print('Отец подарит сыну', money, 'руб.')

##### **Задача 3. Индекс массы тела**

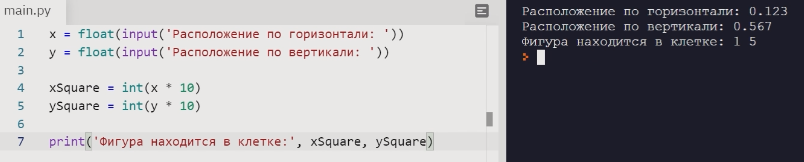
Алексей работает диетологом в частной клинике, каждый день он принимает пациентов разных возрастов и с разными показателями роста (в метрах) и веса (в кг). Для каждого человека ему нужно считать индекс массы тела - это вес поделить на рост в квадрате. По государственным стандартам индекс округляется до двух знаков после точки. Затем по этому индексу определяется, всё ли в порядке у человека с массой тела: если до 18.5, то недобор; до 25 - всё нормально, до 30 уже идёт избыток, а после 30 - ожирение. Напишите такую программу для Алексея.

weight = float(input('Введите вес: '))  
height = float(input('Введите рост: '))  
  
bmi = round(weight // (height \*\* 2), 2)  
  
print('Индекс массы тела равен:', bmi)  
  
if bmi < 18.5:  
 print('Недобор массы тела.')  
elif bmi < 25:  
 print('Все нормально).')  
elif bmi < 30:  
 print('Избыток.')  
else:  
 bmi >= 30  
 print('У вас ожирение(.')

## 11.3 Приведение типов между int и float





#### **Практика**

##### **Задача 1. Космические рейнджеры**

В далеком (а может и не очень) будущем на некоторой планете можно купить космический корабль за пол-кредита (CR). Один CR это 2200 чатлов. Причем чатлы неделимы и являются всегда целым числом. Напишите простую программу-конвертор валют. В программу вводится количество чатлов, а она сообщает сколько это CR и сколько кораблей можно купить на эту сумму.

Пример 1:

Сколько чатлов? 3150  
Это 1.4318181818181819 CR  
Можно купить кораблей: 2

chatl = int(input('Сколько чатлов? '))  
  
cr = chatl / 2200  
print('Это', cr, 'CR')  
  
ship = round(cr \* 1 // 0.5)  
print('Можно купить', ship, 'кораблей(-я)')

Пример 2:  
Сколько чатлов? 4400  
Это 2.0 CR  
Можно купить кораблей: 4

chatl = int(input('Сколько чатлов? '))  
  
cr = float(chatl / 2200)  
print('Это', cr, 'CR')  
  
ship = round(cr \* 1 // 0.5)  
print('Можно купить', ship, 'кораблей(-я)')

##### **Задача 2. Компьютерное зрение**

Вы участвуете в разработке робота, который играет в шахматы на реальной, физической шахматной доске размером 0.8 х 0.8 метра. Робот смотрит камерой на доску и видит расположение фигур в вещественных числах с очень высокой точностью.

Напишите программу, в которую вводятся вещественные координаты шахматной фигуры, а программа определяет в какой клетке доски находится эта фигура. Каждая клетка доски имеет размер 10х10 см и целочисленные координаты, состоящие из двух чисел. Например левая верхняя клетка имеет целые координаты (0, 0), справа от неё клетка (1, 0) а снизу (0, 1). Также обеспечьте **контроль ввода**: нельзя выходить за границы доски.

Пример:

Введите местоположение фигуры  
По горизонтали: 0.731  
По вертикали: 0.149

Фигура находится в клетке (7, 1)

Введите местоположение фигуры  
По горизонтали: 0.833  
По вертикали: -0.132

Клетки с такой координатой не существует

print('Введите местоположение фигуры')  
x = float(input('По горизонтали: '))  
y = float(input('По вертикали: '))  
  
# Координаты квадрата  
# cell\_x = int(x \* 10)  
# cell\_y = int(y \* 10)  
  
if 0.0 < x < 0.8 and 0.0 < y < 0.8:  
 print('Фигура находится в клетке ', '(', int(x \* 10), ',', int(y \* 10), ')')  
else:  
 print('Клетки с такой координатой не существует.')

##### **Задача 3. Точность и аккуратность**

Робот из задачи “Компьютерное зрение” правильно определяет на какой клетке стоят фигуры. Но вот беда, часто по вине соперника-человека фигуры стоят на доске не ровно по центру клетки, а со смещением. Если во время игры такое смещение станет слишком велико, то станет непонятно в какой клетке стояла фигура. Чтобы избежать этой ситуации нужно чтобы робот поправлял фигуры на доске, выставляя их по центру клетки. Модифицируйте программу “Компьютерное зрение” так, чтобы она выдавала не только номера клетки, в которой находится фигура но и две вещественных поправки: на сколько нужно подвинуть фигуру по горизонтали и вертикали для того чтобы она оказалась по центру своей клетки.

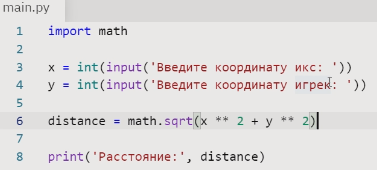
Пример:

Введите местоположение фигуры  
По горизонтали: 0.731  
По вертикали: 0.167  
Фигура находится в клетке (7, 1)

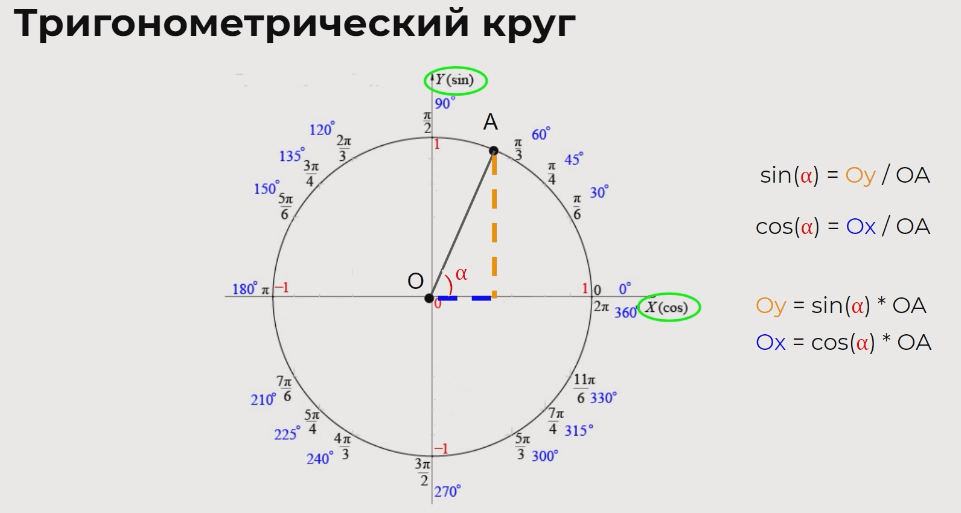
Поправьте положение фигуры на (0.019, -0.017)

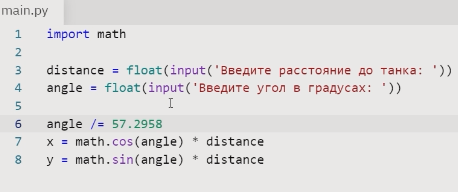
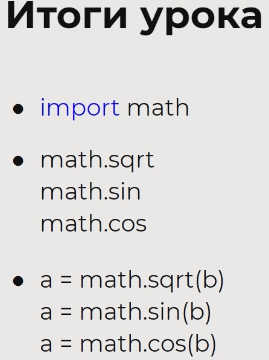
print('Введите местоположение фигуры')  
x = float(input('По горизонтали: '))  
y = float(input('По вертикали: '))  
  
if 0 < x < 0.8 and 0 < y < 0.8:  
 xSquare = int(x \* 10)  
 ySquare = int(y \* 10)  
 print('Фигура находится в клетке ', '(', xSquare, ',', ySquare, ')')  
 xCentre = xSquare / 10 + 0.05 # Координаты квадрата  
 yCentre = ySquare / 10 + 0.05 # Координаты квадрата  
 xPosition = round(xCentre - x, 3)  
 yPosition = round(yCentre - y, 3)  
 print('Поправьте положение фигуры на', '(', xPosition, ',', yPosition, ')')  
else:  
 print('Клетки с такой координатой не существует.')

## 11.4 Математические функции. Работа с модулем math







#### **Практика**

##### **Задача 1. Герон**

Существует, так называемая, формула Герона, позволяющая вычислить площадь треугольника, зная длины его сторон.

S= √ (p \* (p - a)\*(p - b)\*(p - c)) ,где

S - площадь;  
p - полупериметр треугольника (a+b+c)/2;  
a,b,c - длины сторон треугольника.

Напишите программу, которая принимает на вход длины сторон треугольника и выводит его площадь

import math  
a = float(input('Введите длину 1 стороны треугольника:'))  
b = float(input('Введите длину 2 стороны треугольника:'))  
c = float(input('Введите длину 3 стороны треугольника:'))  
  
p = (a + b + c) / 2  
s = math.sqrt(p \* (p - a) \* (p - b) \* (p - c))  
print('Площадь треугольника равна:', s)

##### **Задача 2. World of tanks**

Мы делаем программу для военного радара, который крутится вокруг своей оси и сканирует местность на наличие вражеской техники. Когда радар засекает танк неприятеля, он получает на каком расстоянии этот танк находится от нас и в каком направлении, то есть угол поворота. Генералу в штабе нужно узнать только координаты танка.

Напишите программу, которая получает на вход расстояние до танка и угол поворота. В результате выведите координаты танка.

import math  
distance = float(input('Введите растояние до танка:'))  
angle = float(input('Введите угол поворота:'))  
angle /= 57.2956 # конвертация угла из градусов в радианы  
y = math.cos(angle) \* distance # передает угол в радианах  
x = math.sin(angle) \* distance  
  
print('Координаты танка:', x, y)

##### **Задача 3. Мега-калькулятор**

Кеша учится в третьем классе, и уже умеет программировать на питоне. Как и многие его одноклассники, он очень любит сразу применять все полученные знания на практике. Вчера Кеша узнал про модуль math и его основные возможности, поэтому решил написать мега-калькулятор, который бы применял сразу все функции к введенному пользователем числу. Чтобы ничего не забыть, он пользуется шпаргалкой, которую прикрепили к уроку

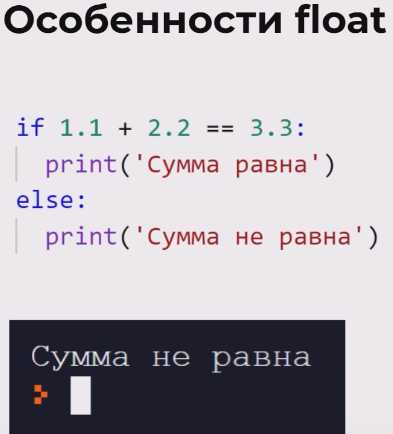
Напишите программу, которая получает от пользователя вещественное число, по очереди применяет к нему функции модуля Math и выводит результат:

* округляет вниз
* округляет вверх
* берет модуль числа
* извлекает квадратный корень
* возводит экспоненту в степень, равную числу
* считает натуральный логарифм числа
* считает логарифм числа по основанию 2
* считает логарифм числа по основанию 10
* считает синус и косинус числа

И так как Кеша самый умный в классе, он решил попробовать посчитать факториал числа. Для этого ему пришлось придумать и реализовать контроль ввода: факториал вычисляется, только если введенное число было натуральным.

import math  
number = float(input('Введите число:'))  
  
print('округляет вниз', math.floor(number))  
print()  
print('округляет вверх', math.ceil(number))  
print()  
print('берет модуль числа', abs(number))  
print()  
print('извлекает квадратный корень', math.sqrt(number))  
print()  
print('возводит экспоненту в степень, равную числу', math.exp(number))  
print()  
print('считает натуральный логарифм числа', math.log(number))  
print()  
print('считает логарифм числа по основанию 2', math.log2(number))  
print()  
print('считает логарифм числа по основанию 10', math.log10(number))  
print()  
print('считает синус числа', math.sin(number))  
print()  
print('считает косинус числа', math.cos(number))  
print()  
  
number = int(input('Введите целое число:'))  
print('считает факториал целого числа', math.factorial(number))

## 11.5 О следующем модуле



## Практическая работа к модулю №11

**Задача 1. Конвертация**

При покупках за рубежом с помощью карты банки делают конвертацию через промежуточную валюту. Например, если купить что-то в евро, то сначала эта сумма конвертируется в доллары, а уже потом - в рубли.

Напишите программу, которая получает на вход стоимость покупки в евро, затем выводит ответ в рублях. Мы живём в альтернативной реальности, где 1 евро = 1.25 доллара, а 1 доллар = 60.87 рублей.

Задание 1. Числа у которых нет имени принято называть магическими. Если другой разработчик на них посмотрит он не сможет сходу понять что это за число, тоже касается вас. Вы можете вернуться к проекту спустя длительное время и вам будет сложно понять что это за число перед вами.Поэтому записывайте такие числа в переменные и давайте им говорящие имена.

rubls = round(euro \* euro = float(input('Введите стоимость покупки в евро: '))

euro\_yuan\_exchange\_rate = 7.40

yuan\_ruble\_exchange\_rate = 10.69

euro\_yuan\_exchange\_rate \* yuan\_ruble\_exchange\_rate, 2)

print('Стоимость в рублях: ', rubls)

**print('Задача 2. Грубая математика')**

# В одном аналитическом центре,

# где занимаются разного рода математическим анализом,

# работал практикант,

# который написал программу для расчёта некоторых функций.

# Правда, он в тот день очень устал

# и немного не так прочитал техническое задание

# и функции теперь рассчитываются довольно грубо.

Вводится последовательность из N вещественных чисел.

