Занятие 1.
Установка Python. Написание первых программ

Установка Anaconda

Anaconda – это дистрибутив Python, содержащий дополнительные пакеты. Зайти на сайт https://www.anaconda.com/products/individual загрузить и установить.

Установка по умолчанию в папку: C:\Users\<имя пользователя>\anaconda3

Установка Jupiter notebook

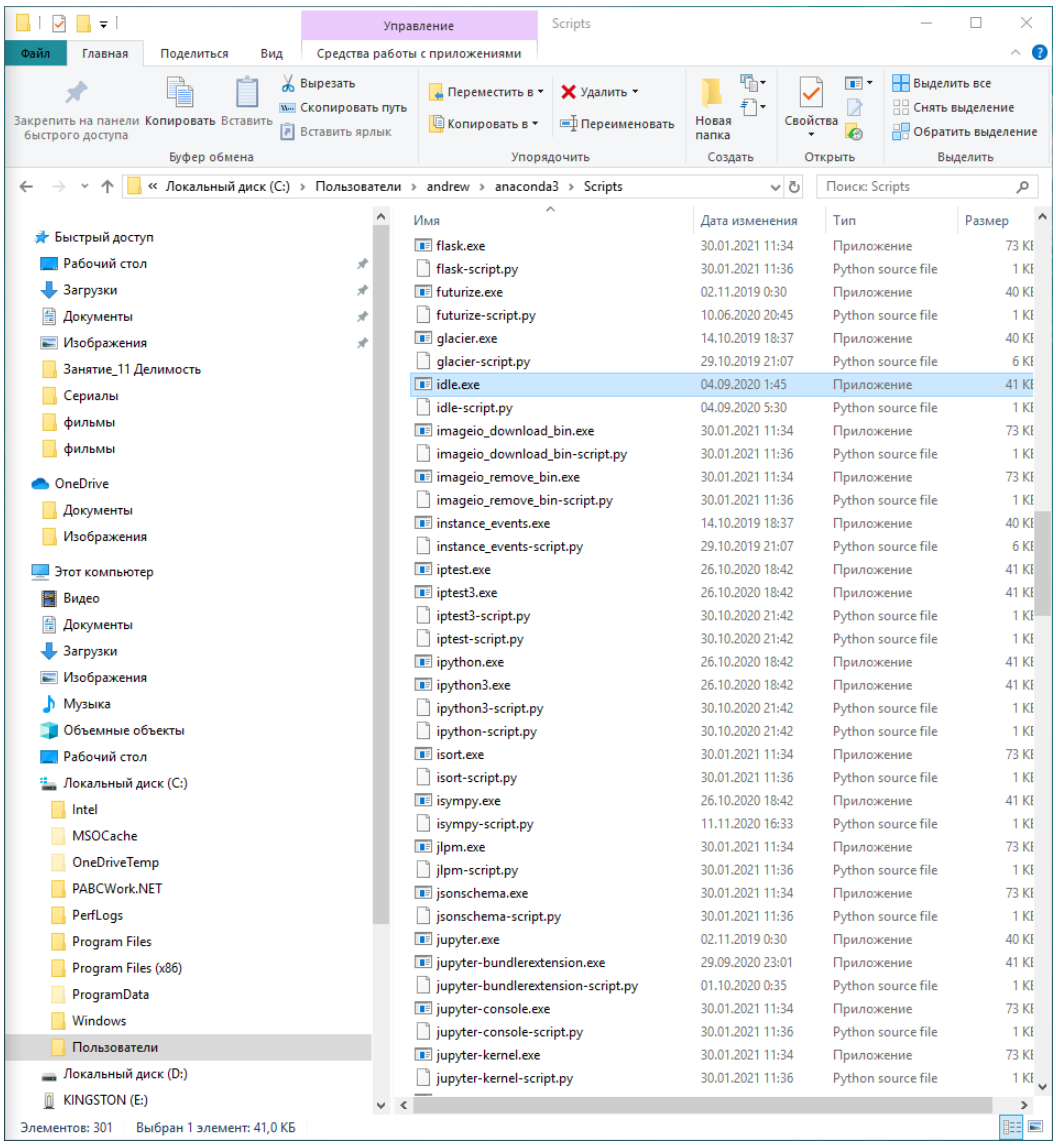
Jupiter notebook – инструмент для разработки и представления проектов в интерактивном виде (https://jupyter.org/install).

Вместе с Anaconda был установлен сам Python и Jupiter notebook.

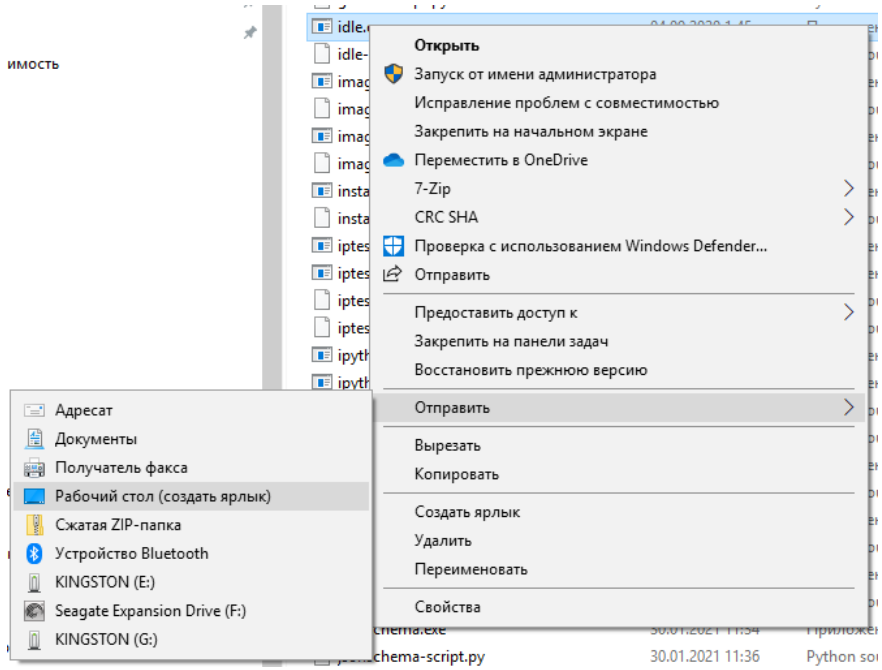
Замечание. Если бы мы устанавливали Python без дополнительных пакетов (не Anaconda), то Jupiter notebook пришлось бы устанавливать дополнительно.

Интегрированная среда разработки IDLE

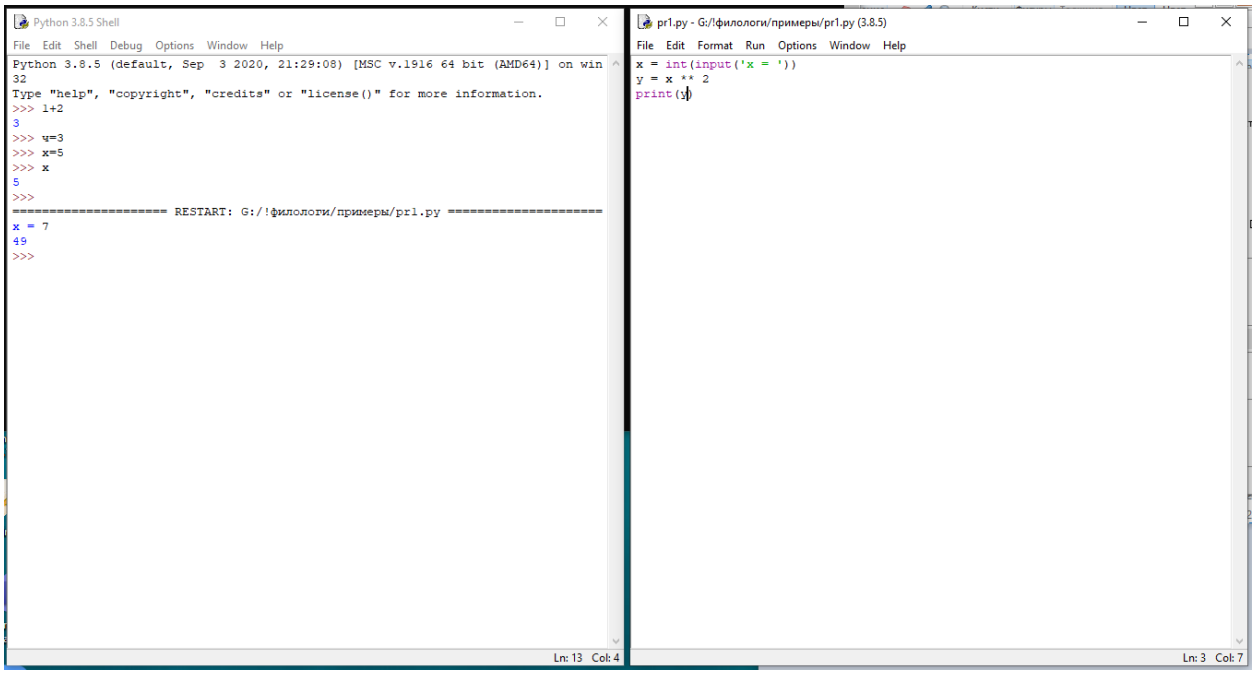
Простая среда разработки, устанавливается вместе с Python. Если мы устанавливали Anaconda, то найти ее можно на компьютере: C:\Users (или Пользователи)\<имя пользователя>\anaconda3\Scripts\idle.exe



Можно создать на рабочем столе ярлык:



и использовать IDLE для написания программ.