# При этом положительные числа округляются вверх, отрицательные округляются вниз.

# Напишите программу,

# которая выводит натуральный логарифм от числа,

# если оно положительное, и экспоненту в степени числа, если оно отрицательное.

# Пример:

# Введите кол-во чисел: 3

# Введите число: 1.3

# x = 2 log(x) = 0.6931471805599453

# Введите число: -2.1

# x = -3 exp(x) = 0.049787068367863944

# Введите число: -5.9

# x = -6 exp(x) = 0.0024787521766663585

import math

number = int(input('Введите кол-во чисел: '))

for i in range (number):

num = float(input('Введите число: '))

if num > 0:

num = math.ceil(num)

log = math.log(num)

print('x=', num, 'log(x) =', log)

elif num < 0:

num = math.floor(num)

exp = math.exp(num)

print('x=', num, 'exp(x) =', exp)

**print('Задача 3. Убийца Steam')**

# Вы пишете программу-инсталлятор для компьютерной игры.

# Пока инсталлятор скачивает обновление,

# пользователю нужно показать сколько процентов уже скачалось,

# чтобы он мог решить пойти заварить чаю, или подождать у экрана компьютера.

# Обновления игры всегда занимают разное количество мегабайт,

# да и скорость интернет-соединения у игроков разная.

# Напишите программу,

# принимающую на вход размер файла обновления в мегабайтах

# и скорость интернет соединения в мегабайтах в секунду.

# Для каждой секунды программа рассчитывает

# и выводит на экран сколько процентов от всего объема уже скачано,

# до тех пор пока не будет скачан весь объем.

# В конце программа должна показать сколько всего секунд заняло скачивание обновления.

# Обеспечьте контроль ввода.

#

# Пример:

# Укажите размер файла для скачивания: 123

# Какова скорость вашего соединения? 27

#

# Прошло 1 сек. Скачано 27 из 123 Мб (22%)

# Прошло 2 сек. Скачано 54 из 123 Мб (44%)

# Прошло 3 сек. Скачано 81 из 123 Мб (66%)

# Прошло 4 сек. Скачано 108 из 123 Мб (88%)

# Прошло 5 сек. Скачано 123 из 123 Мб (100%)

import math

file = int(input('Укажите размер файла для скачивания: ' )) # 123

speed = int(input('Какова скорость вашего соединения? ' )) # 27

print()

sec = 0

for volume in range(speed, file + speed, speed):

sec += 1

percent = math.ceil(volume \* 100 / file)

if volume >= file or percent >= 100:

volume = file

percent = 100

print('Прошло', sec, 'сек.', 'Скачано', volume, 'из', file, 'Мб', "(" + str(percent) + "%)")

**print('Задача 4. Первая цифра')**

# Дано положительное действительное число X.

# Выведите его первую цифру после десятичной точки.

# При решении этой задачи нельзя пользоваться условной инструкцией, циклом или строками

num = float(input('Введите X: '))

print('Первая цифра после десятичной точки равна:', int(num \* 10) % 10)

**print('Задача 5. Вот это объёмы!')**

# Для курсовой работы по физике

# Андрею нужно сравнить объёмы двух планет: Земли и какой-нибудь случайной,

# которая может в теории существовать в нашей вселенной.

# Андрей хорошо разбирается в формулах,

# а вот считать что-то, а уж тем более самому - это явно не его.

# Объём Земли ему известен заранее - это 10.8321 \* 10 \*\* 11 км3

# А вот объём случайной планеты ему нужно будет посчитать.

# Благо, у него есть формула

#

# V = 4/3 πR \*\* 3

#

# где V - это объём, π - число пи, а R - радиус планеты.

#

# Напишите программу,

# которая получает на вход радиус случайной планеты

# и выводит на экран во сколько раз планета Земля меньше или больше по объёму.

# Ответ округлите до трёх знаков после запятой

# Пример:

# Введите радиус случайной планеты: 3389.5

# Объём планеты Земля больше в 6.641 раз

# Пример 2:

# Введите радиус случайной планеты: 7000

# Объём планеты Земля меньше в (1/0.754) = 1.326 раз

import math

RadiusXPlanet = float(input('Введите радиус случайной планеты: '))

earth = 10.8321 \* 10 \*\* 11

XPlanet = (4 / 3 \* math.pi \* RadiusXPlanet \*\* 3)

if XPlanet < earth:

difference = round(1 / (XPlanet \* 1 / earth), 3)

print('Объём планеты Земля больше в', difference, 'раз')

elif XPlanet > earth:

difference = round(1 / (XPlanet \* 1 / earth), 3)

print('Объём планеты Земля меньше в (1/0,754) =', round(1 / difference, 3), 'раз')

else:

XPlanet == earth

print('Объём планеты Земля равен объему случайной планеты.')

**print('Задача 6. Метеостанция')**

# Для удобства работы сотрудников международной метеостанции

# каждый день нужно распечатывать различные таблицы

# соответствия градусов по шкале Цельсия и Фаренгейта.

# Напишите программу,

# которая принимает на вход три целых числа

# в градусах Цельсия: нижняя граница температуры, верхняя граница температуры и шаг.

# Программа выводит на экран

# таблицу соответствия градусов Цельсия градусам Фаренгейта

# от нижней до верхней границы с указанным шагом.

# Обеспечьте контроль ввода.

# Верхняя граница должна печататься, даже если последний шаг “перепрыгнул “ ее.

# Известно, что 0С соответствует 32F,

# а каждый градус Цельсия эквивалентен 1.8 градусам Фаренгейта.

# Пример:

# Ввод:

# Нижняя граница: 0

# Верхняя граница: 50

# Шаг: 20

#

# Вывод:

# C F

# 0 32

# 20 68

# 40 104

# 50 122

down = int(input('Нижняя граница: '))

up = int(input('Верхняя граница: '))

step = int(input('Шаг: '))

print('Вывод:')

print('C', ' ', ' F')

print()

for row in range(down, up + step, step):

if row >= up:

row = up

downCol = round(32 + (1.8 \* row))

print(row, ' ', downCol)

**print('Задача 7. Ход конём')**

# В рамках разработки шахматного ИИ стоит новая задача.

# По заданным вещественным координатам коня

# и второй точки программа должна определить может ли конь ходить в эту точку.

# Используйте как можно меньше конструкций if и логических операторов.

# Обеспечьте контроль ввода.

# Введите местоположение коня:

# 0.071

# 0.118

# Введите местоположение точки на доске:

# 0.213

# 0.068

# Конь в клетке (0, 1). Точка в клетке (2, 0).

# Да, конь может ходить в эту точку.

print('Введите местоположение коня:')

xHorse = float(input())

yHorse = float(input())

print('Введите местоположение на доске:')

xPoint = float(input())

yPoint = float(input())

if (0 < xHorse < 0.8 and 0 < yHorse < 0.8) and (0 < xPoint < 0.8 and 0 < yPoint < 0.8):

xHorseM = int(xHorse \* 10)

yHorseM = int(yHorse \* 10)

xPointM = int(xPoint \* 10)

yPointM = int(yPoint \* 10)

print(f'Конь в клетке {xHorseM, yHorseM}. Точка в клетке {xPointM, yPointM}')

print('Да, конь может ходить в эту точку.')

else:

print('Ход не верен. Пожалуйста попробуйте еще раз.')

**print('Задача 8. Часы')**

# С начала суток часовая стрелка повернулась на угол в α градусов.

# Определите на какой угол повернулась минутная стрелка с начала последнего часа.

# Входные и выходные данные — действительные числа.

#

# При решении этой задачи нельзя пользоваться условными операторами и циклами.

a = int(input('Введите угол поворота часовой стрелки: '))

print(f'Угол поворота минутной стрелки равен:{((a % 30) \* 12)}°')

**print('Задача 9. Уравнение')**

# Даны действительные коэффициенты a, b, c,

# при этом a≠0.

# Решите квадратное уравнение ax^2+bx+c=0 и выведите все его корни.

#

# Если уравнение имеет два корня,

# выведите два корня в порядке возрастания,

# если один корень — выведите одно число,

# если нет корней — не выводите ничего

a = float(input('Введите первый коэффициент: '))

b = float(input('Введите второй коэффициент: '))

c = float(input('Введите третий коэффициент: '))

import math

if a != 0:

d = b \*\* 2 - (4 \* a \* c)

if d == 0:

x = -b / (2 \* a)

print(f'Один корень {x}')

elif d > 0:

x1 = float(((-b) - math.sqrt(d)) / (2 \* a))

x2 = float(((-b) + math.sqrt(d)) / (2 \* a))

print(f'Два корня: {x1} и {x2}')

else:

print('Корней нет')

**print('Задача 10. За что?')**

# Вы встретились со своим старым другом,

# который тоже изучает программирование, правда, в другом учебном заведении.

# За чашкой кофе он пожаловался вам,

# что сумасбродный препод дал им задание написать программу,

# которая из двух введённых чисел определяет наибольшее,

# не используя при этом условных операторов, циклов и встроенную функцию max.

#

# Радуясь, что на вашем курсе такого не требуют,

# вы всё-таки решаете помочь своему другу.

#

# Напишите для него эту программу.

#

# Пример:

#

# Введите первое число: 10

# Введите второе число: 5

#

# Наибольшее число: 10

firstNum = int(input('Введите первое число: '))

secondNum = int(input('Введите второе число: '))

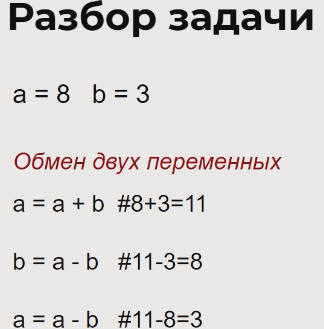
myList = [firstNum, secondNum]

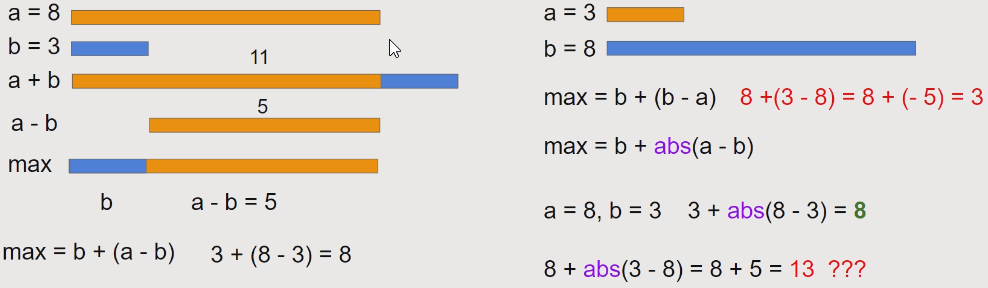
myList.sort()

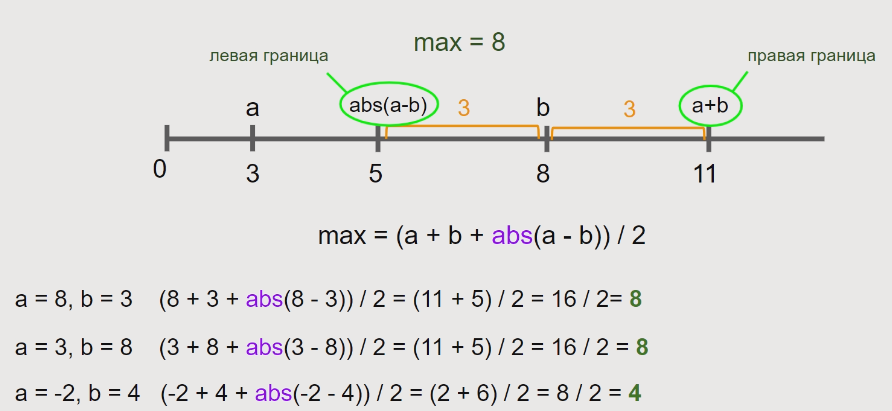
print()

print(f'’Наибольшее число: {myList[1]}')

Разбор задачи № 10







# Модуль\_12

Функции в Python для начинающих

В этой части мы изучим функции — составные инструкции, которые могут принимать данные ввода, выполнять указания и возвращать данные вывода. Функции позволяют определять и повторно использовать определенную функциональность в компактной форме.

Вызвать функцию — значит передать ей входные данные, необходимые для выполнения и возвращения результата. Когда вы передаете функции входные данные, это называется передача параметра функции.

Функции в Python похожи на математические функции из алгебры. Например, в алгебре функция определяется как‑то так:

f(x) = x \* 2

Левая часть определяет функцию *f*, принимающую один параметр, *x*. А правая часть — это определение функции, которое использует переданный параметр *x*, чтобы произвести вычисление и вернуть результат. В этом случае значением функции является ее параметр, умноженный на два.

Как в Python функция записывается следующим образом: *имя\_функции* (*параметры\_через\_запятую*). Чтобы вызвать функцию, после ее имени нужно указать круглые скобки и поместить внутрь параметры, отделив каждый из них запятой. Для создания функций в Python выберите ее имя, определите параметры, укажите, что функция должна делать и какое значение возвращать.

def имя\\_функции(параметры) :

определениие \_функции

Математическая функция *f(x) = x \* 2* в Python будет выглядеть вот так:

**def** **f**(x): **return** x \* 2

Ключевое слово *def* сообщает Python, что вы определяете функцию. После *def* вы указываете имя функции; оно должно отвечать тем же правилам, что и имена переменных. Согласно конвенции, в имени функции нельзя использовать заглавные буквы, а слова должны быть разделены подчеркиванием вот\_так.

Как только вы присвоили своей функции имя, укажите после него круглые скобки. Внутри скобок должен содержаться один или несколько параметров.

После скобок ставится двоеточие, а новая строка начинается с отступа в четыре пробела. Любой код с отступом в четыре пробела после двоеточия является телом функции. В этом случае тело нашей функции состоит только из одной строки:

**return** x \* 2

Ключевое слово *return* используется для определения значения, которое функция возвращает при вызове.

Чтобы вызвать функцию в Python, мы используем синтаксис *имя\_функции (параметры, через, запятую)*.

Ниже описан вызов функции *f* из предыдущего примера с параметром 2.

f(2)

Консоль ничего не вывела. Можно сохранить вывод вашей функции в переменной и передать ее функции *print*.

# Продолжение# предыдущего примера**def** **f**(x): **return** x \* 2 result = f(2) print(result) # 4

Вы можете сохранить результат, возвращаемый вашей функцией, в переменной и использовать это значение в программе позднее.

**def** **f**(x): **return** x + 1 z = f(4) **if** z == 5: print("z равно 5") **else**: print ("z не равно 5")

У функции может быть один параметр, несколько параметров или вообще их не быть. Чтобы определить функцию, не требующую параметров, оставьте круглые скобки пустыми.

**def** **f**(): **return** 1 + 1 result = f() print(result) # 2

Если хотите, чтобы функция принимала больше одного параметра, отделите каждый параметр в скобках запятой.

**def** **f**(x, y, z): **return** x + y + z result = f(1, 2, 3) print(result) # 6

Наконец, функция не обязана содержать инструкцию return. Если функции нечего возвращать, она возвращает значение *None*.

**def** **f**(): z = 1 + 1 result = f() print (result) # None

**Обязательные и необязательные параметры**

Функция может принимать параметры двух типов. Те, что встречались вам до этого, называются обязательными параметрами. Когда пользователь вызывает функцию, он должен передать в нее все обязательные параметры, иначе Python сгенерирует исключение.

В Python есть и другой вид параметров — опциональные. Опциональные параметры определяются с помощью следующего синтаксиса: имя\_функции(имя\_параметра = значение\_параметра). Как и обязательные, опциональные параметры нужно отделять запятыми. Ниже приведен пример функции, в коде которой используется опциональный параметр.

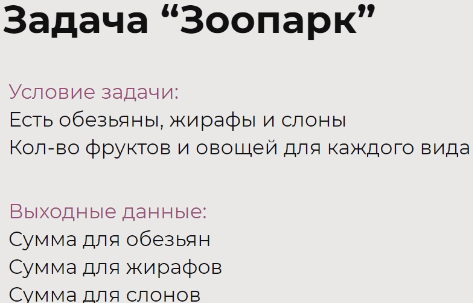
**def** **f**(x=2): **return** x\*\*x print (f()) # 4print (f(4)) # 16

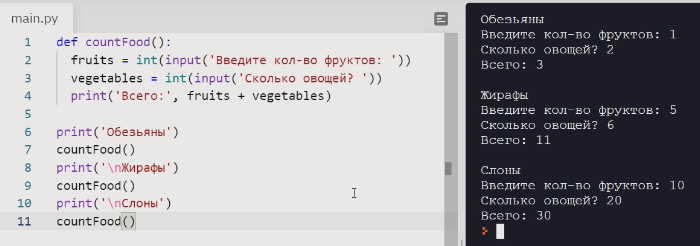
Сначала функция вызывается без передачи параметра. Так как параметр необязательный, *x* автоматически становится равен 2, и функция возвращает 4.