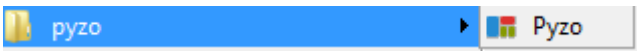


Интегрированная среда разработки Pyzo

Сред разработки на Python много, есть простые и сложные, бесплатные и платные. Pyzo более удобная чем IDLE среда разработки, но вместе с тем такая же простая и бесплатная.

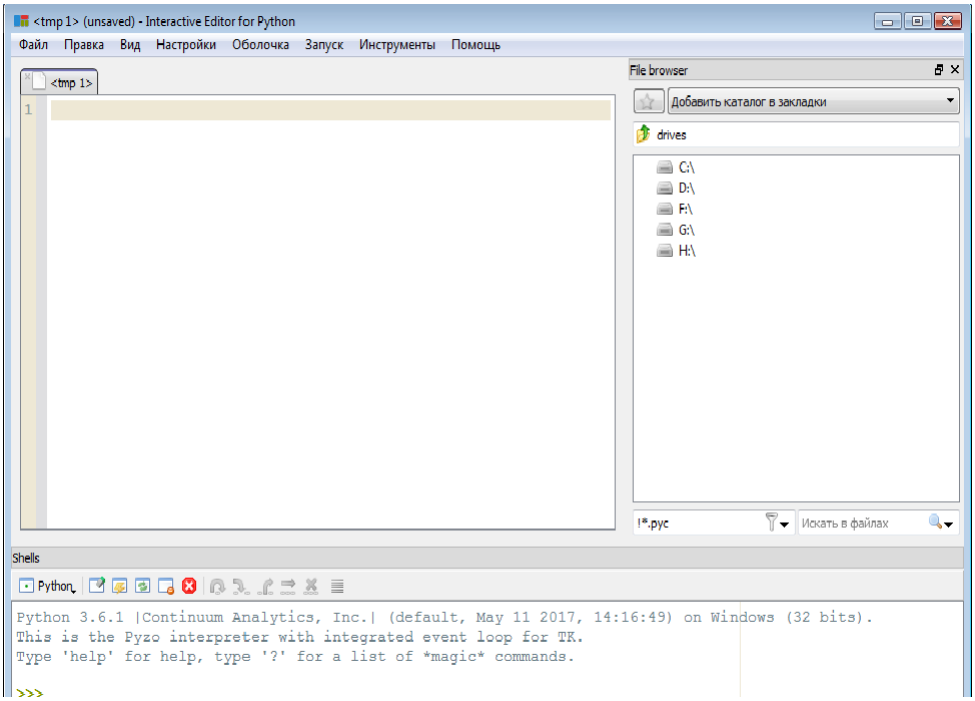
На сайте http://www.pyzo.org/start.html скачайте Windows: Pyzo installer и установите.

Ярлык программы Pyzo появится в меню Пуск Windows и на рабочем столе.



ярлык Pyzo: 

Pyzo – простая интегрированная среда разработки для работы с Python, имеет вид:



Она позволяет набирать текст программы, запускать и видеть результаты ее работы.

Окна Pyzo:

1. Окно кода программы, в нем набираем текст программы (на рис. слева вверху <tmp 1>);

2. Дерево каталогов (на рис. справа “File browser”). В этом окне выбираем текущий каталог, удобно искать и открывать сохраненные файлы.

3. В окне Shells будет запусаться программа (на рис. внизу “Shells”).

Окна можно перемещать, например, удобно расположить их так, как на приведенном рисунке: окно кода программы занимает большую часть экрана, внизу Shell, а справа – дерево каталогов.

Минимальный набор команд для работы в среде:

1. Справа в окне File browser установите текущую папку.
2. Создать новый файл: File→New
3. Открыть существующий файл: File→Open или лучше в окне File browser.
4. Сохранить открытый файл: File→Save или нажать Ctrl+S. Периодически в процессе набора и редактирования программы ее надо сохранять. Желательно сохранять программу как можно чаще. Это делается для того, чтобы не потерять текст программы и его не пришлось переписывать заново.

При исправлении ошибок рекомендуется сохранять перед каждым новым запуском.

Чтобы быстро находить файлы, лучше в названии файла сохранять номер лабораторной и номер задания, например, так: lab1_pr1.py. Удобно также для каждой лабораторной заводить отдельную папку.

5. Запуск программы: Run→Execute File или нажать Ctrl+E.
6. Можно изменить язык оболочки на русский: Settings→Select language→Russian.

Интегрированная среда разработки PyCharm

Можно еще использовать PyCharm, есть бесплатная и платная версии. У этой среды больше возможностей, но на старых компьютерах она может очень медленно работать. На сайте https://www.jetbrains.com/ru-ru/pycharm/download/#section=windows скачиваем и устанавливаем бесплатную (Community) версию.

В PyCharm нужно создавать проект, который может содержать несколько файлов Python. В интегрированной среде разработки PyCharm можно работать с разными языками программирования.

Первые программы

Пример 1. Первая программа (ввод/вывод данных)

```
# Ввод строки, функция input с одним параметром:
name = input('Как Вас зовут? : ')

# Вывод, функция print, может иметь несколько параметров:
print('Здравствуй,', name, '!')
print('Добро пожаловать в мир программирования на Python!!!')
```

Тест (запуск программы):

Как Вас зовут? : Петя
Здравствуй, Петя !
Добро пожаловать в мир программирования на Python!!!

Пример 2. Арифметические операции

```
# Строку можно перевести в другой тип (int - целое число):
x = int(input('x = '))
y = int(input('x = '))

# Арифметические операции:
add = x + y
sub = x - y
mul = x * y
div = x / y

# Вывод чисел (результат):
print('x + y =', add)
print('x - y =', sub)
print('x * y =', mul)
print('x / y =', div)
```

Тест:

x = 8
x = 2
x + y = 10
x – y = 6
x * y = 16
x / y = 4.0

Пример 3. Вычисление значение выражения (TF–IDF) по формулам:

1) TF (term frequency – частота слова):

$$TF = \frac{n_t}{\sum_k n_k},$$

где n_t – число вхождений слова t в документ, $\sum_k n_k$ – общее количество слов в документе.

2) IDF (inverse document frequency – обратная частота документа):

$$IDF = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

где $|D|$ – общее число документов в коллекции D, $|\{d_i \in D \mid t \in d_i\}|$ – число документов из коллекции D, в которых встречается слово t. Основание логарифма в формуле можно брать разное (например, 2, 10, $e \approx 2.718$ – натуральный логарифм).

3) И их произведение TF–IDF – статистическая мера, используемая для оценки важности слова в контексте документа:

$$TF\text{-}IDF = TF \times IDF.$$

```
# Вычисление TF-IDF для одного слова.
# Подключаем математический пакет (для логарифма):
import math

# Запрашиваем данные у пользователя:
nt = int(input('Введите число вхождений слова в документ: '))
N  = int(input('Введите общее число слов в документе: '))
D  = int(input('Введите количество документов: '))
Dt = int(input('Введите количество документов, содержащих это
слово: '))

# Вычисление TF:
TF = nt / N

# Вычисление IDF:
IDF = math.log( D / Dt, 2) # - логарифм по основанию 2

# Вычисление TF-IDF:
TF_IDF = TF * IDF

# Выводим значения на экран:
print('TF =', TF)
print('IDF =', IDF)
print('TF-IDF =', TF_IDF)
```

Пример работы программы (тест 1):

Введите число вхождений слова в документ: 5

Введите общее число слов в документе: 10
Введите количество документов: 100
Введите количество документов, содержащих это слово: 5
TF = 0.5
IDF = 4.321928094887363
TF-IDF = 2.1609640474436813

Пример работы программы (тест 2):
Введите число вхождений слова в документ: 1
Введите общее число слов в документе: 10
Введите количество документов: 100
Введите количество документов, содержащих это слово: 90
TF = 0.1
IDF = 0.15200309344505006
TF-IDF = 0.015200309344505007

Пример работы программы (тест 3):
Введите число вхождений слова в документ: 10
Введите общее число слов в документе: 10
Введите количество документов: 100
Введите количество документов, содержащих это слово: 1
TF = 1.0
IDF = 6.643856189774725
TF-IDF = 6.643856189774725

Пример 4. Выделение цифр натурального числа
Применим операции деления числа с остатком на 10 (a // 10) и получения остатка при делении на 10 (a % 10):

```
12345 // 10 # получим 1234
12345 % 10  # получим 5;  Разрезает: 1234 | 5
```

Применим операции // 100 и % 100 к числу:

```
12345 // 100 # получим 123
12345 % 100  # получим 45; Разрезает: 123 | 45
```

Пример. Дано целое трехзначное число. Выделим из него все цифры.

```
n = int(input('n = '))
c1 = n % 10;              # младшая цифра
c2 = n // 10 % 10         # цифра десятков
c3 = n // 100             # цифра сотен
print('Цифры:', c3, c2, c1)
```

Например, если ввести: **n = 327**
Программа вернет: **Цифры: 3 2 7**

Если нужно, наоборот, собрать число из цифр, то записываем:

```
m =  c3 * 100 + c2 * 10 + c1
```

Получим число 723.