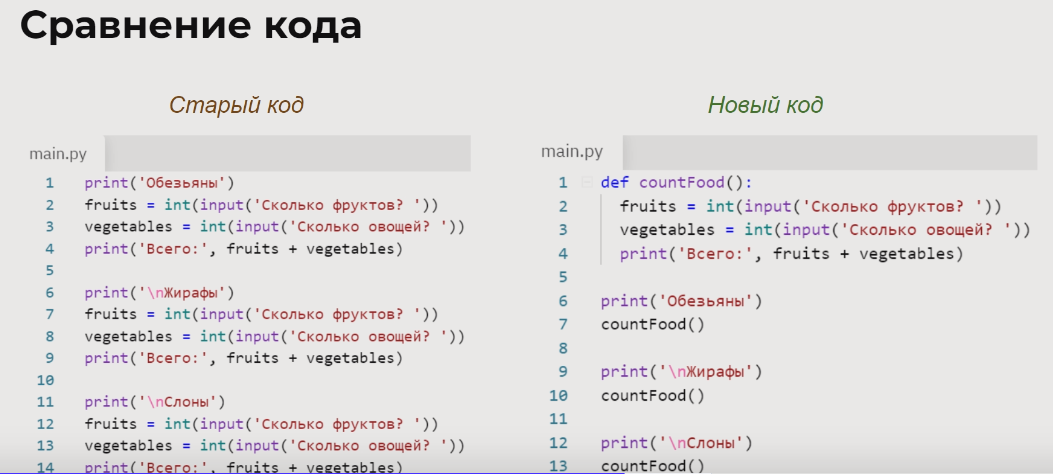
Затем та же функция вызывается с параметром 4. То есть *x* будет равен 4 и функция вернет 16. Вы можете определить функцию, которая принимает как обязательные, так и опциональные параметры, но обязательные нужно определять в первую очередь.

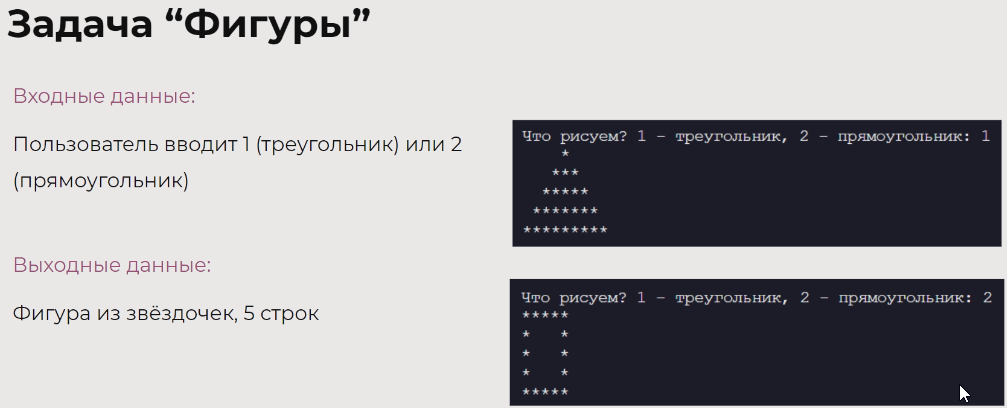
**def** **add**(x, y=10): **return** x + y result = add(2) print(result)

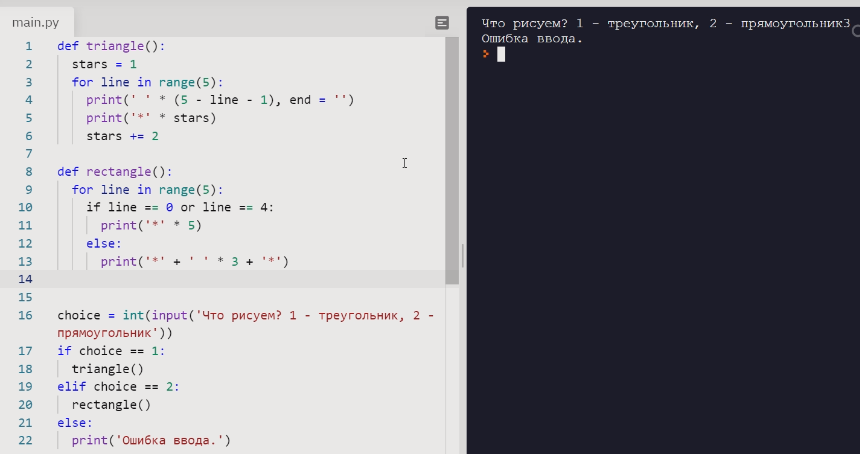
## 12.2 Функции и их вызов













#### **Практика**

##### **Задача 1. Робот**

В одном отеле для эксперимента на вход хотят поставить робота, который будет спрашивать у прохожих, не желают ли они зайти. При ответе «Да» робот должен приветствовать человека и пустить в отель. Для робота написали вот такую программу с использованием функции:

Пример результата:

Зайдёте? Да/Нет: Да

Привет!

Добро пожаловать!

Следующий.

def greeting():  
 print('Привет!')  
 print('Добро пожаловать!')  
  
while True:  
 a = input('Зайдёте? Да/Нет: ')  
 if a == 'Да':  
 greeting()  
 print('Следующий.\n')

##### **Задача 2. Провизия**

Одна государственная компания поставляет еду на разные труднодоступные базы (полярные, горные и так далее) в разных уголках страны. В компании для удобства расчёта количества еды была реализована такая программа:

def fish\_meat():

a = int(input())

b = int(input())

print("Всего", a + b, "шт.")

def bread():

a = int(input())

b = int(input())

print("Всего", a + b, "шт.")

def water\_milk():

a = int(input())

b = int(input())

print("Всего", a + b, "шт.")

print("Сколько мешков рыбы и мяса?")

fish\_meat()

print("Сколько буханок белого и чёрного хлеба?")

bread()

print("Сколько вёдер воды и молока?")

water\_milk()

##### **Задача 3. Почта**

Василий пришёл получать посылку на почту. Разумеется, в почтовом извещении ему нужно было написать свои фамилию, имя и адрес проживания, чтобы кто-нибудь не получил посылку за него, например члены его семьи (а им бы хотелось!).

Напишите функцию для вывода фамилии, имени и адреса для конкретного члена семьи. Выведите информацию о нём три раза (без цикла).

Пример результата:

Фамилия: Иванов

Имя: Василий

Улица: Пушкина

Дом: 32

def data():

surname = input('Фамилия:')

name = input('Имя:')

street = input('Улица:')

home = input('Дом:')

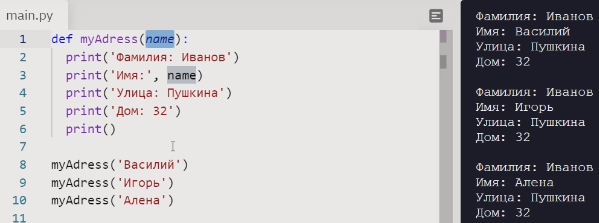
print()

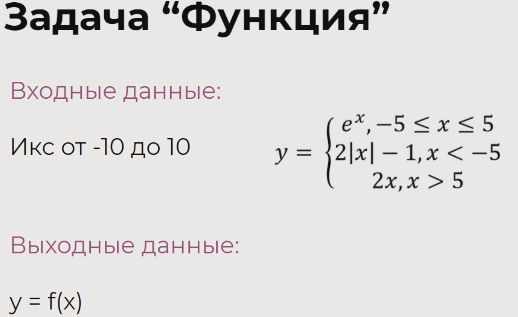
data()

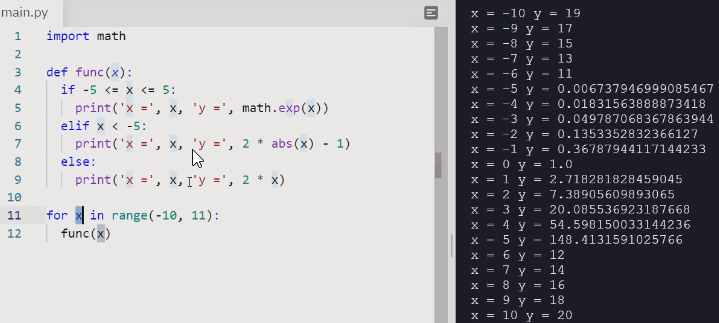
data()

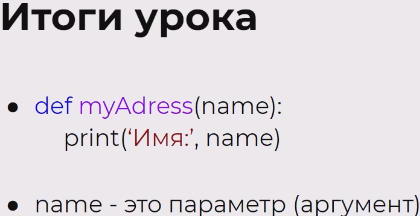
data()

## 12.3 Функции с одним параметром









#### **Практика**

##### **Задача 1. Вода**

Одна бутылка воды «КлирВотер» от производителя «ВодЗавод» в разных магазинах стоит по-разному.

Напишите программу, которая три раза вызывает функцию aboutWater, передаёт в неё один аргумент — цену на воду и выводит на экран название воды, производителя и цену.

**Пример:**

Название: КлирВотер

Производитель: ВодЗавод

Цена: 25

Название: КлирВотер

Производитель: ВодЗавод

Цена: 30

Название: КлирВотер

Производитель: ВодЗавод

Цена: 40

Решение:

def aboutWater(price):

print('Название: КлирВотер')

print('Производитель: ВодЗавод')

print('Цена:', price)

aboutWater('25')

aboutWater('30')

aboutWater('40')

**2 вариант**

import math

def about\_water(price):

print("Название: КлирВотер\n"

"Производитель: ВодЗавод\n"

f"Цена: {price}")

for \_ in range(3):

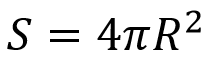
price\_of\_water = int(input("Введите цену "))

about\_water(price\_of\_water)

##### **Задача 2. Вот это объёмы 2**

Андрей продолжает писать курсовую работу по физике, и теперь ему нужно находить не только объём планеты, но и её площадь. Для этого он использует две такие формулы:

Формула для площади сферы:



Формула для объёма шара:



Так как в самом курсовом проекте эти формулы пригодятся ещё не раз, Андрей решил поступить рационально и просто написать функцию для каждой формулы.

Напишите программу, которая на вход получает от пользователя радиус планеты (вещественное число) и вызывает функции sphereArea и sphereVolume. Реализуйте эти функции: первая считает и выводит на экран площадь сферы, вторая — объём шара.

import math

def sphereArea(radius):

print(f'Площадь сферы равна: {float(2 \* math.pi \* (radius \*\* 2))} кв.км.')

def sphereVolume(radius):

print(f'Объем шара равен: {((4 / 3) \* math.pi \* (radius \*\* 2))} куб.км.')

sphereArea(6371)

sphereVolume(6371)

**2 вариант**

def sphere\_area(radius):

print(4 \* math.pi \* radius \*\* 2)

def sphere\_volume(radius):

print(4 / 3 \* math.pi \* radius \*\* 3)

radius\_of\_planet = float(input("Введите радиус планеты: "))

sphere\_area(radius\_of\_planet)

sphere\_volume(radius\_of\_planet)

##### **Задача 3. Простые числа**

Пользователь вводит число N — количество чисел в последовательности. Напишите программу, которая проверяет, сколько из этих чисел являются простыми. Для проверки простоты числа реализуйте функцию isPrime.

**1 вариант**

def is\_prime(number):

for i in range(2, int(number \*\* 0.5) + 1):

if number % i == 0:

print("Не простое")

break

else:

print("Простое")

# \*\* 0.5 - это по сути взятие корня из числа. Нам нет необходимости проверять числа дальше корня, т.к.

# проверив все множители до корня - мы проверим все возможные варианты делимости числа.

n = int(input("Введите количество чисел в последовательности: "))

for i in range(n):

new\_number = int(input("Введите число: "))

is\_prime(new\_number)

# В целом с нашими текущими знаниями этого решения достаточно. Посчитать количество придётся вручную.

**2 вариант**

# Но решим задачку и вторым вариантом с забеганием вперед:

def is\_prime(number):

for i in range(2, int(number \*\* 0.5) + 1):

if number % i == 0:

return False

return True

n = int(input("Введите количество чисел в последовательности: "))

count = 0

for i in range(n):

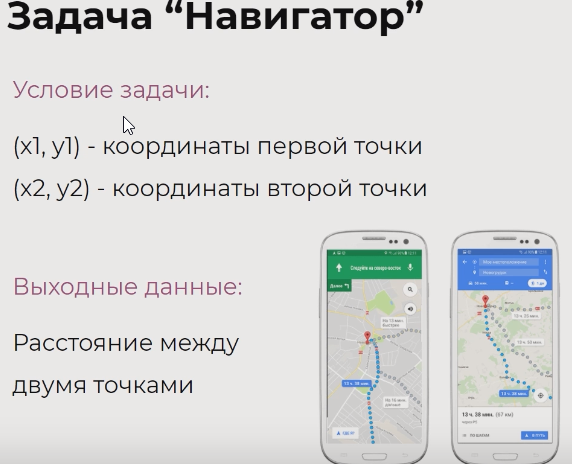
new\_number = int(input("Введите число: "))

if is\_prime(new\_number):

count += 1

print(count)

## 12.4 Функции с несколькими параметрами



#### **Практика**

##### Задача 1. Среднее арифметическое

Программа получает от пользователя два числа — a и b. Реализуйте функцию, которая принимает на вход числа a и b, считает и выводит в консоль среднее арифметическое всех чисел из отрезка [a; b]. Обеспечьте контроль ввода: не забывайте, что а всегда должно быть меньше, чем b.

Пример:

Введите левую границу: 3

Введите правую границу: 8

Среднее: 5.5

**Вариант 1**

def arithmetic\_mean(a, b):

for i in range(a, b + 1):

my\_arithmetic\_mean = (a + b) / 2

if a < b:

print('Среднее: ', my\_arithmetic\_mean)

else:

print('Ошибка ввода.')

a = int(input('Введите левую границу:'))

b = int(input('Введите правую границу:'))

arithmetic\_mean(a, b)

**Вариант 2**

Усложнение: сделайте это без использования циклов.

def arithmetic\_mean(a, b):

if a < b:

my\_arithmetic\_mean = (a + b) / 2

print('Среднее: ', my\_arithmetic\_mean)

else:

print('Ошибка ввода.')

a = int(input('Введите левую границу:'))

b = int(input('Введите правую границу:'))

arithmetic\_mean(a, b)

**или вариант от skillbox**

import math

def mean\_calc(x, y):

sum\_of\_numbers, count\_of\_numbers = 0, 0

for i in range(x, y + 1):

sum\_of\_numbers += i

count\_of\_numbers += 1

print("Среднее:", round(sum\_of\_numbers / count\_of\_numbers, 2))

# Без цикла

print("Среднее:", round((x + y) / 2, 2))

a = int(input("Введите a: "))

b = int(input("Введите b: "))

if a > b:

a, b = b, a

mean\_calc(a, b)

##### Задача 2. Почта 2

На почте немного поменялись правила: теперь в почтовом извещении нужно указывать фамилию, имя, страну проживания, город, улицу, номер дома и номер квартиры.