Пример. Поменять местами первую и последнюю цифры трехзначного числа (например, 345 → 543).

```
n = int(input('n = '))
c1 = n % 10;
c2 = n // 10 % 10
c3 = n // 100
m = c1 * 100 + c2 * 10 + c3      # собрать число
print('Результат:', m)
```

Если ввести: **n = 345**

Программа вернет: **Результат: 543**

Самостоятельная работа в компьютерном классе

1. Переделайте «Пример 2. Арифметические операции», добавьте к нему еще операции: возведение в степень, деление с остатком, остаток от деления и одну любую битовую операцию
2. Напишите программу перевода температуры в градусах по Фаренгейту в градусы по Цельсию

$$Celsius = (5/9)*(Fahrenheit - 32).$$

3. Переделайте пример 3, добавьте возможность ввода пользователем основания логарифма в формуле.
4. На основе примера 4 сделайте следующую программу. Ввести двузначное натуральное число n, вырезать из его записи первую и вторую цифры, вывести их на экран.
5. Ввести трехзначное натуральное число n, найти сумму его цифр.

Занятие 2. Функции округления. Тип bool и условный оператор. Модуль random. Ввод/вывод данных. Форматирование строк %

Примеры

Функции округления	
int(x)	Перевод к целому типу (усекает дробную часть)
round(x,[n])	Округляет до ближайшего кратного 10^{-n}
из модуля math:	
ceil(*x*)	Возвращает округленное до большего целого значение (потолок) числа *x*.
floor(*x*)	Возвращает округленное до меньшего целого значение (пол) числа *x*.
trunc(*x*)	Усекает дробную часть числа *x*. (= int)

Замечание. Функция round немного отличается от обычного округления! round – округляет до ближайшего целого. Но, если расстояние одинаковое, например, число 4.5, то округляет к ближайшему четному (банковское округление)!

https://docs.python.org/3.7/library/functions.html?highlight=round#round

Пример 1.

```
round(4.501)          # вернет ближайшее: 5
round(4.499)          # вернет ближайшее: 4
# Особенность round в Python (банковское округление):
round(4.5)       # вернет ближайшее четное: 4
round(5.5)       # вернет ближайшее четное: 6
```

У round также есть второй необязательный параметр:

```
round(12.537, 2)          # округлять до 2 знаков: 12.54
# Особенность round в Python (банковское округление, из-за
ошибок округления при представлении дробных чисел может ок-
руглить в любую сторону):
round(12.125, 2)          # округлять до 2 знаков: 12.12
round(12.135, 2)          # округлять до 2 знаков: 12.13
round(3.135, 2)           # к нечетному: 3.13
```

Если нужно обычное округление, можно использовать формулу:

```
round_x = int(x + (0.5 if x > 0 else -0.5))
```

Пример 2.

```
import math
x = float(input('x = '))     # x = 3.7    -3.7
print(int(x))                # x = 3      -3
print(round(x))              # x = 4      -4
```

```
print(math.ceil(x))          # x = 4      -3
print(math.floor(x))         # x = 3      -4
print(math.trunc(x))         # x = 3      -3
```

Тип bool и условный оператор

Пример 3. Использование переменных bool

```
b1 = True
b2 = False      # Существуют две константы: True и False
b3 = 5 > 3      # b3 = True, т.к. 5 больше 3
if b3: print('5 больше 3')
```

Условия можно объединять с помощью операций and, or, not:

Условие	Описание
(x > y) and (x > z) или x > y and x > z	истина, если x больше y и z (здесь скобки не обязательны, см. таблицу “Порядок вычисления операторов” в Лекции 1 на стр. 33–34)
x == 0 or x == 1	истина, если x равно 0 или x равно 1
x >= a and x <= b или a <= x <= b	$x \in [a, b]$ принадлежит отрезку
x < a or x > b	$x \notin [a, b]$ не принадлежит отрезку

Пример 4. Проверка положительности целого числа

Эта программа вводит с клавиатуры целое число N, и в зависимости от выполнения условия выводит на экран одно из двух сообщений.

```
N = int(input('N = '))
# 1. Используем оператор if:
if N > 0: print('Число N - положительное')
else: print('Число N – неположительное')

# 2. Используем условное выражение if:
print('Число N - положительное' if N > 0 else 'Число N –
неположительное')

# 3. Или так:
print('Число N - положительное: ', N > 0)
```

Пример 5. Проверка четности целого числа

Эта программа вводит с клавиатуры целое число N, вычисляет остаток от деления его на 2 и в зависимости от значения остатка выводит на экран одно из двух сообщений.

```
N = int(input('N = '))
if N % 2 == 0:
    print('Число N - четное')    # с отступом в новой строке
else:
    print('Число N – нечетное')
```

Некоторые функции из модуля random
(генерация псевдослучайных чисел)

https://docs.python.org/3.7/library/random.html?highlight=random#module-random

Функция	Описание
seed([n])	Инициализация генератора случайных чисел; без параметров – используется значение системного времени
randint(a, b)	Возвращает случайное целое в диапазоне [a, b]
choice(seq)	Возвращает случайный элемент из последовательности seq
shuffle(L)	Перемешивает элементы в списке L
random( )	Возвращает случайное вещественное число в диапазоне [0, 1)
random.uniform(*a*, *b*)	Возвращает случайное вещественное число в диапазоне [a, b], формула: a + (b – a) * random( )

Пример 6.

```
import random  # Подключаем модуль
random.seed()  # Инициализация генератора
print(random.randint(-5, 5)) # Случайное целое [-5, 5]
print(random.random())  # Случайное вещественное [0, 1)
a = -5
b = 5
print(uniform(a, b))    # Случайное вещественное [a, b]
# или так:
print(a + (b - a) * random.random())
```

Пример 7. Использования random. Программа «Угадай число». Генерируется случайное число от 1 до 10. Пользователю предлагается его угадать.

```
import random
random.seed() # Инициализация генератора
RandomN = random.randint(1, 10) # Случайное от 1 до 10

N = int(input('N = '))
if N == RandomN: print('Ура, правильно!!!')
else: print('Не угадал :(')
```

Ввод данных – функция input

Считывает и возвращает в программу строку входных данных

s = input(строка-приглашение)

Строка-приглашение – необязательный параметр, выводится перед вводом данных. Функция считывает строку данных, полученную с устройства ввода.

s = input( )

s = input('Введите строку: ')

Введенную строку можно сразу перевести в требуемый тип:

***n* = int(input('Введите целое число: '))**

***x* = float(input('Введите вещественное число: '))**

Вывод данных – функция print (Python 3.0)

Имеет произвольное количество позиционных аргументов и необязательные аргументы, задаваемые только по имени:

print(*value, sep = ' ', end = '\n')

Все позиционные аргументы выводятся в текущий поток вывода, с предварительным переводом их в строку методом str.

Необязательные аргументы передаются в конце только по имени:

sep – строка-разделитель, по умолчанию пробел;

end – строка добавляется в конец вывода, по умолчанию переход на следующую строку;

Пример.

```
print('abc', 100, 3.14)          # Выведет на консоль:    abc 100 3.14
print(1, 2, 3, sep = '; ')       #     1; 2; 3
print(1, 2, 3, sep = ' - ', end = ' --- ' )
print(4, 5)                      #  1 - 2 - 3 --- 4 5
```

Форматирование строк с помощью операции %

https://docs.python.org/3.7/library/stdtypes.html#printf-style-string-formatting

Чтобы применить форматирование к строке надо слева от % задать саму строку с одним или несколькими знаками формата %, а справа от % указать одно или несколько значений, количество параметров и значений должно совпадать:

(с одним параметром)

'В коробке осталось %d конфет' % 5

 # результат: 'В коробке осталось 5 конфет'

или (с двумя параметрами)

'У нас есть %d конфет и %d ребят' % (5, 3)

 # результат: 'У нас есть 5 конфет и 3 ребят'

Можно еще использовать словари:

'Число %(n)d и строка %(s)s из словаря' % {'n' : 11, 's': 'ЭТА'}

 # результат: 'Число 11 и строка ЭТА из словаря'

Спецификатор	Описание
s	строка (вызов метода str)
r	строка (вызов метода repr)
c	символ
d, i	десятичное целое число
o	восьмеричное целое число
x, X	шестнадцатеричное целое число
e, E	вещественное число в экспоненциальной форме
f, F	вещественное число с десятичной точкой
g, G	вещественное число f, e, F, E
%	символ %

Общий вид спецификатора формата:

%[(имя)] [флаги] [ширина] [.точность] спецификатор формата

Замечание. [ ] – означают необязательные части

(имя) – если используются значения из словаря

Флаг	Значение
0	заполнение нулями (для чисел)
–	выравнивание по левому краю
пробел	пробел перед положительным числом
+	знак + или – будет стоять перед числом
#	альтернативная форма

ширина – минимальное количество символов на значение

точность – сколько символов выделено на дробную часть

Примеры

Формат	Результат
Строки:	
'%5s' % 'abc'	'□□abc'
'%-5s' % 'abc'	'abc□□'
'%6.3s' % '1234567890'	'□□□123'
Целые числа:	
'% +d' % 123	'+123'
'%5d' % 1234567890	'1234567890'
'%5d' % 123	'□□123'
Вещественные числа:	
'%f' % 3.1415	'3.141500'
'%6.4f' % 123.456789	'123.4568'
'%10.4f' % 12.34	'□□□12.3400'
print('Сумма: %5.3f' % (5.4 + 3.3))	Сумма: 8.700
'%E' % 3.1415	'3.141500E+00'
'%10.4E' % 123.456789	'1.2346E+02'
'%4.2e' % 123.456789	'1.23e+02'
'%10.2e' % 1.234	'□□1.23e+00'
'%g' % 0.0001	'0.0001'
'%g' % 0.00001	'1e-05'
'%g' % 100000	'100000'
'%g' % 1000000	'1e+06'
'%8.4G' % 1234	' 1234'
'%8.3G' % 1234	'1.23E+03'

G, g – использует экспоненциальный формат, если показатель степени меньше –4 или не меньше точности

Пример. **Что будет выведено на экране:**

```
print('a=%10.3f;' % 12.34567)
print('b=%10.3e' % 12.34567, end = '.')
```

Результат:

```
a=□□□□12.346;
b=□1.235e+01._
```

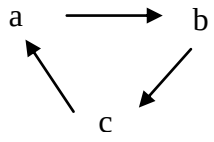
□ – отмечены пробелы

нет перехода на следующую строку

Вывод в 2 разных форматах, на числа выделено по 10 символов всего и по 3 после запятой: дробная часть округляется до 3 знаков; запись дополняется пробелами до 10 символов.

Самостоятельная работа в компьютерном классе
Сделайте одну любую задачу из каждого блока, выделенного линиями.

1. Даны два числа: a, b. Напишите программу, меняющую значения переменных a ↔ b (с использованием третьей вспомогательной переменной).
2. Даны два числа: a, b. Напишите программу, меняющую значения переменных a ↔ b (без использования третьей переменной).
3. Даны три числа: a, b, c. Напишите программу, меняющую значения этих переменных.



Программы этого блока напишите в 2 вариантах: (1) используйте условный оператор if и (2) уловное выражение if.

4. Составьте программу, которая вводит с клавиатуры два целых числа N и M, и проверяет, делится ли N на M.
5. Составьте программу, которая вводит с клавиатуры натуральное число N, проверяет, является ли оно двузначным и выводит на экран сообщение.
6. Ввести двузначное натуральное число N; проверить, совпадают ли первая и вторая цифры в его записи; выдать на экран сообщение.

---

7. Составьте программу, которая вводит с клавиатуры два целых числа x, y и выводит на экран значение $\min(x, y)$.
8. Составьте программу, которая вводит с клавиатуры три целых числа x, y, z и выводит на экран значение $\max(x, y, z)$.
9. Составьте программу, которая вводит с клавиатуры три целых числа x, y, z и выводит на экран значение $\min(x, y, z)$.

---

10. Ввести два числа: X и Y. Если они оба положительны или оба отрицательны, поменять их знаки на противоположные. Например, если $X = 5, Y = 4$, то сделать их $X = -5, Y = -4$.

В следующих трех программах используйте только один условный оператор if.

11. Ввести три целых: N, M и P. Проверить, являются ли они все положительными.
12. Ввести три целых: N, M и P. Проверить, является ли хотя бы одно из них положительным.

13. Ввести три целых: N, M и P. Проверить, является ли только одно из них положительным.

14. Программа «Угадай число» (второй вариант). Программа генерирует случайное число от 1 до 10. Пользователь должен угадать это число с трех попыток. После ввода очередного числа, программа сообщает ему, больше или меньше его число загаданного, например, так: «Введенное число больше загаданного, у Вас есть еще одна попытка».
Для получения случайного числа используйте модуль random *(см. пример 7).*

---

15. Составьте программу, которая вводит целое число n и вычисляет значение функции:

$$f(n)=\begin{cases} n^2, & \text{при } n\in[0,10] \\ |n|, & \text{в противном случае} \end{cases}$$

16. Вычислить $\sqrt[3]{x}$, используя формулу: Нужно учесть, что дробная степень корректно вычисляется только для $x>0$, поэтому вычисление можно разбить на три случая:

$$\sqrt[3]{x}=\begin{cases} \sqrt[3]{x} & при \quad x>0 \\ 0 & при \quad x=0 \\ -\sqrt[3]{-x} & при \quad x<0 \end{cases}$$

17. Вычислить значение ступенчатой функции

$$f(x)=\begin{cases} 1 & при \quad x<0 \\ 2 & при \quad 0\le x<10 \\ 3 & при \quad 10\le x<20 \\ 4 & при \quad 20\le x<30 \\ 5 & при \quad 30\le x \end{cases}$$

18. Ввести натуральное число M. Проверить, является ли оно квадратом другого натурального числа; выдать на экран сообщение.
Подсказка: если вычислить $\sqrt{M}$ *и округлить, то это и будет целым числом – кандидатом, для которого может выполниться равенство:* $N^2=M$.

Например, если M=4, то $N = round(\sqrt{4}) = 2$ *и действительно* $N^2 = M$ *– число 4 является квадратом 2.*

Если же M=5, то $N = round(\sqrt{5}) = 2$, *но* $N^2 \neq M$ *– число 5 не является квадратом другого целого числа.*

19. Ввести натуральное число M . Проверить, является ли оно кубом другого натурального числа; выдать на экран сообщение.

См. подсказку к предыдущему примеру.

---

20. Составьте программу, которая вводит с клавиатуры три целых числа N, M и P, и проверяет, сколько из них являются положительными.
21. Составьте программу, которая вводит с клавиатуры трехзначное число N, и проверяет, есть ли среди его цифр цифра 5.
22. Ввести пятизначное натуральное число N; проверить, является ли оно палиндромом; выдать на экран сообщение.

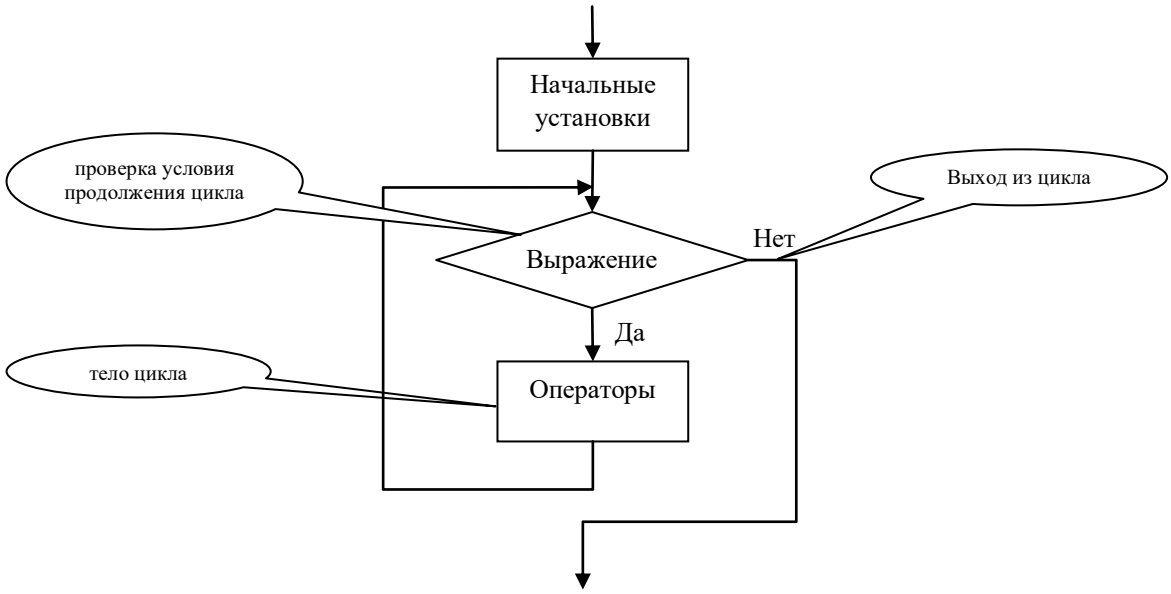
Занятие 3. Циклические алгоритмы. Циклы while и for – in

Часть I. Цикл с предусловием while

Операторы цикла используются для вычислений, повторяющихся многократно. В Python имеется два цикла: цикл с предусловием while и цикл с параметром for. Каждый из них состоит из определенной последовательности операторов.

Блок, ради выполнения которого и организуется цикл, называется *телом цикла*. Остальные операторы служат для управления процессом повторения вычислений: это *начальные установки*, проверка условий продолжения цикла и модификатор условия. Один проход цикла называется *итерацией*.

Структуру цикла с предусловием while можно изобразить так:



Синтаксис цикла с предусловием while:

```
while выражение:
      операторы
[else:
      операторы ветки иначе]
```

Примеры

Пример 1. Вывод на экран чисел от 1 до 5:

```
k = 1            # начальные установки
while k <= 5:    # проверка условия
    print(k)
    k = k + 1    # модификатор условия проверки
```

Пример 2. Вычисление суммы целых чисел из промежутка [1, 10]:

$S = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10.$

```
k = 1           # начальные установки
S = 0
while k <= 10:      # проверка условия
    S = S + k       # вычисление суммы
    k = k + 1       # увеличение слагаемого k
print(S)
```

Пример 3. Печать таблицы значений функции $f(x)=\frac{x+1}{2\cdot(x^2+1)}$ **на отрезке** $x\in[0,1]$ **с шагом 0.1:**

```
x = 0       # x – текущая точка (от 0)
print('| x | f |')      # Шапка таблицы
print('-' * 21)
while x <= 1:       # Идем до 1
    f = (x + 1) / 2 / (x**2 + 1)
    print('| %4.2f | %10.7f |' % (x, f))
    x += 0.1 # Переход с шагом 0.1
print('-' * 21)
```

Часть II. Цикл for – in (перебор элементов итерируемого объекта)

```
for целевой список in итерируемый объект:
        операторы
[else:
        операторы ветки иначе]
```

Оператор for используется для перебора элементов итерируемого объекта (например, строки, кортежа, списка или др.). В качестве «целевого списка» обычно используется либо единственная переменная, либо последовательность переменных. В цикле по очереди перебираются элементы итерируемого объекта, на каждой итерации они присваиваются переменной стоящей между for и in, и выполняются операторы внутри цикла.

Есть необязательная часть else. Выполняется один раз после завершения всех итераций.