Реализуйте функцию, которая получает все эти данные и выводит на экран. В программе вызовите функцию три раза с разными значениями аргументов.

Подсказка: семь аргументов.

def mail(last\_name, first\_name, country\_residence, city, street, house\_number, apartment\_number):

print('Фамилия:', last\_name)

print('Имя:', first\_name)

print('Страна проживания:', country\_residence)

print('Город:', city)

print('Улица:', street)

print('Номер дома:', house\_number)

print('Номер квартиры:', apartment\_number)

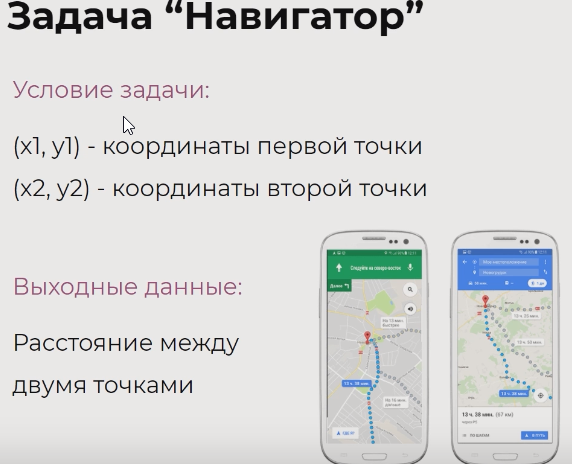
print()

mail('Иванов', 'Илья', 'Россия', 'Магадан', 'Новая', '25', '6')

mail('Петров', 'Иван', 'Китай', 'Гонкнг', 'Высокая', '125', '25')

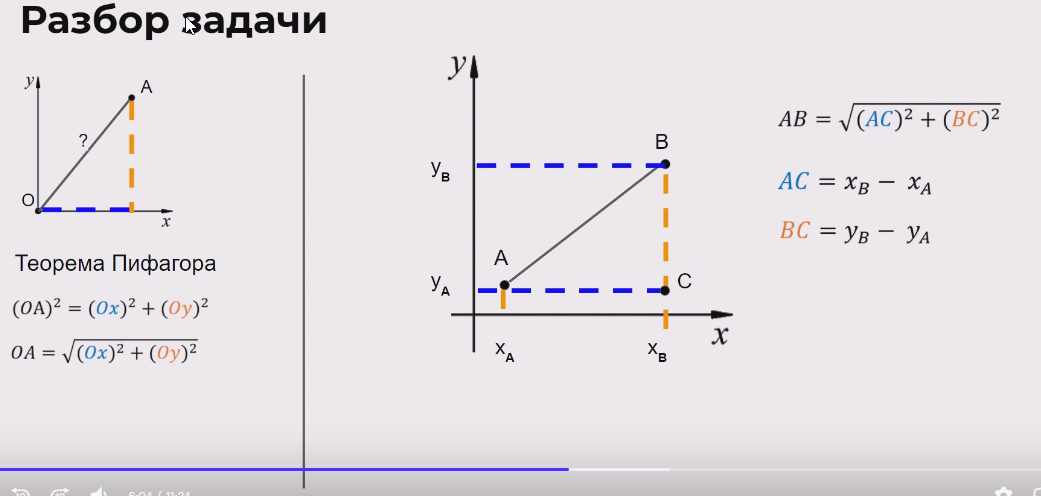
mail('Сидорова', 'Мария', 'Россия', 'Липетск', 'Маршала' 'Жукова', '23', '40')

##### Задача 3. GPS-навигатор 2.0



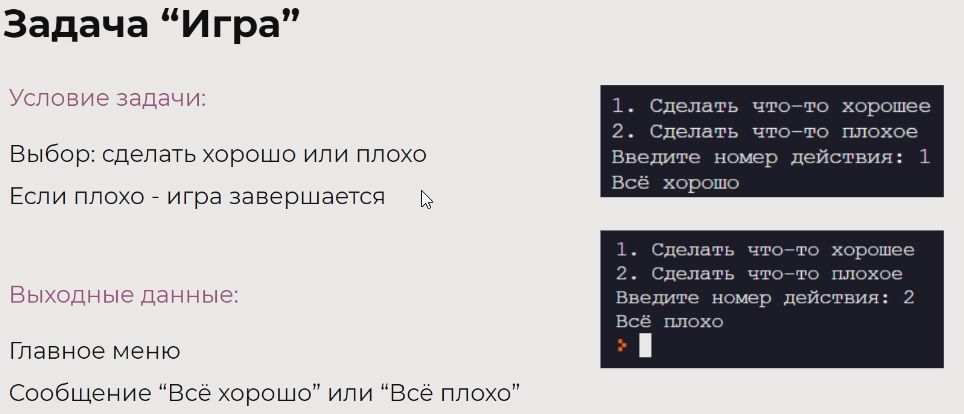
Нам поручили усовершенствовать GPS-навигатор, добавив в него новую фишку. Теперь пользователь может не только смотреть расстояние от себя до объекта, но и задавать в навигаторе две произвольные точки, после чего на экран ему выводится расстояние между ними. Для этого пользователь вводит четыре действительных числа x1, y1, x2, y2 — это как раз координаты этих двух точек.

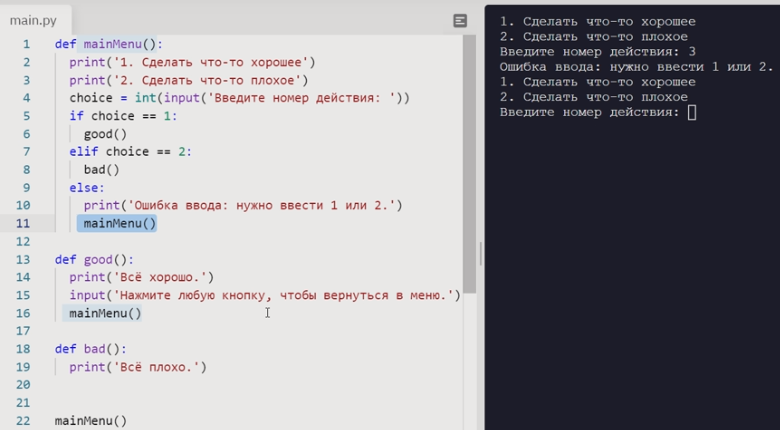
Напишите программу, где у пользователя спрашивается, чего он хочет — найти расстояние от себя до точки или найти расстояние между двумя произвольными точками, после чего запрашиваются необходимые координаты точек и выводится ответ на экран.

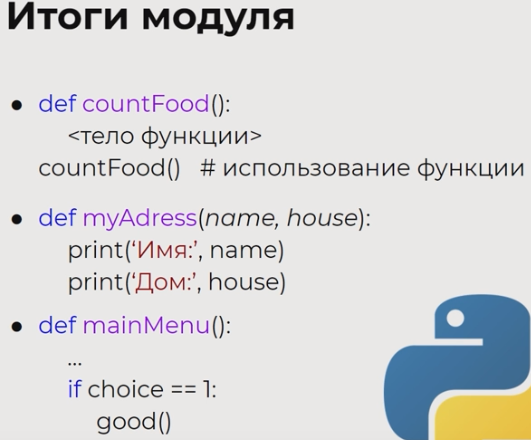


import math  
  
def myDistance(x, y):  
 distance = math.sqrt(x \*\* 2 + y \*\* 2)  
 print(distance)  
  
def betweenDistance(x1, y1, x2, y2):  
 distance = math.sqrt((x2 - x1) \*\* 2 + (y2 -y1) \*\* 2)  
 print(distance)  
  
choice = int(input('1- Расстояние от себя до точки, 2 - расстояние между двумя точками: '))  
if choice == 1:  
 x = float(input('Введите координату икс: '))  
 y = float(input('Введите координату игрек: '))  
 myDistance(x, y)  
elif choice == 2:  
 x1 = float(input('Введите координату икс первой точки: '))  
 x2 = float(input('Введите координату икс второй точки: '))  
 y1 = float(input('Введите координату игрек первой точки: '))  
 y2 = float(input('Введите координату игрек второй точки: '))  
 betweenDistance(x1, y1, x2, y2)  
else:  
 print('Ошибка ввода.')

## 12.5 Вложенный вызов функции







### Практическая работа к модулю №12

**Задача 1 Сумма чисел**

Что нужно сделать

Напишите функцию summa\_n, которая принимает одно целое положительное число N и выводит сумму всех чисел от 1 до N включительно.

Пример работы программы:

Введите число: 5

Я знаю, что сумма чисел от 1 до 5 равна 15

Что оценивается

* Результат вывода соответствует условию.
* В input содержится корректное приглашение для ввода.
* Формат вывода соответствует примеру.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

def summa\_n(numbers):

summa\_n = 0

for number in range(1, numbers + 1):

summa\_n += number

print(f'Я знаю, что сумма чисел от 1 до {numbers} равна {summa\_n}')

numbers = int(input('Введите число:'))

summa\_n(numbers)

**Задача 2. Функция в функции**

Что нужно сделать

Евгений проходит специальный тест по программированию. Всё шло хорошо, пока герой не наткнулся на тему «Функции». Задание звучит так:

Основная ветка программы, не считая заголовков функций, состоит из одной строки кода. Это вызов функции test(). В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит команду вывода на экран слова «Положительное». Если число отрицательное, то вызывается функция negative(), её тело содержит выражение вывода на экран слова «Отрицательное».

Помогите Евгению и реализуйте такую программу.

Что оценивается

* Результат вывода соответствует условию.
* В input содержится корректное приглашение для ввода.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

def test():

number = int(input('Введите целое число:'))

if number > 0:

positive()

elif number < 0:

negative()

else:

print('Ошибка ввода. Попробуйте еще раз.')

def positive():

print('Положительное')

def negative():

print('Отрицательное')

test()

**Задача 3. Апгрейд калькулятора**

Что нужно сделать

Степан использует калькулятор для расчёта суммы и разности чисел, но на работе ему требуются не только обычные арифметические действия. Он ничего не хочет делать вручную, поэтому решил немного расширить функционал калькулятора.

Напишите программу, запрашивающую у пользователя число и действие, которое нужно сделать с числом: вывести сумму его цифр, максимальную или минимальную цифру. Каждое действие оформите в виде отдельной функции, а основную программу зациклите.

Запрошенные числа должны передаваться в функции суммы, максимума и минимума при помощи аргументов.

Что оценивается

* Результат вывода соответствует условию.
* В input содержится корректное приглашение для ввода.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).
* Нет обращения к глобальным переменным снаружи функции.
* Числа, с которыми должны работать функции, передаются через аргументы.

Советы и рекомендации

* Не забывайте, что параметры функции и переменные снаружи могут иметь разные названия.
* При необходимости пересмотрите видео по работе с аргументами либо ознакомьтесь с дополнительными источниками информации по этой теме, например с этой статьёй:  
  [Функции и их аргументы в Python 3](https://pythonist.ru/funkczii-i-ih-argumenty-v-python-3/)

def mainMenu():

number = float(input('Введите число: '))

str\_number = str(number) # число в строку

str\_number = str\_number.replace('.', '') # производим замену десятичного разделителя

lst\_str = list(str\_number) # строку с числом в список строк с цифрами

lst\_num = map(int, lst\_str) # преобразовываем каждый элемент полученного списка строк с цифрами в список целых чисел

operation = int(input('Какие арифметические действия Вы хотите совершить? (1- вывести сумму цифр, 2 - вывести максимальную цифу, 3 - вывести минимальную цифру.): '))

if operation == 1:

summ(lst\_num)

elif operation == 2:

maximum(lst\_num)

elif operation == 3:

minimum(lst\_num)

else:

print('Ошибка операции. Введите 1, 2 или 3: ')

mainMenu()

def summ(lst\_num):

summ = sum(lst\_num)

print('Сумма чисел равна: ', summ)

input('Нажмите любую клавишу для возврата в основное меню: ')

mainMenu()

def maximum(lst\_num):

maximum = max(lst\_num)

print('Максимальное число равно: ', maximum)

input('Нажмите любую клавишу для возврата в основное меню: ')

mainMenu()

def minimum(lst\_num):

minimum = min(lst\_num)

print('Наименьшее число равно: ', minimum)

input('Нажмите любую клавишу для возврата в основное меню: ')

mainMenu()

mainMenu()

**Задача 4. Число наоборот**

Что нужно сделать

Вводится последовательность чисел, оканчивающаяся нулём. Реализуйте функцию, которая принимает в качестве аргумента каждое число, переворачивает его и выводит на экран.

Пример:

Введите число: 1234  
Число наоборот: 4321  
Введите число: 1000  
Число наоборот: 0001  
Введите число: 0  
Программа завершена!

Дополнительно: добейтесь такого вывода чисел, в начале которых идут нули.

Пример:

Введите число: 1230  
Число наоборот: 321

Ноль, который мы убрали, называется ведущим.

Что оценивается

* Результат вывода соответствует условию.
* Формат вывода соответствует примеру.
* В input содержится корректное приглашение для ввода.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

**Функция REVERSED() в Python, разворачивает последовательность.**

Функция reversed() возвращает обратный итератор, то есть возвращает итератор, который перебирает элементы оригинала в обратном порядке.

Перевернем список (реверс списка):

>>> x = [15, 11, 13, 12, 14, 10]

>>> x =list(reversed(x))

>>> x

*# [10, 14, 12, 13, 11, 15]*

*# теперь в обратную сторону*

>>> [i **for** i **in** reversed(x)]

*# [15, 13, 14, 11, 12, 10]*

Перевернем строку (реверс строки) с помощью **reversed()**:

x = 'forest'

**for** i **in** reversed(x):

*# вывод символов строки 'x'*

*# по одному в обратном порядке*

print(i, end='')

print('**\n**' + '-'\*len(x))

print(x)

*# tserof*

*# ------*

*# forest*

Если в итоге нужно снова получить строку, только перевернутую (запишем покороче):

>>> x = 'абракадабра'

>>> line = ''.join(reversed(list(x)))

>>> line

*# 'арбадакарба'*

**Преобразование списка в строку методом join**

Метод join в Python отвечает за объединение списка строк с помощью определенного указателя. Часто это используется при конвертации списка в строку. Например, так можно конвертировать список букв алфавита в разделенную запятыми строку для сохранения.

vowels = ["a", "e", "i", "o", "u"]

vowels\_str = ",".join(vowels)

print("Строка гласных:", vowels\_str)

Этот скрипт выдаст такой результат:

Строка гласных: a,e,i,o,u

Встроенная в Python функция map() используется для применения функции к каждому элементу итерируемого объекта (например, [списка](https://www.digitalocean.com/community/tutorials/understanding-lists-in-python-3) или [словаря](https://www.digitalocean.com/community/tutorials/understanding-dictionaries-in-python-3)) и возврата нового итератора для получения результатов. Функция map() возвращает объект map (итератор), который мы можем использовать в других частях нашей программы. Также мы можем передать объект map в функцию list() или другой тип последовательности для создания итерируемого объекта.

Функция map() имеет следующий синтаксис:

map(function, iterable, [iterable 2, iterable 3, ...])