Вместе с циклом for часто применяют **итератор range**. Итератор – это особый вид функции, генерирует очередной элемент по требованию.

Можно вызвать с 1, 2 и 3 аргументами:

1) `range(stop)` – **числа от 0 до (stop – 1)**
2) `range(start, stop)` – **числа от start до (stop – 1)**

3) `range(start, stop, step)` – **от начала start до конечного числа stop (не включая stop) с шагом step**

Пример. Создание списков с помощью range:

```
list(range(5))          # вернет: [0, 1, 2, 3, 4]
list(range(-2,3))       # вернет: [-2, -1, 0, 1, 2]
list(range(0,10,2))     # вернет: [0, 2, 4, 6, 8]
list(range(1,-6,-2))    # вернет: [1, -1, -3, -5]
```

Пример. Применение range в цикле for

```
for i in range(10):  # числа: 0, 1, 2, ...8, 9
    print('Итерация', i)
```

Результат:

Итерация 0

…

Итерация 9

Примеры

Пример 1а. Вывод на экран чисел от 1 до 5:

```
for i in range(1, 6):
    print(i)
```

Пример 2а. Вычислить сумму первых 10 натуральных чисел:

$$S = \sum_{k=1}^{10} k = 1 + 2 + 3 + \cdots + 10$$

```
S = 0       # Инициализация суммы
for k in 1, 2, 3, 4, 5, 6, 7, 8, 9, 10: # Или range(…)
    S += k          # Вычисление суммы
print('Сумма = ', S)
```

Пример 3а. Печать значений функции $y = 2^x$ на отрезке $x \in [-5,5]$ с шагом 0.5:

```
for k in range(-10, 11):
    x = k / 2       # вычисление x
    y = 2 ** x
    print('x = % 5.2f | y = % 8.4f ' % (x, y))
```

Пример 4. Пользователь вводит последовательность из n целых чисел. Программа находит максимальное из этих чисел.

```
n = int(input('Количество чисел в последовательности (n >
0): '))
# Первый элемент записываем как максимальный:
```

```
max = int(input('a[1] = '))
for i in range(2, n + 1):
    # Считываем остальные (с 2 по n):
    a = int(input('a[' + str(i) + '] = '))
    # Если очередной больше max, он становится max:
    if a > max: max = a
print('Максимум:', max)
```

Работа программы:

Количество чисел в последовательности (n > 0): 5
a[1] = 5
a[2] = 7
a[3] = 3
a[4] = –2
a[5] = 1
Максимум: 7

В дальнейшем циклы будут применяться для обхода элементов различных структур, например, строк, кортежей или списков:

```
S = 'abcd'   # строка, тип str (или двойные кавычки "abcd")
T = (1, 2, 3)    # кортеж, тип tuple (можно T = 1, 2, 3  )
L = [4, 'AB', True]       # список, тип list (изменяемый тип)
# Есть обращение по индексу (нумерация начинается с 0):
S[0], T[1]   # 'a', 2
# Срезы:
S[1:3]   # 'bc'
# Проверка на вхождение (in   и   not in)
'abc'   in   '12abc3'          #  вернет  True
'ABC'   not in   '12abc3'      #  вернет  True
# функция определения размера len( )
len(L)   # 3
# Конкатенация (объединение) + и повторение *
t1 = (1, 2); t2 = (3, 4)       # создали 2 кортежа
t3 = t1 + t2                   # объединили t3 = (1, 2, 3, 4)
t4 = t1 * 3                    # t4 = (1, 2, 1, 2, 1, 2)
# Встроенные функции:
print(all(t1))             #  все истина?   = True
print(any(t1))             #  любой элемент истина? = True
print(max(t1))             #   максимум из (1, 2) = 2
print(min(t1))             #   минимум из  (1, 2) = 1
```

```
print(sum(t1))              #   сумма из (1, 2) = 3
# Обход и печать символов строки в цикле for:
for ch in 'abc':
    print(ch, end = ' ')
```

Самостоятельная работа в компьютерном классе

Часть I. Используйте оператор цикла с предусловием while

1. Перепишите пример 1. Измените его так, чтобы на экран выводились числа:
   a) от 10 до 1
   b) от –5 до 5 и вместе с ними их квадраты
   c) от 5 до –5 и вместе с ними их третьи степени
   d) от –100 до 100 с шагом 10: –100, –90, –80, … , 80, 90, 100.
2. Перепишите пример 2. Измените его так, чтобы вычислялись:
   a) сумма $S = 1 + 2 + 3 + \cdots + 20$
   b) сумма $S = 1 + 3 + 5 + 7 + 9$
   c) сумма $S = 1 + 5 + 10 + 15 + 20 + \cdots + 50$
   d) сумма $S = 1 + 2^2 + 3^2 + 4^2 + 5^2$
   e) произведение $P = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10$
   f) произведение $P = 2 * 4 * 6 * 8 * 10$
   g) произведение $P = 1 * 2^2 * 3^2 * 4^2 * 5^2$
3. Перепишите пример 3. Измените его так, чтобы печатались таблицы значений
   a) функции $f(x) = x^2$ на отрезке $x \in [-1, 1]$ с шагом 0.1
   b) функции $f(x) = \mid x \mid$ на отрезке $x \in [-5, 5]$ с шагом 0.5
   c) функции $f(x) = \sin(x)$ на отрезке $x \in [-\pi, \pi]$ с шагом 0.1
   d) функции $f(x) = \ln(x)$ на отрезке $x \in [0.1, 10]$ с шагом 0.3

---

4. Подсчитать количество цифр в записи натурального числа n.
5. Составьте программу, которая вводит натуральное число и выводит на экран все его цифры.
6. Проверить, есть ли в записи натурального числа n цифра 5.
7. Проверить, есть ли в записи натурального числа n цифры 3 или 7.
8. Проверить, есть ли в записи натурального числа n цифра 0 и 5.
9. Проверить, есть ли в записи натурального числа n нечетные цифры.
10. Проверить, все ли цифры в записи натурального числа n четны.

---

11. Найти и выдать на экран все двухзначные натуральные числа, в десятичной записи которых есть цифра 7.
12. Найти и выдать на экран все трехзначные натуральные числа, в десятичной записи которых есть цифра 5.

13. Дано натуральное число N. Вывести на экран все его делители.
14. Составьте программу, которая проверяет, является ли натуральное число простым, и выдает на экран сообщение.
15. Программа «Угадай число» (Вариант 3). Программа генерирует случайное число от 1 до 10. С помощью цикла while, напишите программу, в которой пользователь угадывает число, пока не угадает. В конце ему выдается сообщение, сколько попыток он использовал. После ввода очередного числа, программа сообщает ему, больше или меньше его число загаданного, например, так: «Введенное число больше загаданного». Можно добавить дополнительное ограничение: не более 10 попыток.

Часть II. Используйте цикл for – in

16. Перепишите пример 1a. Измените его так, чтобы на экран выводились числа:
    a) от 1 до 10
    b) от 10 до 1
    c) от –5 до 5
    d) от 0 до 10 с шагом 2: 0, 2, 4, 6, 8, 10.
    e) от –100 до 100 с шагом 10: –100, –90, –80, … , 80, 90, 100.
17. Перепишите пример 2a. Измените его так, чтобы вычислялись:
    a) сумма $S = 1 + 2 + 3 + \cdots + 20$
    b) сумма $S = 1 + 5 + 10 + 15 + 20 + \cdots + 50$
    c) сумма $S = 1 + \frac{1}{2^3} + \frac{1}{3^3} + \frac{1}{4^3} + \ldots + \frac{1}{50^3}$
    d) произведение $P = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10$
18. Перепишите пример 3a. Измените его так, чтобы печатались таблицы значений
    a) функции $f(x) = \cos(x)$ на отрезке $x \in [-4, 4]$ с шагом 0.2
    b) функции $f(x) = 5^{2x}$ на отрезке $x \in [-5, 5]$ с шагом 0.5

---

19. Дано натуральное число n. Вывести на экран все его делители.
20. Составьте программу, которая проверяет, есть ли в записи натурального числа n цифра 5, и выдает на экран сообщение.
21. Составьте программу, которая находит и выводит на экран все двухзначные натуральные числа, в десятичной записи которых есть цифра 5.
22. Подсчитать количество четырехзначных натуральных чисел, в десятичной записи которых есть цифра 0.

---

23. Пользователь вводит последовательность из n целых чисел. Программа находит максимальное по модулю из этих чисел.

24. Пользователь вводит последовательность из n целых чисел. Программа находит минимальное из этих чисел.
25. Пользователь вводит последовательность из n целых чисел. Программа находит количество положительных чисел в последовательности.
26. Пользователь вводит последовательность из n целых чисел. Программа находит, есть ли среди них двузначные.
27. Пользователь вводит последовательность из n целых чисел. Программа находит, все ли они положительны.

Занятие 4. Функции, Строки

Часть I. Функции

Пример 1. Вычислите выражение (sign(x) + sign(y))*sign(x + y). При вычислении задачи определить и использовать функцию sign:

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

```
def sign(x):
    if x < 0: return -1
    elif x == 0: return 0
    else: return 1

# Ввод данных:
x = float(input('x = '))
y = float(input('y = '))
# Вычислить выражение:
res = (sign(x) + sign(y)) * sign(x + y)
# Результат:
print('res = %7.3f' % res)
```

Результат работы программы:

x = 5

y = 7

res = 2.000

Пример 2. Функция вычисляет min и max из 3-х чисел (возврат нескольких значений).

```
def minmax(a, b, c):
    _max = _min = a
    if b < _min: _min = b
    if c < _min: _min = c
    if b > _max: _max = b
    if c > _max: _max = c
    return _min, _max      # - возврат 2-х значений

print(minmax(5, 2, 7))     # Результат кортеж: (2, 7)
# Присваивание последовательностей:
m1, m2 = minmax(-1.2, 20.5, -7.7)
print('min =', m1)         # Результат: -7.7
print('max =', m2)         # Результат: 20.5
```

Пример 3. Найти все трехзначные натуральные числа, у которых сумма цифр равна 7 (сумма цифр – функция).

```
def sum_digits(n):
    return n // 100 + n // 10 % 10 + n % 10

# Основная программа:
for n in range(100, 1000):
  # В цикле вызываем функцию на каждой итерации:
    if sum_digits(n) == 7: print(n, end = ', ')
```

Результат работы программы:

106, 115, 124, 133, 142, 151, 160, 205, 214, 223, 232, 241, 250, 304, 313, 322, 331, 340, 403, 412, 421, 430, 502, 511, 520, 601, 610, 700,

Пример 4. Определяем функцию, вычисляющую расстояние между двумя точками на плоскости по формуле (евклидово расстояние):

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Берем по умолчанию значения точек равные (0, 0). Используем разные способы вызова: позиционный, по имени, с использованием значений по умолчанию.

```
# Функция со значениями по умолчанию:
def distance(x1 = 0, y1 = 0, x2 = 0, y2 = 0):
    return ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5

# Вызовы функции:
# Со значениями по умолчанию:
print(distance())       # Вернет 0

# Задаем 1-ю точку: (x1 = 3, y1 = 4),
# вторая точка по умолчанию (x2 = 0, y2 = 0):
print(distance(3, 4))   # Вернет 5

# Задаем по имени x1 и x2, y1 и y2 - по умолчанию.
#(x1 = 3, y1 = 0), (x2 = -4, y2 = 0):
print(distance(x1 = 3, x2 = - 4)) # Вернет 7

# Задаем (x1 = 1, y1 = 0), (x2 = 0, y2 = 1).
print(distance(1, y2 = 1)) # Вернет 1.4142135623730951
```

Самостоятельная работа в компьютерном классе
Часть I. Функции

1. Даны два натуральных числа. Выяснить, в каком из них сумма цифр больше. Определить функцию для расчета суммы цифр натурального числа.
2. Даны два натуральных числа. Выяснить, в каком из них больше цифр. Определить функцию для расчета количества цифр натурального числа.
3. Найти значение выражения

$$c_n^k = \frac{n!}{k! \cdot (n-k)!},$$

при $0 \le k \le n$, и $n!$ означает факториал числа n. Определить функцию для расчета факториала числа: $n! = 1 \cdot 2 \ldots \cdot n$, для $n > 0$ и $0! = 1$.

Функции в этом блоке имеют 3 входных параметра-числа и возвращают 2 значения в return (см. пример 2).
Используйте значения по умолчанию в заголовке функции. Примените разные способы вызова: позиционный, по имени, с использованием значений по умолчанию (см. пример 4).

4. Находит сумму и произведение трех чисел.
5. Находит первый и второй максимум из трех чисел трех чисел ($\max_2 \le \max_1$). Например, для чисел 5, 7, 10, результат: $\max_2 = 7$ $\max_1 = 10$.
6. Среднее арифметическое и геометрическое трех чисел (Среднее арифметическое , Среднее геометрическое).

---

7. Найти все трехзначные простые числа. Определить функцию проверки числа на простоту.
8. Найти все четырехзначные числа палиндромы. Определить функцию, позволяющую распознавать числа-палиндромы.
9. Получить все шестизначные счастливые номера. Счастливым называют такое шестизначное число, в котором сумма его первых трех цифр равна сумме его последних трех цифр. Определить функцию для расчета суммы цифр трехзначного числа.

Часть II. Строки

Срезы. Примеры

```
# Определяем строку:
s = '0123456789'
print(len(s))                # 10 – длина строки
# С одним параметром:
print(s[0])                  # Вернет 0
print(s[15])                 # ошибка! Выход за границы
s[1] = 'a'; print(s)         # ошибка! Нельзя менять

# С двумя параметрами:
print(s[0:1])            # 0
print(s[15:])       # пустая строка (выход за границы – можно)
print(s[2:5])            # 234
print(s[0:len(s) - 1])  # 012345678
print(s[:len(s)])        # 0123456789
print(s[-20:20])             # 0123456789
print(s[:])                  # 0123456789
print(s[-len(s): -1])        # 012345678
print(s[-len(s): ])          # 0123456789
print(s[-7: 7])              # 3456  ( == s[len(s)-7: 7])

# С тремя параметрами:
print(s[::2])            # 02468
print(s[1::2])           # 13579
print(s[::-1])           # 9876543210 (переворот строки)
print(s[::-2])           # 97531
print(s[::])    # 0123456789 или ошибка (зависит от версии!)
print(s[3:7:2])          # 35
print(s[-7:-4:2])        # 35
s = s[len(s) // 2:] + s[:len(s) // 2]
print(s)                 # 5678901234
```

Сравнение и коды символов, chr, ord, min, max

Пример. **Символы с кодами от 34 до 128:**

```
for i in range(34, 129):
    print('%d %s' % (i, chr(i)), end = ' | ')
```

34 " | 35 # | 36 $ | 37 % | 38 & | 39 ' | 40 (| 41) | 42 * | 43 + | 44 , | 45 - | 46 . | 47 / | 48 0 | 49 1 | 50 2 | 51 3 | 52 4 | 53 5 | 54 6 | 55 7 | 56 8 | 57 9 | 58 : | 59 ; | 60 < | 61 = | 62 > | 63 ? |

64 @ | 65 A | 66 B | 67 C | 68 D | 69 E | 70 F | 71 G | 72 H | 73 I | 74 J | 75 K | 76 L | 77 M | 78 N | 79 O | 80 P | 81 Q | 82 R | 83 S | 84 T | 85 U | 86 V | 87 W | 88 X | 89 Y | 90 Z | 91 [| 92 \ | 93] | 94 ^ | 95 _ | 96 ` | 97 a | 98 b | 99 c | 100 d | 101 e | 102 f | 103 g | 104 h | 105 i | 106 j | 107 k | 108 l | 109 m | 110 n | 111 o | 112 p | 113 q | 114 r | 115 s | 116 t | 117 u | 118 v | 119 w | 120 x | 121 y | 122 z | 123 { | 124 | | 125 } | 126 ~ | 127 □| 128 □ |

Пример. **Русские буквы:**

```
# Большие:
for i in range(ord('А'), ord('Я') + 1):
print('%d %s' % (i, chr(i)), end = ' | ')
i = 'Ё'
print('%d %s' % (ord(i), i))
```

1040 А | 1041 Б | 1042 В | 1043 Г | 1044 Д | 1045 Е | 1046 Ж | 1047 З | 1048 И | 1049 Й | 1050 К | 1051 Л | 1052 М | 1053 Н | 1054 О | 1055 П | 1056 Р | 1057 С | 1058 Т | 1059 У | 1060 Ф | 1061 Х | 1062 Ц | 1063 Ч | 1064 Ш | 1065 Щ | 1066 Ъ | 1067 Ы | 1068 Ь | 1069 Э | 1070 Ю | 1071 Я | 1025 Ё

```
# Маленькие:
for i in range(ord('а'), ord('я') + 1):
print('%d %s' % (i, chr(i)), end = ' | ')
i = 'ё'
print('%d %s' % (ord(i), i))
```

1072 а | 1073 б | 1074 в | 1075 г | 1076 д | 1077 е | 1078 ж | 1079 з | 1080 и | 1081 й | 1082 к | 1083 л | 1084 м | 1085 н | 1086 о | 1087 п | 1088 р | 1089 с | 1090 т | 1091 у | 1092 ф | 1093 х | 1094 ц | 1095 ч | 1096 ш | 1097 щ | 1098 ъ | 1099 ы | 1100 ь | 1101 э | 1102 ю | 1103 я | 1105 ё

Коды символов:

коды:	**48–57**	**65–90**	**97–122**	**1025**	**1040–1071**	**1072–1105**	**1105**
симво-лы:	**‘0’–‘9’**	**‘A’–‘Z’**	**‘a’–‘z’**	**‘Ё’**	**‘А’–‘Я’**	**‘а’–‘я’**	**‘ё’**

Строки можно сравнивать (< <= > >= == !=). Сравниваются коды символов.

```
'абв' == 'абв'      # True
'абв' > 'АБВ'       # True
'абв' > 'аб'        # True
max('z1+яЯёЁ')      # 'ё'  (возвращает символ с max-м кодом)
min('z1+яЯёЁ')      # '+'  (возвращает символ с min-м кодом)
```

Основные методы строк

s.find(t, start, end)	Возвращает позицию самого первого (крайнего слева) вхождения подстроки t в строку s (или в срез строки s[start:end]); если подстрока t не найдена, возвращается число –1
s.rfind(t, start, end)	Поиск самого последнего (крайнего справа) вхождения подстроки (или в срез строки s[start:end]); если подстрока t не найдена, возвращается число –1

Если надо просто узнать, входит ли одна строка в другую, то можно использовать t in s.