Copy

Вместо использования [цикла for](https://www.digitalocean.com/community/tutorials/how-to-construct-for-loops-in-python-3) функция map() дает возможность применить функцию к каждому элементу итерируемого объекта. Это повышает производительность, поскольку функция применяется только к одному элементу за раз без создания копий элементов в другом итерируемом объекте. Это особенно полезно при обработке больших наборов данных. Также map() может принимать несколько итерируемых объектов в качестве аргументов функции, отправляя в функцию по одному элементу каждого итерируемого объекта за раз.

def number\_opposite(num):

list\_strings = list(str(num)) # строку с числом в список строк с цифрами

list\_strings = list(map(int, list\_strings)) # преобразовываем каждый элемент полученного списка строк с цифрами в список целых чисел

new\_number = ''

for i in range(len(list\_strings) -1, -1, -1):

new\_number += str(list\_strings[i])

print('Число наоборот:', int(new\_number))

while True:

number = int(input('Введите число: '))

if number == 0:

print('Программа завершена!')

break

elif number < 0:

number\_opposite(abs(number))

else:

number\_opposite(number)

print('')

**Задача 5. Текстовый редактор**

Что нужно сделать

Продолжаем разрабатывать новый текстовый редактор. В этот раз нам поручили написать для него код, который считает, сколько раз в тексте встречается любая выбранная буква или цифра (а не только буквы Ы, как раньше).

Напишите функцию count\_letters(), которая принимает на вход текст и подсчитывает, какое в нём количество цифр K и букв N. Функция должна вывести на экран информацию о найденных буквах и цифрах в определённом формате.

Пример:

Введите текст: 100 лет в обед  
Какую цифру ищем? 0  
Какую букву ищем? Л

Количество цифр 0: 2  
Количество букв Л: 1

Что оценивается

* Результат вывода соответствует условию.
* В input содержится корректное приглашение для ввода.
* Формат вывода соответствует примеру.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

phrase\_in = input('Введите текст: ')

count\_number = 0

count\_letter = 0

number = input('Какую цифру ищем? ')

letter = input('Какую букву ищем? ')

print()

for symbol in phrase\_in:

if symbol == number:

count\_number += 1

elif symbol == letter:

count\_letter += 1

print(f'Количество цифр {number}: {count\_number}')

print(f'Количество букв {letter}: {count\_letter}')

Подсчет суммы цифр в числе.

Допустим есть число, [целое](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-int-tselye-chisla/) или [вещественное](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-float-veschestvennye-chisla/) и необходимо [подсчитать сумму цифр](https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/funktsija-sum/) этого числа. Для того, что бы это сделать нужно это число [преобразовать в список](https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/klass-list/) входящих в него цифр, а потом применить к полученному списку функцию sum().

Алгоритм действий будет следующим:

* преобразуем число в строку при помощи [функции str()](https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/klass-str/);
* число может быть как int, так и float, следовательно необходимо произвести замену десятичного разделителя '.' при помощи [строкового метода str.replace()](https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/metod-str-replace/);
* преобразовываем полученную строку с числом в список строк с цифрами [функцией list()](https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/klass-list/);
* далее преобразовываем каждый элемент полученного списка строк с цифрами в список целых чисел используя [функцию map()](https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/funktsija-map/);
* применяем функцию sum() к итоговому списку.
* >>> pi = 3.1415926
* *# число в строку*
* >>> str\_pi = str(pi)
* *# производим замену десятичного разделителя*
* >>> str\_pi = str\_pi.replace('.', '')
* *# строку с числом в список строк с цифрами*
* >>> lst\_str = list(str\_pi)
* *# преобразовываем каждый элемент полученного*
* *# списка строк с цифрами в список целых чисел*
* >>> lst\_num = map(int, lst\_str)
* *# применяем функцию `sum()`*
* >>> s = sum(lst\_num)
* >>> s
* *# 31*

**Задача 6. НОД**

Что нужно сделать

Напишите функцию, вычисляющую наибольший общий делитель двух чисел.

Что оценивается

* Результат вывода соответствует условию.
* В input содержится корректное приглашение для ввода.
* Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

import math

first\_num = int(input('Введите первое число: '))

second\_num = int(input('Введите второе число: '))

print('НОД равен:', math.gcd(first\_num, second\_num))

**Задача 7. Недоделка**

Что нужно сделать

Вы пришли на работу в компанию по разработке игр, целевая аудитория — дети и их родители. У предыдущего программиста было задание сделать две игры в одном приложении, чтобы пользователь мог выбирать одну из них. Однако программист, на место которого вы пришли, перед увольнением не успел выполнить эту задачу и оставил только небольшой шаблон проекта. Используя этот шаблон, реализуйте игры «Камень, ножницы, бумага» и «Угадай число».

Правила игры «Камень, ножницы, бумага»: программа запрашивает у пользователя строку и выводит, победил он или проиграл. Камень бьёт ножницы, ножницы режут бумагу, бумага кроет камень.

Правила игры «Угадай число»: программа запрашивает у пользователя число до тех пор, пока он не отгадает загаданное.

def rock\_paper\_scissors():  
 # Здесь будет игра «Камень, ножницы, бумага»  
  
def guess\_the\_number():  
 # Здесь будет игра «Угадай число»  
  
def mainMenu():  
 # Здесь главное меню игры  
  
mainMenu():  
 pass

import random

def rock\_paper\_scissors():

comp = random.SystemRandom().choice(['камень', 'ножницы', 'бумага'])

user = input('Введите камень, ножницы или бумага: ')

if user == 'камень' and comp == 'ножницы':

print(f'Компютер выбрал: {comp}')

print('Вы победили')

elif user == 'камень' and comp == 'бумага':

print(f'Компютер выбрал: {comp}')

print('Вы проиграли')

elif user == 'ножницы' and comp == 'камень':

print(f'Компютер выбрал: {comp}')

print('Вы проиграли')

elif user == 'ножницы' and comp == ',бумага':

print(f'Компютер выбрал: {comp}')

print('Вы выйграли')

elif user == 'бумага' and comp == 'ножницы':

print(f'Компютер выбрал: {comp}')

print('Вы проиграли')

elif user == 'бумага' and comp == 'камень':

print(f'Компютер выбрал: {comp}')

print('Вы выйрали')

else:

print(f'Компютер выбрал: {comp}')

print('Ничья')

def guess\_the\_number():

while True:

comp = random.randint(1, 10)

user = int(input('Введите число от 1 до 10: '))

if comp != user:

print(f'Компютер загадал число: {comp}')

print('Вы не отгадали, попробуйте еще раз:')

print()

else:

print(f'Компютер загадал число: {comp}')

print('Вы отгадали. Молодец.')

print('Игра завершена.')

break

def mainMenu():

a = int(input(f'Выберите игру:\n 1) Камень.Ножницы.Бумага.\n 2) Угадай число.\n Ваш выбор: '))

if a == 1:

print()

rock\_paper\_scissors()

elif a == 2:

print()

guess\_the\_number()

else:

print('Неправильный выбор. Выберите игру.')

mainMenu()

## Список ****методов модуля random**** в Python:

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| [seed()](https://python-scripts.com/random#seed) | Инициализация генератора случайных чисел |
| [getstate()](https://python-scripts.com/random#getstate) | Возвращает текущее внутренне состояние (state) генератора случайных чисел |
| [setstate()](https://python-scripts.com/random#setstate) | Восстанавливает внутреннее состояние (state) генератора случайных чисел |
| getrandbits() | Возвращает число, которое представляет собой случайные биты |
| [randrange()](https://python-scripts.com/random#randrange) | Возвращает случайное число в пределах заданного промежутка |
| [randint()](https://python-scripts.com/random#randint) | Возвращает случайное число в пределах заданного промежутка |
| [choice()](https://python-scripts.com/random#choice) | Возвращает случайный элемент заданной последовательности |
| [choices()](https://python-scripts.com/random#choices) | Возвращает список со случайной выборкой из заданной последовательности |
| [shuffle()](https://python-scripts.com/random#shuffle) | Берет последовательность и возвращает ее в перемешанном состоянии |
| [sample()](https://python-scripts.com/random#sample) | Возвращает заданную выборку последовательности |
| random() | Возвращает случайное вещественное число в промежутке от 0 до 1 |
| [uniform()](https://python-scripts.com/random#uniform) | Возвращает случайное вещественное число в указанном промежутке |
| [triangular()](https://python-scripts.com/random#triangular) | Возвращает случайное вещественное число в промежутке между двумя заданными параметрами. Также можно использовать параметр mode для уточнения середины между указанными параметрами |
| betavariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **Бета-распределении**, которое используется в статистике |
| expovariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, или же между 0 и -1, когда параметр отрицательный. За основу берется **Экспоненциальное распределение**, которое используется в статистике |
| gammavariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **Гамма-распределении**, которое используется в статистике |
| gauss() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **Гауссовом распределении**, которое используется в теории вероятности |
| lognormvariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **Логнормальном распределении**, которое используется в теории вероятности |
| normalvariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **Нормальном распределении**, которое используется в теории вероятности |
| vonmisesvariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **распределении фон Мизеса**, которое используется в направленной статистике |
| paretovariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **распределении Парето**, которое используется в теории вероятности |
| weibullvariate() | Возвращает случайное вещественное число в промежутке между 0 и 1, основываясь на **распределении Вейбулла**, которое используется в статистике |

Цели данной статьи

Далее представлен список основных операций, которые будут описаны в руководстве:

* **Генерация случайных чисел** для различных распределений, которые включают целые и вещественные числа с плавающей запятой;
* Случайная выборка нескольких элементов последовательности population;
* Функции модуля random;
* Перемешивание элементов последовательности. Seed в генераторе случайных данных;
* Генерация случайных строки и паролей;
* Криптографическое обеспечение **безопасности генератора случайных данных** при помощи использования **модуля secrets**. Обеспечение безопасности токенов, ключей безопасности и URL;
* Способ настройки работы генератора случайных данных;
* Использование numpy.random для генерации случайных массивов;
* Использование модуля UUID для генерации уникальных ID.

В статье также даются ссылки на некоторые другие тексты сайта, связанные с рассматриваемой темой.

## Как использовать модуль random в Python

Для достижения перечисленных выше задач **модуль random** будет использовать разнообразные функции. Способы использования данных функций будут описаны в следующих разделах статьи.

В самом начале работы необходимо импортировать модуль random в программу. Только после этого его можно будет полноценно использовать. Оператор для импорта модуля random выглядит следующим образом:

Python

import random

Теперь рассмотрим использование самого модуля random на простом примере:

Python

import random

print("Вывод случайного числа при помощи использования random.random()")

print(random.random())

Вывод:

Shell

Вывод случайного числа при помощи использования random.random()

0.9461613475266107

Как видите, в результате мы получили 0.9461613475266107. У вас, конечно, выйдет другое случайно число.

random() является базовой функцией модуля random;

Почти все функции модуля random зависят от базовой функции random();

random() возвращает следующее случайное число с плавающей запятой в промежутке [0.0, 1.0].

Перед разбором функций модуля random давайте рассмотрим основные сферы их применения.

Генерация случайных чисел в Python

Давайте рассмотрим самый популярный случай использования модуля random — генерацию случайного числа. Для получения случайного целого числа в Python используется функция randint().

Для генерации случайных целых чисел можно использовать следующие две функции:

randint()

randrange()

В следующем примере показано, как получить случайно сгенерированное число в промежутке между 0 и 9.

Python

from random import randint

print("Вывод случайного целого числа ", randint(0, 9))

print("Вывод случайного целого числа ", randrange(0, 10, 2))

Вывод:

Shell

Вывод случайного целого числа 5

Вывод случайного целого числа 2

В следующих разделах статьи будут рассмотрены некоторые другие способы генерации случайного числа в Python.

Выбор случайного элемента из списка Python

Предположим, вам дан python список городов, и вы хотите вывести на экран случайно выбранный элемент из списка городов. Посмотрим, как это можно сделать:

Python

import random

city\_list = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Philadelphia']

print("Выбор случайного города из списка - ", random.choice(city\_list))

Вывод:

Shell

Выбор случайного города из списка - Houston

Python функции модуля random

Рассмотрим разнообразные функции, доступные в модуле random.

Случайное целое число — randint(a, b) модуль random

Возвращает случайное целое число Number, такое что a <= Number <= b;

randint(a,b) работает только с целыми числами;

Функция randint(a,b) принимает только два параметра, оба обязательны;

Полученное в результате случайно число больше или равно a, а также меньше или равно b.

Пример использования random.randint() в Python:

Python

import random

print("Использование random.randint() для генерации случайного целого числа")

print(random.randint(0, 5))

print(random.randint(0, 5))

Вывод:

Shell

Использование random.randint() для генерации случайного целого числа

4

2

Генерация случайного целого числа — randrange() модуль random

Метод random.randrange() используется для генерации случайного целого числа в пределах заданного промежутка. Скажем, для получения любого числа в диапазоне между 10 и 50.

Шаг показывает разницу между каждым числом заданной последовательности. Шаг по умолчанию равен 1, однако его значение можно изменить.

Пример использования random.randrange() в Python:

Python

import random

print("Генерация случайного числа в пределах заданного промежутка")

print(random.randrange(10, 50, 5))

print(random.randrange(10, 50, 5))

Вывод:

Shell

Генерация случайного числа в пределах заданного промежутка

10

15

Выбор случайного элемента из списка choice() модуль random

Метод random.choice() используется для выбора случайного элемента из списка. Набор может быть представлен в виде списка или python строки. Метод возвращает один случайный элемент последовательности.

Пример использования random.choice() в Python:

Python

import random

list = [55, 66, 77, 88, 99]

print("random.choice используется для выбора случайного элемента из списка - ", random.choice(list))

Вывод:

Shell

random.choice используется для выбора случайного элемента из списка - 55

Метод sample(population, k) из модуля random

Метод random.sample() используется, когда требуется выбрать несколько элементов из заданной последовательности population.

Метод sample() возвращает список уникальных элементов, которые были выбраны из последовательности population. Итоговое количество элементов зависит от значения k;

Значение в population может быть представлено в виде списка или любой другой последовательности.

Пример использования random.sample() в Python:

Python

import random

list = [2, 5, 8, 9, 12]

print ("random.sample() ", random.sample(list,3))

Вывод:

Shell

random.sample() [5, 12, 2]

Случайные элементы из списка — choices() модуль random

random.choices(population, weights=None, \*, cum\_weights=None, k=1)

Метод random.choices() используется, когда требуется выбрать несколько случайных элементов из заданной последовательности.

Метод choices() был введен в версии Python 3.6. Он также позволяет повторять несколько раз один и тот же элемент.

Пример использования random.choices() в Python:

Python

import random

# Выборка с заменой

list = [20, 30, 40, 50 ,60, 70, 80, 90]

sampling = random.choices(list, k=5)

print("Выборка с методом choices ", sampling)

Вывод:

Shell

Выборка с методом choices [30, 20, 40, 50, 40]

Генератор псевдослучайных чисел — seed() модуль random

Метод seed() используется для инициализации генератора псевдослучайных чисел в Python;

Модуль random использует значение из seed, или отправной точки как основу для генерации случайного числа. Если значения seed нет в наличии, тогда система будет отталкиваться от текущего времени.

Пример использования random.seed() в Python:

Python

import random

random.seed(6)

print("Случайное число с семенем ",random.random())

print("Случайное число с семенем ",random.random())

Вывод:

Shell

Random number with seed 0.793340083761663

Random number with seed 0.793340083761663

Перемешивание данных — shuffle() из модуля random

Метод random.shuffle() используется для перемешивания данных списка или другой последовательности. Метод shuffle() смешивает элементы списка на месте. Самый показательный пример использования — тасование карт.

Пример использования random.shuffle() в Python:

Python

list = [2, 5, 8, 9, 12]

random.shuffle(list)

print ("Вывод перемешанного списка ", list)

Вывод:

Shell

Вывод перемешанного списка [8, 9, 2, 12, 5]

Генерации числа с плавающей запятой — uniform() модуль random

random.uniform() используется для генерации числа с плавающей запятой в пределах заданного промежутка

Значение конечной точки может включаться в диапазон, но это не обязательно. Все зависит от округления значения числа с плавающей запятой;

Метод может, например, сгенерировать случайно вещественное число в промежутке между 10.5 и 25.5.

Пример использования random.uniform() в Python:

Python

import random

print("Число с плавающей точкой в пределах заданного промежутка")

print(random.uniform(10.5, 25.5))

Вывод:

Shell

Число с плавающей точкой в пределах заданного промежутка

22.095283175159786

triangular(low, high, mode) из модуля random

Функция random.triangular() возвращает случайное вещественное число N с плавающей запятой, которое соответствует условию lower <= N <= upper, а также уточняющему значению mode.

Значение нижнего предела по умолчанию равно нулю, в верхнего — единице. Кроме того, пик аргумента по умолчанию установлен на середине границ, что обеспечивает симметричное распределение.

Функция random.triangular() используется в генерации случайных чисел для треугольного распределения с целью использования полученных значений в симуляции. Это значит, что в при генерации значения применяется треугольное распределение вероятности.

Пример использования random.triangular() в Python:

Python

import random

print("Число с плавающей точкой через triangular")

print(random.triangular(10.5, 25.5, 5.5))

Вывод:

Shell

Число с плавающей точкой через triangular

16.7421565549115

Генератор случайной строки в Python

В данном разделе будет подробно расписано, как сгенерировать случайную строку фиксированной длины в Python.

Основные аспекты раздела:

Генерация случайной строки фиксированной длины;

Получение случайной алфавитно-цифровой строки, среди элементов которой будут как буквы, так и числа;

Генерация случайного пароля, который будет содержать буквы, цифры и специальный символы.

Криптографическая зашита генератора случайных данных в Python

Случайно сгенерированные числа и данные, полученные при помощи модуля random в Python, лишены криптографической защиты. Следовательно, возникает вопрос — как добиться надежной генерации случайных чисел?

Криптографически надежный генератор псевдослучайных чисел представляет собой генератор чисел, который обладает особенностями, что делают его подходящим для использования в криптографических приложениях, где безопасность данных имеет первостепенное значение.

Все функции криптографически надежного генератора возвращают полученные случайным образом байты;

Значение случайных байтов, полученных в результате использования функции, зависит от источников ОС.

Качество генерации также зависит от случайных источников ОС.

Для обеспечения криптографической надежности генерации случайных чисел можно использовать следующие подходы:

Применение модуля secrets для защиты случайных данных;

Использование из модуля os os.urandom();

Использование класса random.SystemRandom.

Пример криптографически надежной генерации данных в Python:

Python

import random

import secrets

number = random.SystemRandom().random()

print("Надежное число ", number)

print("Надежный токен байтов", secrets.token\_bytes(16))

Вывод:

Shell

Надежное число 0.11139538267693572

Надежный токен байтов b'\xae\xa0\x91\*.\xb6\xa1\x05=\xf7+>\r;Y\xc3'

getstate() и setstate() в генераторе случайных данных Python

Функции getstate() и setstate() модуля random позволяют зафиксировать текущее внутреннее состояние генератора.

Используя данные функции, можно сгенерировать одинаковое количество чисел или последовательностей данных.

Состояние генератора getstate() модуль random

Функция getstate() возвращает определенный объект, зафиксировав текущее внутреннее состояние генератора случайных данных. Данное состояние передается методу setstate() для восстановления полученного состояния в качестве текущего.

На заметку: Изменив значение текущего состояния на значение предыдущего, мы можем получить случайные данные вновь. Например, если вы хотите получить аналогичную выборку вновь, можно использовать данные функции.

Восстанавливает внутреннее состояние генератора — setstate() модуль random

Функция setstate() восстанавливает внутреннее состояние генератора и передает его состоянию объекта. Это значит, что вновь будет использован тот же параметр состояния state. Объект state может быть получен при помощи вызова функции getstate().

Зачем нужны функции getstate() и setstate() ?

Если вы получили предыдущее состояние и восстановили его, тогда вы сможете оперировать одними и теми же случайными данными раз за разом. Помните, что использовать другую функцию random в данном случае нельзя. Также нельзя изменить значения заданных параметров. Сделав это, вы измените значение состояния state.

Для закрепления понимания принципов работы getstate() и setstate() в генераторе случайных данных Python рассмотрим следующий пример:

Python

import random

number\_list = [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

print("Первая выборка ", random.sample(number\_list,k=5))

# хранит текущее состояние в объекте state

state = random.getstate()

print("Вторая выборка ", random.sample(number\_list,k=5))

# Восстанавливает состояние state, используя setstate

random.setstate(state)

#Теперь будет выведен тот же список второй выборки

print("Третья выборка ", random.sample(number\_list,k=5))

# Восстанавливает текущее состояние state

random.setstate(state)

# Вновь будет выведен тот же список второй выборки

print("Четвертая выборка ", random.sample(number\_list,k=5))

Вывод:

Shell

Первая выборка [18, 15, 30, 9, 6]

Вторая выборка [27, 15, 12, 9, 6]

Третья выборка [27, 15, 12, 9, 6]

Четвертая выборка [27, 15, 12, 9, 6]

Как можно заметить в результате вывода — мы получили одинаковые наборы данных. Это произошло из-за сброса генератора случайных данных.

Numpy.random — Генератор псевдослучайных чисел

PRNG является англоязычным акронимом, который расшифровывается как «pseudorandom number generator» — генератор псевдослучайных чисел. Известно, что в Python модуль random можно использовать для генерации случайных скалярных числовых значений и данных.

Для генерации массива случайных чисел необходимо использовать numpy.random();

В модуле numpy есть пакет numpy.random, который содержит обширный набор функций для генерации случайных n-мерных массивов для различных распределений.

Рассмотрим несколько примеров использования numpy.random в Python.

Генерация случайного n-мерного массива вещественных чисел

Использование numpy.random.rand() для генерации n-мерного массива случайных вещественных чисел в пределах [0.0, 1.0)

Использование numpy.random.uniform() для генерации n-мерного массива случайных вещественных чисел в пределах [low, high)

Python

import numpy

random\_float\_array = numpy.random.rand(2, 2)

print("2 X 2 массив случайных вещественных чисел в [0.0, 1.0] \n", random\_float\_array,"\n")

random\_float\_array = numpy.random.uniform(25.5, 99.5, size=(3, 2))

print("3 X 2 массив случайных вещественных чисел в [25.5, 99.5] \n", random\_float\_array,"\n")

Вывод:

Shell

2 X 2 массив случайных вещественных чисел в [0.0, 1.0]

[[0.08938593 0.89085866]

[0.47307169 0.41401363]]

3 X 2 массив случайных вещественных чисел в [25.5, 99.5]

[[55.4057854 65.60206715]

[91.62185404 84.16144062]

[44.348252 27.28381058]]

Генерация случайного n-мерного массива целых чисел

Для генерации случайного n-мерного массива целых чисел используется numpy.random.random\_integers():

Python

import numpy

random\_integer\_array = numpy.random.random\_integers(1, 10, 5)

print("1-мерный массив случайных целых чисел \n", random\_integer\_array,"\n")

random\_integer\_array = numpy.random.random\_integers(1, 10, size=(3, 2))

print("2-мерный массив случайных целых чисел \n", random\_integer\_array)

Вывод:

Shell

1-мерный массив случайных целых чисел

[10 1 4 2 1]

2-мерный массив случайных целых чисел

[[ 2 6]

[ 9 10]

[ 3 6]]

Выбор случайного элемента из массива чисел или последовательности

Использование numpy.random.choice() для генерации случайной выборки;

Использование данного метода для получения одного или нескольких случайных чисел из n-мерного массива с заменой или без нее.

Рассмотрим следующий пример:

Python

import numpy

array =[10, 20, 30, 40, 50, 20, 40]

single\_random\_choice = numpy.random.choice(array, size=1)

print("один случайный выбор из массива 1-D", single\_random\_choice)

multiple\_random\_choice = numpy.random.choice(array, size=3, replace=False)

print("несколько случайных выборов из массива 1-D без замены", multiple\_random\_choice)

multiple\_random\_choice = numpy.random.choice(array, size=3, replace=True)

print("несколько случайных выборов из массива 1-D с заменой", multiple\_random\_choice)

Вывод:

Shell

один случайный выбор из массива 1-D [40]

несколько случайных выборов из массива 1-D без замены [10 40 50]

несколько случайных выборов из массива 1-D с заменой [20 20 10]

В будущих статьях будут описаны другие функции пакета random из nympy и способы их использования.

Генерация случайных универсально уникальных ID

Модуль Python UUID предоставляет неизменяемые UUID объекты. UUID является универсально уникальным идентификатором.

У модуля есть функции для генерации всех версий UUID. Используя функцию uuid.uuid4(), можно получить случайно сгенерированное уникальное ID длиной в 128 битов, которое к тому же является криптографически надежным.

Полученные уникальные ID используются для идентификации документов, пользователей, ресурсов и любой другой информации на компьютерных системах.

Пример использования uuid.uuid4() в Python:

Python

import uuid

# получить уникальный UUID

safeId = uuid.uuid4()

print("безопасный уникальный id ", safeId)

Вывод:

Shell

безопасный уникальный id fb62463a-cd93-4f54-91ab-72a2e2697aff

Игра в кости с использованием модуля random в Python

Далее представлен код простой игры в кости, которая поможет понять принцип работы функций модуля random. В игре два участника и два кубика.

Участники по очереди бросают кубики, предварительно встряхнув их;

Алгоритм высчитывает сумму значений кубиков каждого участника и добавляет полученный результат на доску с результатами;

Участник, у которого в результате большее количество очков, выигрывает.

Код программы для игры в кости Python:

Python

import random

PlayerOne = "Анна"

PlayerTwo = "Алекс"

AnnaScore = 0

AlexScore = 0

# У каждого кубика шесть возможных значений

diceOne = [1, 2, 3, 4, 5, 6]

diceTwo = [1, 2, 3, 4, 5, 6]

def playDiceGame():

"""Оба участника, Анна и Алекс, бросают кубик, используя метод shuffle"""

for i in range(5):

#оба кубика встряхиваются 5 раз

random.shuffle(diceOne)

random.shuffle(diceTwo)

firstNumber = random.choice(diceOne) # использование метода choice для выбора случайного значения

SecondNumber = random.choice(diceTwo)

return firstNumber + SecondNumber

print("Игра в кости использует модуль random\n")

#Давайте сыграем в кости три раза

for i in range(3):

# определим, кто будет бросать кости первым

AlexTossNumber = random.randint(1, 100) # генерация случайного числа от 1 до 100, включая 100

AnnaTossNumber = random.randrange(1, 101, 1) # генерация случайного числа от 1 до 100, не включая 101

if( AlexTossNumber > AnnaTossNumber):

print("Алекс выиграл жеребьевку.")

AlexScore = playDiceGame()

AnnaScore = playDiceGame()

else:

print("Анна выиграла жеребьевку.")

AnnaScore = playDiceGame()

AlexScore = playDiceGame()

if(AlexScore > AnnaScore):

print ("Алекс выиграл игру в кости. Финальный счет Алекса:", AlexScore, "Финальный счет Анны:", AnnaScore, "\n")

else:

print("Анна выиграла игру в кости. Финальный счет Анны:", AnnaScore, "Финальный счет Алекса:", AlexScore, "\n")

Вывод:

Shell

Игра в кости использует модуль random

Анна выиграла жеребьевку.

Анна выиграла игру в кости. Финальный счет Анны: 5 Финальный счет Алекса: 2

Анна выиграла жеребьевку.

Анна выиграла игру в кости. Финальный счет Анны: 10 Финальный счет Алекса: 2

Алекс выиграл жеребьевку.

Анна выиграла игру в кости. Финальный счет Анны: 10 Финальный счет Алекса: 8

# Модуль\_13

- Код функции принято отделять двумя пустыми строками друг от друга и основного кода. Это нужно что бы выделить их в коде.

- Старайтесь придерживаться принципа единственной ответственности. Когда функция отвечает исключительно за что-то одно. Например запрос ввода у пользователя осуществляется в основном коде программы или в другой функции. Данные передаются в функцию через аргументы, сама же функция отвечает только за вычисления. Такой подход позволит легче масштабировать ваш код и сделает его более гибким. Что упростит поддержку в дальнейшем.

- Когда пишите функцию, делайте её независимой от остального кода. В ней не должны использоваться внешние переменные. Все необходимые данные она должна получать через аргументы.

- В программирование есть такое понятие как стек вызовов, возможно вы о нём слышали. Суть в том что когда одна функция вызывает другую функцию она не завершается. Она ждёт пока завершится вызванная ею функцию. Так вызов за вызовом этот стек наполняется. И как вы понимаете он не безграничен. Поэтому когда вам необходимо что бы какая-то функция выполнялась бесконечно долго её нужно помещать в цикл. Вот так например

## 13.2 Возврат значений из функций. Оператор return

**print('Задача 1. Сумма чисел 2')**

# Пользователь вводит число N. Напишите функцию summa\_n, которая принимает одно целое положительное число N и находит сумму всех чисел от 1 до N включительно. Функция вызывается два раза: сначала от числа N, а затем от полученной суммы.

# Пример работы программы:

# Введите число: 5

# Сумма от 1 до 5 = 15

# Сумма от 1 до 15 = 120

def summa\_n(number):

summa\_count = 0

for i in range(1, number + 1):

summa\_count += i

return summa\_count

num = int(input('Введите целое число: '))

next\_num = summa\_n(num)

print('Сумма от 1 до', num, '=', next\_num)

final\_result = summa\_n(next\_num)

print('Сумма от 1 до', next\_num, '=', final\_result)

print('Задача 2. «Назад в будущее»')

# Вы — один из разработчиков языка программирования Python, и вы пишете специальный математический модуль, который можно было бы просто подключить внутри программы и облегчить жизнь всем программистам.

# Реализуйте функцию gcd, которая получает два параметра — два числа — и возвращает наибольший общий делитель этих двух чисел.

# Пример работы программы:

# Введите первое число: 6

# Введите второе число: 10

# НОД = 2

import math

def nod():

nod = math.gcd(first\_num, second\_num)

print('НОД =', nod)

first\_num = int(input('Введите первое число: '))

second\_num = int(input('Введите второе число: '))

nod()

print('Задача 3. Приоритет задач')

print()

# В одном дата-центре ресурсы распределены так, что сначала обрабатываются крупные задачи, а затем уже идут небольшие. Каждая из этих задач, по сути, просто огромный поток цифр. Ваша задача, как программиста этого центра, написать программу, которая поможет определять, какую из задач нужно решать в первую очередь.