Пример 1. Из строк вида "26.11.2020" или "1.1.21" будем вырезать число, месяц, год:

```
s = input('dd.mm.yyyy:')
date = s[:s.find('.')]
month = s[s.find('.') + 1 : s.rfind('.')]
year = s[s.rfind('.') + 1:]
print('дата: ', date)
print('месяц: ', month)
print('год: ', year)
```

Пример 2. Ищем вхождение строки с пересечениями. Например, строка 'aa' входит в строку 'aaa aa' 3 раза (без пересечений 2 раза найдет метод count).

```
s = input('s = ')
subs = input('subs = ')
count_substring = 0          # количество вхождений подстрки
pos = 0                      # текущая позиция
pos = s.find(subs, pos)      # ищем начиная с pos и до конца
while pos != -1:
    count_substring += 1
    pos = s.find(subs, pos + 1) # начиная с (pos + 1) и до
конца
print('Количество вхождений: ', count_substring)
```

s.replace(t, u, n)	Заменяет в строке s, каждое (но не более n, если этот аргумент определен) вхождение подстроки t на подстроку u

Пример 1. Заменим все вхождения строки

```
s = ' Пример строки '
subs = ' '      # пробелы
newsubs = '*'   # будем менять на звездочки
s = s.replace(subs, newsubs) # обязательно нужно присваивать
print(s)
```

Примерстроки*

Пример 2. Заменим одно вхождения строки

```
verse = '''Наша Таня громко плачет:
Уронила в речку мячик.
- Тише, Танечка, не плачь:
Не утонет в речке мяч.'''
# Заменяем только первое вхождение:
verse_new = verse.replace('мяч', 'МЯЧ', 1)
print(verse_new)
```

Наша Таня громко плачет:
Уронила в речку МЯЧик.
- Тише, Танечка, не плачь:
Не утонет в речке мяч.

Пример 3. Удаление всех вхождений подстроки

```
s = '1101011101'
subs = '101' # удалим все 101
s = s.replace(subs, '') # замена на пустую строку
print(s)
```

101

Пример 4. Напишем свою замену одного вхождения с помощью поиска и срезов

```
s = input('s = ')
subs = input('subs = ')
newsubs = input('newsubs = ')
pos = s.find(subs)      # Поиск подстроки
if pos >= 0:            # Если нашли, то заменяем
    s = s[:pos] + newsubs + s[pos + len(subs):]
print(s)
```

s.count(t, start, end)	Возвращает число вхождений строки t в строку s (или в срез строки s[start:end])

Пример. Ищет без пересечений, т.е. в строку ‘аааааа’ подстрока ‘аа’ входит 3 раза

```
'aaaaaa'.count('aa')                    # вернет 3
# Сколько нулей в левой половине строки:
s = '10101101011001010101'
s.count('0', 0, len(s) // 2)            # вернет 4
```

s.join(seq)	Объединяет все элементы последовательности строк seq, вставляя между ними строку s
s.split(sep = None, maxsplit = –1)	Возвращает список слов в строке, используя sep в качестве строки-разделителя. Если задан maxsplit, то выполняется не более maxsplit расщеплений. Если задан sep, то последовательные разделители не группируются вместе и считаются разделителями пустых строк (например, '1,,2'.split(',') возвращает ['1', '', '2']). Аргумент sep может состоять из нескольких символов (например, '1<>2<>3'.split('<>') возвращает ['1', '2', '3']). Если sep не указан или отсутствует, применяется другой алгоритм разбиения: несколько последовательных пробелов рассматриваются как один разделитель, и результат не будет содержать пустых строк. Есть еще функция **rsplit**

Пример 1. Разбиваем строку на слова (split)

```
s = ' abc 123 АВ АБВГ0 '
L = s.split(' ')    # Разбиваем на слова по пробелам
print(L)
# ['', '', 'abc', '', '', '123', 'АВ', '', 'АБВГ0', '', '']
# Лучше так:
L = s.split()  # Разбиваем на слова
print(L)       # ['abc', '123', 'АВ', 'АБВГ0']
# Если не пустое слово и состоит лишь из цифр, выведем его:
for slovo in L:
    if slovo.isdigit(): print(slovo)
```

Результат:

123

Пример 2. **Объединяем слова в одну строку (join)**

```
res = '*'.join(['123', ' abc', ' ', 'AB']) # список или кортеж
print(res)      # Результат: 123* abc* *AB
```

Самостоятельная работа в компьютерном классе

Часть II. Строки

Замечание 1. Одно и то же задание для строк часто можно сделать несколькими способами. Например, посимвольно проходя строку в цикле или используя срезы или встроенные методы строк. Рекомендуется делать задания разными способами, можно одно и то же задание сделать несколькими способами.

Замечание 2. Основной алгоритм можно оформлять в виде функции.

В этом разделе используйте срезы строк.

1. Дана строка. Определить, является ли строка палиндромом.
2. Дана строка. В строке с помощью срезов поменяйте первую и вторую половину. Если в строке нечетное количество символов, то 1-я половина большая (например: '123ab' → 'ab123').
3. Дана строка. В строке с помощью срезов поменяйте 2-ю букву на букву 'A' и удалите последнюю букву (примеры: '12345' → '1A34', 'abababab' → 'aAababa').
4. Дана строка. С помощью срезов переверните ее и удалите первый символ (пример: '12345' → '5432').
5. Дана строка. В строке с помощью срезов выделите только четные символы (пример: 'абвгд' → 'авд').

В этом разделе используйте методы строк.

6. В начале и конце строки удалить символы: пробелы, табуляции \t, переноса \n.
7. В начале и конце строки удалить символы: ', . : ; ! ?'.
8. В строке удалить все пробелы и все буквенные символы привести к нижнему регистру.

---

9. В строке заменить все символы табуляции \t и переноса \n на пробелы.
10. В строке посчитать количество латинских букв “Aa”.
11. В строке найти позицию первого вхождения подстроки “cat”.
12. Продублировать в строке все символы “/”. Например, из строки “/temp/1” сделать строку “//temp//1”.

---

13. Найти в строке позицию первого справа латинского символа. Если латинского символа в строке нет, выдать сообщение об этом.

14. Удалить из строки все символы-цифры (‘0’…‘9’).
15. Определить, чего больше в строке: последовательности символов ‘dog’ в ее первой половине или последовательности символов ‘cat’ во второй? (Например, в строке ‘dogdogdogcat’ 2 – dog и 1 – cat). В ответе указать больше, меньше или равно.
16. Даны две строки s1 и s2. Найти позицию начала первого появления в строке s1 строки s2 или перевернутой строки s2. Например, для строк s1 = '03400430' и s2 = ‘43’ вернет позицию 1. Если вхождения нет, выдать сообщение об этом.
17. Дана строка, состоящая из малых латинских символов. Посчитать сколько всего различных символов в ней есть. Например, в строке ‘abcabab’ всего 3 разных символа ‘abc’).

В этом разделе вводится строка, состоящая из слов, разделенных пробелами. Для разбиения на слова удобно использовать метод split. Например, для строки s = ' abc 123 AB' следующий код L = s.split( ) вернет список L = ['abc', '123', 'AB'].

18. Подсчитать количество слов в тексте.
19. Напечатать слова из текста, в которых есть большие буквы.
20. Напечатать слова из текста, в которых есть цифры.
21. Напечатать слова из текста, в которых есть только латинские символы.

Занятие 5. Списки

В Python одни и те же действия можно выполнить разными способами, поэтому некоторые примеры приведены с несколькими вариантами решений.

Создание списка

Пример 1 (В1). Заполнение списка с клавиатуры (for и append)

```
n = int(input('Количество элементов: '))
L = []          # В начале берем пустой список
for i in range(n):
    # Добавляем целое число:
    L.append(int(input('Элемент L[' + str(i) + '] = ')))

print(L)
```

Результат работы программы:

```
Количество элементов: 3
Элемент L[0] = 5
Элемент L[1] = -7
Элемент L[2] = 0
[5, -7, 0]
```

Пример 1 (В2). Заполнение списка с клавиатуры (for и L[i] =)

```
n = int(input('Количество элементов: '))
L = [None] * n     # Создаем список из n элементов
for i in range(n):
    # Заполняем список с клавиатуры:
    L[i] = int(input('Элемент L[%d] = ' % i))
print(L)
```

Пример 1 (В3). Заполнение списка с клавиатуры (генератор)

```
n = int(input('Количество элементов: '))
L = [int(input('Элемент L[%d] = ' % i)) for i in
range(n)]
print(L)
```

Замечание. Для добавления можно использовать еще: методы L.extend(m), L.insert(i, x), L += m и срезы L[i:i] = [x].