# Вводится последовательность из N чисел. Нужно определить номер числа, у которого больше всего цифр, и вывести на экран соответствующее сообщение. Если число отрицательное, то считать его за 0. Для подсчёта количества цифр реализуйте функцию numeral\_count.

# Пример работы программы:

# Введите кол-во задач: 4

# Введите число: 6

# Введите число: 14

# Введите число: 1

# Введите число: 13434

# Первая задача на обработку: 13434

def numeral\_count(number): #подсчет количества цифр в числе

count = 0

while number > 0:

number //= 10

count += 1

return count

def numeral\_check(n): #подсчет выполняемого количества задач и максимальной цифры

max\_count = 0

max\_number = 0

for \_ in range(1, n+1):

new\_value = int(input('Введите число: ')) #Вводим числа

if new\_value < 0:

new\_value = 0

cipher\_count = numeral\_count(new\_value) #если число больше максимального значения всех чисел, то присваиваем этому числу значение максимального.

if cipher\_count > max\_count:

max\_count = cipher\_count # максимальное кол-во задач

max\_number = new\_value # максимальное число равное новому значению.

return max\_number

how\_many = int(input('Введите кол-во задач: '))

print('Первая задача на обработку:', numeral\_check(how\_many))

## 13.3 Представление вещественных чисел в программе

# Задача 1. Возможности компьютера

#

# В одной IT-компании тестируют возможности различных языков программирования, компиляторов и, конечно же, компьютеров.

# Компания дала вам задачу понять, какое самое маленькое возможное число можно получить путём постоянного деления числа на 2.

# Изначально число равно единице. Также, помимо самого числа, компания просит вывести количество делений. Реализуйте такую программу.

#

#

#

number = 1

count = 0

save\_number = number

while number:

save\_number = number

number /= 2

count += 1

print(save\_number, count)

# Задача 2. Тестирование

#

# Команде программистов отдали на тестирование новую модель суперкомпьютера. Для начала программисты решили проверить,

# как у компьютера обстоят дела с вычислениями вещественных чисел. Разработчики компьютера предупредили,

# что на входе он работает только с экспоненциальной формой числа.

#

# Пользователь вводит число N в экспоненциальной форме, где мантисса всегда равна числу от 1 до 9, а порядок меньше нуля.

# Также есть переменная Х, которая изначально равна единице. Посчитайте, сколько раз нужно прибавить N к Х, чтобы оно перевалило за двойку.

#

# Дополнительно: обеспечьте контроль ввода.

#

#

#

# Пример 1:

#

# Введите число в эксп. форме: 1e-3

#

# Кол-во прибавлений: 1001

#

#

#

# Пример 2:

#

# Введите число в эксп. форме: 5.02e-1

#

# Кол-во прибавлений: 2

#

#

#

start\_number = 1

delta = float(input("Введите число в эксп. форме "))

count = 0

while start\_number <= 2:

start\_number += delta

count += 1

print("Кол-во прибавлений: ", count)

# Задача 3. Урок информатики

#

# На одном из уроков информатики учитель объяснял тему «Числа с плавающей точкой», но несколько учеников никак не могли понять,

# почему эта точка «плавает» и как вообще выглядят числа в такой форме. Для наглядности учитель написал программу,

# которая берёт число больше десяти и выводит его в формате плавающей точки.

#

# Пользователь вводит положительное число x (x > 10). Напишите функцию, которая выводит его в формате плавающей точки,

# то есть x = a \*10 \*\* b, где 1 ≤ a < 10.

#

#

#

# Пример 1:

#

# Введите число: 16

#

# Формат плавающей точки: x = 1.6 \* 10 \*\* 1

#

#

#

# Пример 2:

#

# Введите число: 92345

#

# Формат плавающей точки: x = 9.2345 \* 10 \*\* 4

start\_number = float(input("Введите число: "))

count = 0

while start\_number > 10:

count += 1

start\_number /= 10

print(f"Формат плавающей точки: x = {start\_number} \* 10 \*\* {count}")

## 13.4 Особенности работы с вещественными числами

# Задача 1. Опять налоги

#

# У правительства одной из стран есть бухгалтерская программа, которая суммирует налоги её граждан, компаний плюс НДС с товаров.

# Страна развивалась, суммарные налоги увеличивались, и бухгалтеры заметили, что после добавления к общей сумме налогов некого НДС

# от какого-то продукта общая сумма перестала меняться…

#

# Нужно помочь бухгалтерам: напишите функцию, на вход которой подаются два числа — общая сумма налога tax и новый налог new\_tax,

# который нужно добавить к общей сумме. Функция должна проверять, возможно ли сложить эти два числа или нет,

# и выводить соответствующее сообщение о том, увеличится ли бюджет или нет.

#

#

#

# Пример 1:

#

# Введите бюджет страны: 1.23e2

#

# Новые поступления (налог): 1.2e1

#

# Результат: Бюджет увеличится

#

#

#

# Пример 2:

#

# Введите бюджет страны: 1.231221200034e12

#

# Новые поступления (налог): 1.2e-4

#

# Результат: Бюджет не изменится

#

#

#

def check\_budget(old\_budget, new\_tax):

while old\_budget % 10 == 0:

old\_budget /= 10

new\_tax /= 10

if int(old\_budget + new\_tax) == int(old\_budget):

print("Бюджет не изменился")

else:

print("Бюджет изменился")

# budget = float(input("Введите бюджет страны: "))

# new\_tax = float(input("Новые поступления (налог): "))

budget = 1.23e2

new\_tax = 1.2e1

check\_budget(budget, new\_tax)

# Задача 2. Сравнение

#

# Так как в Python операции с вещественными числами могут давать неожиданные результаты

# (в частности, 0.1 + 0.2 не будет в точности равняться 0.3), стоит задача с этим как-то справляться.

#

# Напишите функцию eqv, которая принимает три числа и затем сравнивает сумму первых двух чисел с третьим с определённой степенью точности:

# до 15-го знака после точки. Если равенство выполняется, то функция возвращает True, иначе возвращает False.

#

#

#

# Пример 1:

#

# Введите первое число: 1.1

#

# Введите второе число: 2.2

#

# Введите третье число: 3.3

#

# True

#

#

#

# Пример 2:

#

# Введите первое число: 1e-14

#

# Введите второе число: 1e-14

#

# Введите третье число: 3e-14

#

# False

def eqv(a, b, c):

return abs((a + b) - c) <= 1e-15

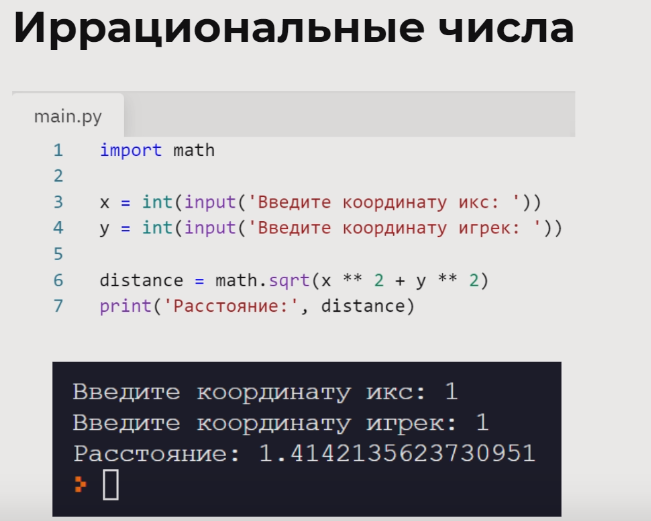
first = float(input("Введите первое число: "))

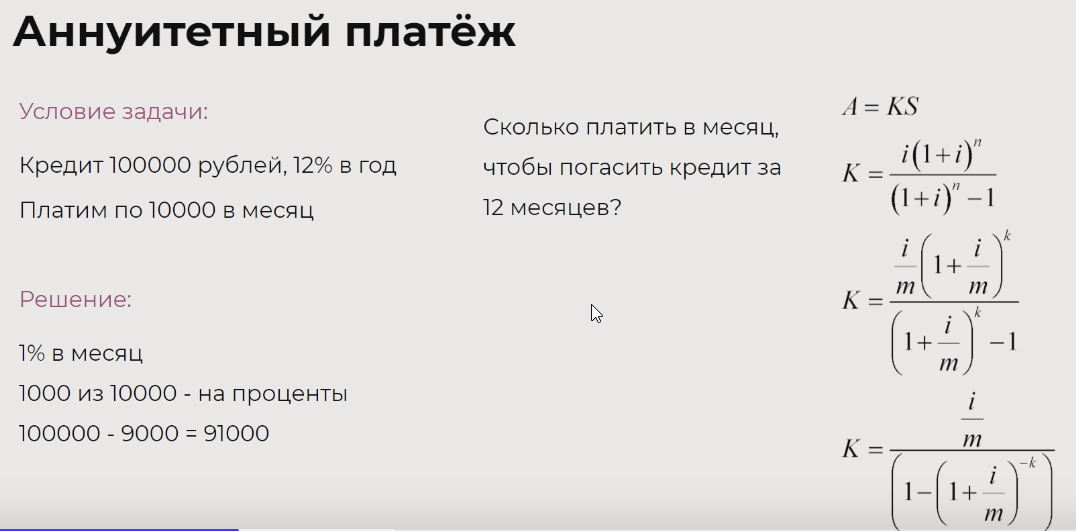
second = float(input("Введите второе число: "))

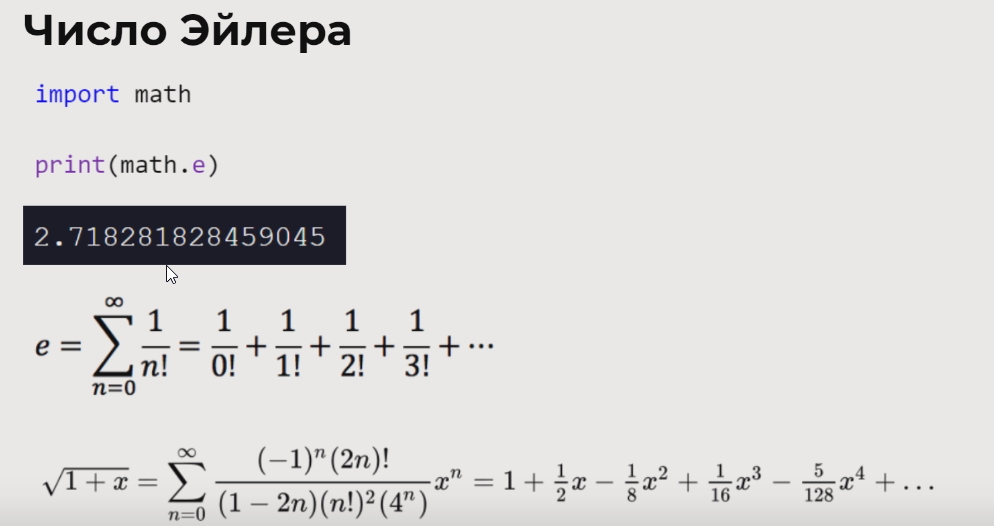
third = float(input("Введите третье число: "))

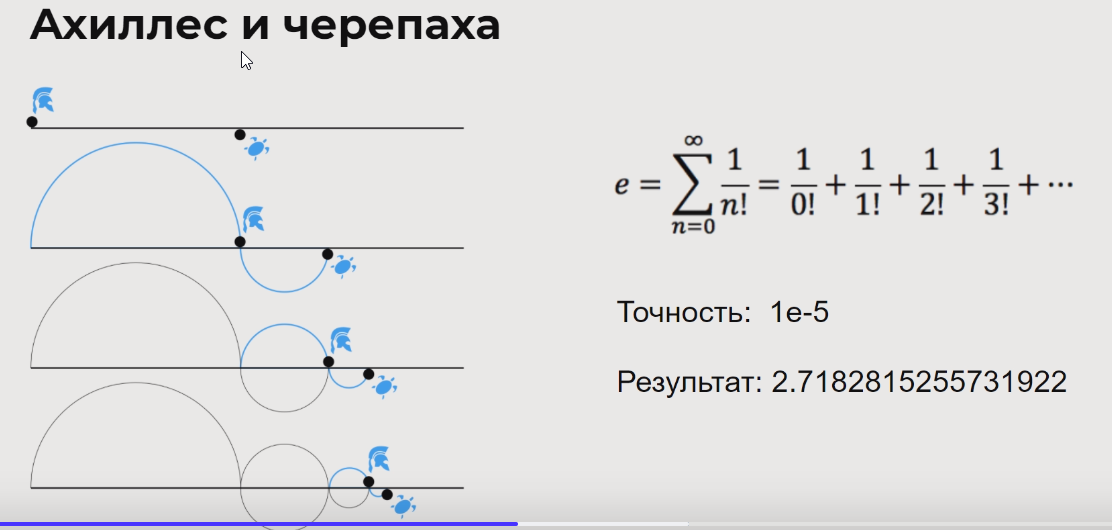
print(eqv(first, second, third))

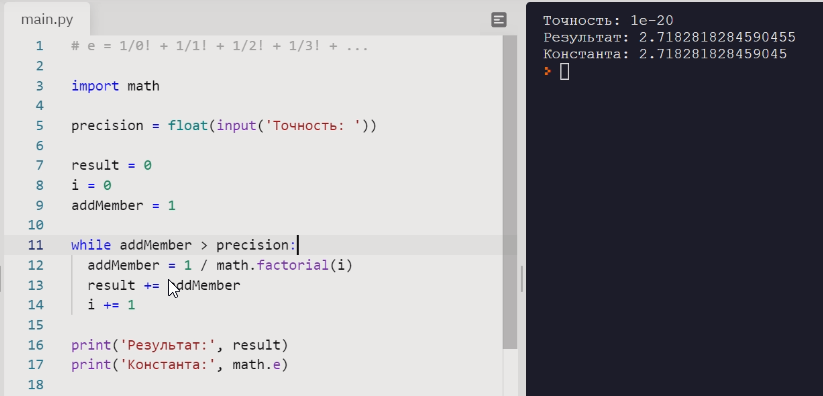
## 13.4 Алгоритмы с заданной точностью

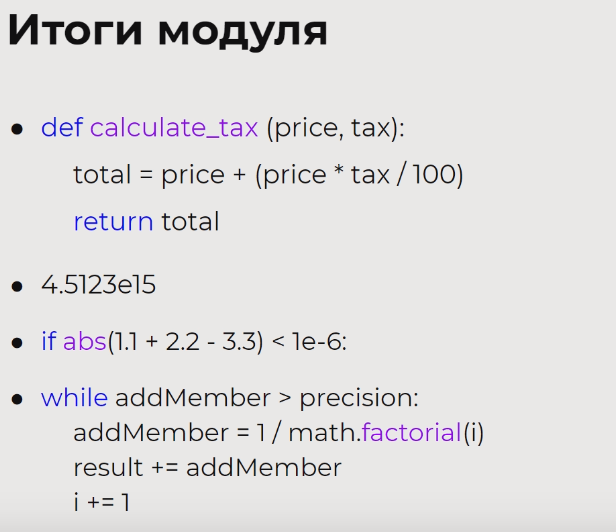












## 13.6 Практическая работа

print('Задача 1. Урок информатики 2')

# В прошлый раз учитель написал программу,

# которая выводит числа в формате плавающей точки, однако он вспомнил,

# что не учёл одну важную штуку: числа-то могут идти от нуля.

#

# Задано положительное число x (x > 0).

# Ваша задача преобразовать его в формат плавающей точки,

# то есть x = a \* 10 \*\* b, где 1 ≤ а < 10

#

# Обратите внимание, что x теперь больше нуля, а не больше единицы.

# Обеспечьте контроль ввода.

#

# Пример 1:

## Введитечисло: 92345

# Формат плавающей точки: x = 9.2345 \* 10 \*\* 4

#

# Пример 2:

# Введите число: 0.0012

# Формат плавающей точки: x = 1.2 \* 10 \*\* -3

def NumMore(user\_num):

count = 0 # b

while user\_num > 10:

user\_num /= 10

count += 1

# x = user\_num \* 10 \*\* count

print('\nФормат с плавающей точки: x =', user\_num, '\* 10 \*\*', count)

def NumLess(user\_num):

count = 0

while user\_num <= 1:

user\_num \*= 10

count -= 1

print('\nФормат с плавающей точки: x =', round(user\_num, 1), '\* 10 \*\*', count)

def Start():

user\_num = float(input('Введите число: ')) # x

if user\_num >= 11:

NumMore(user\_num)

elif 0 < user\_num < 1:

NumLess(user\_num)

else:

print('Ошибка ввода. (X > 10 или 0 < X < 1)')

Start()

print('Задача 2. Функция максимума')

# Юра пишет различные полезные функции для Python, чтобы остальным программистам стало проще работать. Он захотел написать функцию, которая будет находить максимум из перечисленных чисел. Функция для нахождения максимума из двух чисел у него уже есть. Юра задумался: может быть, её можно как-то использовать для нахождения максимума уже от трёх чисел?

# Помогите Юре написать программу, которая находит максимум из трёх чисел. Для этого используйте только функцию нахождения максимума из двух чисел.

# По итогу в программе должны быть реализованы две функции:

# 1) maximum\_of\_two — функция принимает два числа и возвращает одно (наибольшее из двух);

# 2) maximum\_of\_three — функция принимает три числа и возвращает одно (наибольшее из трёх); при этом она должна использовать для сравнений первую функцию maximum\_of\_two.

**Вариант №1.**

**Алгоритм решения задачи**

**1) Сначала предположим, что переменная a содержит наибольшее значение. Присвоим его переменной m.**

**2) Если текущее значение m меньше, чем у b, то следует присвоить m значение b. Если это не так, то не изменять значение m.**

**3) Если текущее значение m меньше, чем у c, то присвоить m значение c. Иначе ничего не делать.**

a = int(input('Введите число 1 -'))

b = int(input('Введите число 2 -'))

c = int(input('Введите число 3 -'))

m = a

if m < b:

m = b

if m < c:

m = c

print(m)

### **Переворачивание чисел**

**Использование цикла while для переворачивания числа**

Использование цикла while — один из способов перевернуть число в Python. Этот метод основан на последовательном извлечении последней цифры числа и добавлении ее в начало результирующего числа.

Для начала необходимо определить переменную, в которой будет храниться перевернутое число. Затем, пока исходное число не станет равным 0, будем извлекать последнюю цифру числа с помощью операции взятия остатка от деления на 10 и добавлять ее в конец результирующего числа, умножая его на 10. После каждой итерации исходное число целочисленно делится на 10, чтобы отбросить уже обработанную последнюю цифру.

Вот пример кода, демонстрирующий использование цикла while для переворачивания числа:

number = 12345

reversed\_number = 0

while number != 0:

digit = number % 10

reversed\_number = reversed\_number \* 10 + digit

number //= 10

print("Перевёрнутое число:", reversed\_number)

Copy

В этом примере мы используем цикл while для получения каждой цифры числа, начиная с младших разрядов. Мы сохраняем каждую цифру в переменной digit и затем добавляем ее в reversed\_number, умножив его на 10 и добавив digit. Затем мы целочисленно делим исходное число на 10 и повторяем цикл, пока number не станет равным нулю. В результате мы получаем перевернутое число в переменной reversed\_number.

**Использование цикла for и строки для переворачивания числа**

Для переворачивания числа в Python можно использовать цикл for в сочетании со строками. Для этого мы преобразуем число в строку, затем переворачиваем ее и затем преобразуем обратно в число.

Ниже приведен пример кода для переворачивания числа с использованием цикла for и строк:

num = 12345

reverse\_num = ""

for digit in str(num):

reverse\_num = digit + reverse\_num

reverse\_num = int(reverse\_num)

print(reverse\_num)

Copy

В этом примере мы объявляем переменную num и инициализируем ее значением 12345. Затем мы создаем пустую строку reverse\_num. Затем мы начинаем цикл for, который перебирает каждую цифру в строковом представлении числа num. В каждой итерации цикла мы добавляем текущую цифру в начало строки reverse\_num (используя операцию конкатенации строк), чтобы перевернуть число. В конце мы преобразуем строку reverse\_num в целое число с помощью функции int() и выводим результат.

**Использование срезов для переворачивания числа**

Для переворачивания числа в Python можно использовать срезы. Срезы позволяют извлекать из последовательности подстроку по указанным индексам. Для переворота числа мы можем сначала преобразовать его в строку, а затем применить срезы для изменения порядка символов.

Для начала преобразуем число в строку с помощью функции str(), а затем применим срезы для переворота числа. Чтобы перевернуть число, нужно указать отрицательный шаг в срезе. Вот пример:

num = 123456789

num\_str = str(num)

reversed\_num\_str = num\_str[::-1]

reversed\_num = int(reversed\_num\_str)

print(reversed\_num)

Copy

В этом примере мы сначала преобразуем число num в строку num\_str. Затем мы используем срез с отрицательным шагом [::-1], чтобы получить обратную последовательность символов. Наконец, мы преобразуем строку reversed\_num\_str обратно в число reversed\_num с помощью функции int().

**Рекурсивный метод для переворачивания числа**

Рекурсия — это процесс, в котором функция вызывает саму себя, чтобы решить задачу. В Python можно использовать рекурсию для переворачивания числа.

Для переворачивания числа с помощью рекурсии мы создадим функцию, которая будет вызывать саму себя, пока не достигнет базового случая. Базовый случай — это когда мы достигаем последней цифры числа. Затем мы будем добавлять каждую цифру к новому числу, которое мы создаем внутри функции.

Вот пример рекурсивной функции для переворачивания числа:

def reverse\_number(num):

if num < 10:

return num

else:

return int(str(num % 10) + str(reverse\_number(num // 10)))

Copy

В этой функции мы проверяем, является ли число num однозначным, т.е. меньше 10. Если это так, то мы просто возвращаем это число, так как оно уже перевернуто. Если число num имеет более одной цифры, то мы с помощью оператора % получаем последнюю цифру числа, а затем добавляем ее к новому числу, созданному с помощью рекурсивного вызова функции для оставшейся части числа, которая получается при делении num на 10 и отбрасывании остатка.

Вот как мы можем использовать эту функцию для переворачивания числа:

num = 12345

reversed\_num = reverse\_number(num)

print(reversed\_num) # 54321

Copy

**Обработка отрицательных чисел и чисел с десятичной точкой**

При работе с числами в Python необходимо учитывать, что некоторые методы и операции могут не поддерживать отрицательные числа или числа с десятичной точкой. При переворачивании числа также могут возникнуть проблемы с обработкой отрицательных чисел и чисел с десятичной точкой.

Если вы работаете с отрицательными числами, то можно использовать функцию abs() для получения модуля числа. Например, для переворачивания числа -123 мы можем использовать следующий код:

num = -123

reversed\_num = int(str(abs(num))[::-1]) \* (-1 if num < 0 else 1)

print(reversed\_num)

Copy

Здесь мы сначала получаем модуль числа с помощью функции abs(), затем преобразуем число в строку и переворачиваем с помощью среза. Для восстановления знака числа умножаем его на -1, если исходное число отрицательное, и на 1, если оно положительное.

Если вы работаете с числами с десятичной точкой и вам нужно сохранить дробную часть, то можно использовать модуль decimal и класс Decimal. Например, для переворачивания числа 3.14159 мы можем использовать следующий код:

from decimal import Decimal

num = Decimal('3.14159')

reversed\_num = Decimal(str(num)[::-1])

print(reversed\_num)

**print('Задача 3. Число наоборот 2')**

# Пользователь вводит два числа — N и K.

# Напишите программу,

# которая заменяет каждое число на число,

# которое получается из исходного записью его цифр в обратном порядке,

# затем складывает их,

# снова переворачивает и выводит ответ на экран.

# Пример:

# Введите первое число: 102

# Введите второе число: 123

# Первое число наоборот: 201

# Второе число наоборот: 321

# Сумма: 522

# Сумма наоборот: 225

def number\_opposite(numm):

new\_num = 0

summ = 0

while numm > 0:

new\_num = numm % 10

summ = summ \* 10 + new\_num

numm //= 10

return summ

num\_one = 102 #int(input('Введите первое число: '))

num\_two = 123 #int(input('Введите второе число: '))

print('\nПервое число наоборот:', number\_opposite(num\_one))

print('Второе число наоборот:', number\_opposite(num\_two))

summ = number\_opposite(num\_one) + number\_opposite(num\_two)

print('\nСумма:', summ)

print('Сумма наоборот:', number\_opposite(summ))

**print('Задача 4. Недоделка 2')**

# Вы всё так же работаете в конторе по разработке игр и смотрите различные программы прошлого горе-программиста. В одной из игр для детей, связанной с мультяшной работой с числами, вам нужно было написать код согласно следующим условиям: программа получает на вход два числа; в первом числе должно быть не менее трёх цифр, во втором — не менее четырёх, иначе программа выдаёт ошибку. Если всё нормально, то в каждом числе первая и последняя цифры меняются местами, а затем выводится их сумма.

# И тут вы натыкаетесь на программу, которая была написана предыдущим программистом и которая как раз решает такую задачу. Однако старший программист попросил вас немного переписать этот код, чтобы он не выглядел так ужасно. Да и вам самим становится, мягко говоря, не по себе от него.

# Постарайтесь разделить логику кода на три отдельные логические части (функции):

# count\_numbers — получает число и возвращает количество цифр в числе;

# change\_number — получает число, меняет в нём местами первую и последнюю цифры и возвращает изменённое число;

# main — функция ничего не получает на вход, внутри она запрашивает нужные данные от пользователя, выполняет дополнительные проверки и вызывает функции 1 и 2 для выполнения задачи (проверки и изменения двух чисел).

# Разбейте приведённую ниже программу на функции. Повторений кода должно быть как можно меньше. Также сделайте, чтобы в основной части программы был только ввод чисел, затем изменённые числа и вывод их суммы.

def count\_numbers(num):

count = 0

temp = num

while temp > 0:

count += 1

temp = temp // 10

return count

def change\_number(num):

last\_digit = num % 10

first\_digit = num // 10 \*\* (count\_numbers(num) - 1)

between\_digits = num % 10 \*\* (count\_numbers(num) - 1) // 10

return last\_digit \* 10 \*\* (count\_numbers(num) - 1) + between\_digits \* 10 + first\_digit

def main():

first\_n = int(input("\nВведите первое число: "))

if count\_numbers(first\_n) < 3:

print("В первом числе меньше трёх цифр.")

else:

first\_n = change\_number(first\_n)

print('Изменённое первое число:', first\_n)

second\_n = int(input("\nВведите второе число: "))

if count\_numbers(second\_n) < 4:

print("Во втором числе меньше четырёх цифр.")

else:

second\_n = change\_number(second\_n)

print('Изменённое второе число:', second\_n)

print('\nСумма чисел:', first\_n + second\_n)

main()

**print('Задача 5. Маятник ')**

# Известно, что амплитуда качающегося маятника с каждым разом затухает

# на 8,4% от амплитуды прошлого колебания.

# Если качнуть маятник,

# то, строго говоря, он не остановится никогда,

# просто амплитуда будет постоянно уменьшаться до тех пор,

# пока мы не сочтём такой маятник остановившимся.

# Напишите программу,

# определяющую, сколько раз качнётся маятник, прежде чем он, по нашему мнению, остановится.

# Программа получает на вход

# начальную амплитуду колебания в сантиметрах

# и конечную амплитуду его колебаний,

# которая считается остановкой маятника.

# Обеспечьте контроль ввода.

# Пример:

# Введите начальную амплитуду: 1

# Введите амплитуду остановки: 0.1

# Маятник считается остановившимся через 27 колебаний

start = float(input('Введите начальную амплитуду: '))

stop = float(input('Введите амплитуду остановки: '))

count = 0

while start > stop:

start \*= 0.916 # уменьшаем амплитуду на 8,4% (1 - (8,4/100))

count += 1

print('\nМаятник считается остановившимся через', count, 'колебаний.')

**print('Задача 6. Яйца')**

# В рамках программы колонизации Марса

# компания «Спейс Инжиниринг» вывела особую породу черепах,

# которые, по задумке, должны размножаться, откладывая яйца в марсианском грунте.

# Откладывать яйца слишком близко к поверхности опасно из-за радиации,

# а слишком глубоко — из-за давления грунта и недостатка кислорода.

# Вообще, факторов очень много,

# но специалисты проделали большую работу и предположили,

# что уровень опасности для черепашьих яиц рассчитывается по формуле

# D = x\*\*3 − 3x\*\*2 − 12x + 10,

# где x — глубина кладки в метрах,

# а D — уровень опасности в условных единицах.

#

# Для тестирования гипотезы

# нужно взять пробу грунта на безопасной, согласно формуле, глубине.

#

# Напишите программу,

# находящую такое значение глубины "х", при котором уровень опасности как можно более близок к нулю.

# На вход программе подаётся максимально допустимое отклонение уровня опасности от нуля,

# а программа должна рассчитать приблизительное значение "х",

# удовлетворяющее этому отклонению.

#

# Известно, что глубина точно больше нуля и меньше четырёх метров.

#

# Обеспечьте контроль ввода.

#

# Пример:

# Введите максимально допустимый уровень опасности: 0.01

#

# Приблизительная глубина безопасной кладки: 0.732421875 м

def f(x):

return x \*\* 3 - 3 \* x \*\* 2 - 12 \* x + 10

# Две точки a и b на отрезке, такие что d(a) < 0 и d(b) > 0.

def find\_depth(d):

a, b = 0, 4 # зададим начальный интервал [a, b], где a = 0 и b = 4 (примем максимально допустимую глубину), и найдем корень на этом интервале с заданной точностью.

while abs(b - a) > d:

c = (a + b) / 2 # Находим середину отрезка a-b: c = (a + b) / 2. Если d(c) близко к 0, то c является приближенным решением уравнения. Иначе, если d(c) < 0, то корень находится в интервале [c, b], иначе в интервале [a, c].

if f(c) == 0:

return c

elif f(c) \* f(a) < 0:

b = c

else:

a = c

return (a + b) / 2

d = float(input("Введите максимально допустимый уровень опасности: "))

# Контроль ввода

if d > 0:

depth = find\_depth(d)

print("Приблизительная глубина безопасной кладки:", depth, "м.")

else:

print("Некорректный ввод. Попробуйте еще раз.")