Пример 2. Заполнение случайными числами **(модуль random)**

Некоторые функции из модуля random

https://docs.python.org/3/library/random.html?highlight=random#module-random

Функция	Описание
seed([n])	Инициализация генератора случайных чисел; без параметров – используется значение системного времени
randint(a, b)	Возвращает случайное целое в диапазоне [a, b]
choice(seq)	Возвращает случайный элемент из последовательности seq
shuffle(L)	Перемешивает элементы в списке L
random( )	Возвращает случайное вещественное число в диапазоне [0, 1)

Пример 2 (B1). Заполнение случайными числами

```
import random   # Подключаем модуль
random.seed( )  # Инициализация генератора
n = int(input('Количество элементов: '))
L = []
for i in range(n):
    # Получить случайное число из [-10, 10]:
    L.append(random.randint(-10, 10))
print(L)        # Результат работы программы:
```

```
Количество элементов: 10
[-5, -5, 10, -10, 7, 8, 4, 5, 2, -3]
```

Пример 2 (B2). Заполнение случайными числами (генератор)

```
import random   # Подключаем модуль
random.seed()   # Инициализация генератора
L = [random.randint(-10, 10) for i in
                    range(random.randint(5, 10))]
```

Линейный поиск

Методы	Описание
in, not in	Проверка на вхождение (операция)
L.count(x)	Число вхождений элемента x в список L
L.index(x [, start, end])	Индекс самого первого (слева) вхождения элемента x (или в срез start : end); нет – ValueError
L.remove(x)	Удаляет самый первый слева найденный элемент x из списка L; нет – ValueError *(поиск и удаление)*

Поиск 1 (Встроенные функции). Определить, содержится ли заданное число x в списке. Если содержится, вернуть его номер.

```
if x in L: ind = L.index(x) # два прохода списка
else: ind = -1 # - признак, что не нашли
print(ind)
```

[6, –8, 7, –7, 10, 1]
x = 7
2

Поиск 2 (Линейный поиск. В1). Определить, содержится ли заданное число x в списке. Если содержится, вернуть его номер.

```
ind = -1
for i in range(len(L)): # Поиск
   if L[i] == x:
      ind = i
      break
```

Поиск 2 (Линейный поиск. В2). С помощью while

```
i = 0;
while (i < len(L)) and (L[i] != x): i += 1   # Поиск
# Анализ результата:
if i == len(L): print('Числа нет в списке')
else: print('Номер', i)
```

Поиск 3 (Линейный поиск с барьером). В конец списка добавляем число x. Тогда можно убрать одну проверку (i < n) из цикла.

```
L.append(x)    # Добавляем барьер
i = 0;
while L[i] != x: i += 1 # Поиск
# Анализ результата:
```

```
if i == len(L) - 1: print('Числа нет в списке')
else: print('Номер', i)
# L.pop( )          # Удаляем барьер
```

Сравнение скорости работы на 3 000 000 эл. сп. (нет, посл., сред.):

Встроенные методы	**(1):**	**0.15 сек.**	**0.3 сек.**	**0.15 сек.**
for и break	**(2):**	**1.38 сек.**	**1.21 сек.**	**0.64 сек.**
while с двумя условиями	**(4):**	**2.01 сек.**	**2.13 сек.**	**1.07 сек.**
while и барьер	**(3):**	**1.6 сек.**	**1.6 сек.**	**0.81 сек.**

Для измерения используется функция default_timer( ) из модуля timeit.

Линейный поиск с двумя проходами по списку

Пример. Элементы списка L – целые числа. Определить, все ли они различны (легче проверять обратное условие: есть ли хотя бы одно совпадение; все → хотя бы один).

```
b = True  # По умолчанию все различны
i = 0     # берем i-й элемент от 0 до последнего:
while i < len(L) and b:
    j = i + 1  # и сравниваем с j-м элементом
    while j < len(L) and b:
        if L[i] == L[j]: b = False   # совпадение!
        j += 1
    i += 1
if b: print('Все различны')
else: print('Есть совпадения:', i-1, j-1)
```

```
[-9, 7, 3, 3, 9, 3]
Есть совпадения: 2 3
```

Замечание. Есть еще поиск одного списка в другом (известны разные оптимизированные алгоритмы).

Самостоятельная работа в компьютерном классе

Основной алгоритм в заданиях можно оформлять в виде функций

Дан список целых чисел.

1. Найти и вывести на экран произведение элементов списка.
2. Найти и вывести на экран сумму положительных элементов списка.
3. Вывести на экран сначала все положительные элементы списка, потом все остальные.
4. Подсчитать количество отрицательных элементов.
5. Вывести на экран те из них, которые являются четными.
6. Определить, делится ли хотя бы один элемент на заданное целое число.

Дан список. С помощью срезов и операций +, * выполните следующие задания.

7. Дан список. Создайте новый список, заполненный нулями такой же длины как исходный.
8. Замените нулями элементы в списке, начиная с определенной позиции и до конца списка.
9. Добавьте новый элемент в конец списка.
10. Добавьте пять нулей в начало списка.
11. Удалите последний элемент из списка.
12. Удалите из списка элемент с заданным номером.

Создание списка.

13. Создайте список целых чисел, с помощью цикла for и метода append (заполнение случайными числами). После этого удалите из этого списка максимальный элемент.
14. Создайте список целых чисел, с помощью цикла for и присваивания очередного элемента L[i] (данные вводятся с клавиатуры пользователем). После этого замените в списке все отрицательные числа на число ноль.
15. Создайте список целых чисел, с помощью цикла for и метода append (данные вводятся с клавиатуры пользователем). После этого удалите из этого списка последний отрицательный элемент (если отрицательного нет, выдать сообщение).
16. Создайте список натуральных чисел, с помощью цикла for и присваивания очередного элемента L[i] (заполнение случайными числами). После этого замените в списке все четные числа на число ноль.

Создание списка с помощью генератора.

17. Создайте список целых чисел, с помощью генератора списков (данные вводятся с клавиатуры пользователем). Напишите генератор с условием: добавлять только положительные числа.

18. Создайте список целых чисел, с помощью генератора списков (заполнение случайными числами).
19. С помощью генератора списков, создайте список квадратов от 1 до 10: 1, 4, 9, 16 и т.д.

Основной алгоритм оформите в виде функции

20. Удалить из целочисленного списка все четные элементы. Функция создает новый список и возвращает его в return. Старый список не меняет (см. в лекции Способ 1., стр. 22, 23).
21. Удалить из целочисленного списка все отрицательные элементы. Внутри функции создать копию списка и удалять из копии. Старый список не меняет (см. в лекции Способ 2., стр. 23–25).
22. Удалить из целочисленного списка все элементы с максимальным значением (предполагается, что имеется несколько таких элементов). Функция преобразует исходный список, return не нужен (см. в лекции Способ 3., стр. 25, 26).
23. Вставить в целочисленный список перед каждым четным элементом цифру ноль. Функция создает новый список и возвращает его в return. Старый список не меняет (см. в лекции Способ 1., стр. 22, 23).
24. Вставить в одномерный целочисленный список после каждого нечетного элемента цифру ноль. Функция преобразует исходный список, return не нужен (см. в лекции Способ 3., стр. 25, 26).

Основной алгоритм оформите в виде функции с логическим значением: функция возвращает в return True/False.

25. Элементы списка A – натуральные числа. Определить, есть ли среди элементов списка числа с одинаковой последней цифрой.
26. Элементы списков A и B – натуральные числа. Определить, верно ли, что каждый элемент списка A содержится в списке B.
27. Элементы списков A и B – натуральные числа. Определить, есть ли среди элементов списка A числа, отсутствующие в списке B.
28. Элементы списка A – натуральные числа. Определить, есть ли среди элементов списка пары чисел такие, что одно число является квадратом другого (т.е. A[i] == A[j] ** 2).

Напишите функцию с переменным количеством позиционных аргументов (см. лекцию стр. 31–34). Вызовите функцию с разным количеством параметров, используйте при вызове распаковку списка (см. лекцию стр. 29, 30).

29. Функция находит сумму произвольного количества переданных ей чисел.
30. Функция находит произведение произвольного количества переданных ей чисел.
31. Функция печатает все переданные ей данные, каждую переменную с новой строки.

32. Функция переводит все переданные ей данные в одну строку и возвращает ее в качестве результата. Для перевода используйте функцию str.

Занятие 6. Поиск, сортировка, двумерные списки

Самостоятельная работа в компьютерном классе

Основной алгоритм в заданиях можно оформлять в виде функций

Дан список чисел. Сортировка списка методом sort (см. Лекцию 10, стр.33–37)

1. Отсортировать список по возрастанию младшей цифры.
2. Отсортировать список по убыванию.
3. Отсортировать список по возрастанию модулей.
4. Отсортировать список по убыванию модулей.
5. Отсортировать список по возрастанию суммы цифр.
6. Отсортировать список по убыванию количества цифр.

Дан упорядоченный по возрастанию список. Поиск в упорядоченном списке (модуль bisect, см. Лекцию 10, стр. 30–32)

7. Найдите в списке место для вставки нового числа x, не нарушающего его упорядоченность (метод bisect).
8. Найдите в списке место для вставки нового числа x, не нарушающего его упорядоченность (метод bisect_right).
9. Вставить в список новое число x, не нарушая его упорядоченность (метод insort).
10. Вставить в список новое число x, не нарушая его упорядоченность (метод insort_right).

Создание двумерного списка A[n, m] (см. Лекцию 11)

11. Создайте двумерный список целых чисел с помощью двух циклов for и метода append (заполнение случайными числами).
12. Создайте двумерный список целых чисел с помощью вложенных генераторов (данные вводятся с клавиатуры пользователем).
13. Создайте двумерный список целых чисел с помощью двух циклов for и метода append (данные вводятся с клавиатуры пользователем).
14. Создайте двумерный список целых чисел с помощью вложенных генераторов (заполнение случайными числами).

Поиск в двумерном списке. Дан двумерный список целых чисел A[n, m]

15. Найти номер строки с максимальной суммой элементов.
16. Найти номер столбца с минимальной суммой элементов.
17. Определить, есть ли в нем столбец, сумма чисел в котором равна нулю. Если есть, вернуть номер этого столбца, если нет – выдать сообщение.
18. Определить есть ли в нем строка, состоящая только из нулей. Если есть, вернуть ее номер, если нет – выдать сообщение.
19. Определить есть ли в нем число x. Если есть, вернуть номер строки и столбца этого числа, если нет – выдать сообщение.

20. Определить есть ли в нем столбец, состоящий только из отрицательных чисел. Если есть, вернуть его номер, если нет – выдать сообщение.

Изменение двумерного списка (матрицы) A[n, m]

21. Поменять местами в матрице A две строки с заданными номерами.
22. Поменять местами в матрице A два столбца с заданными номерами.
23. В матрице удалить строку с заданным номером.
24. В матрице удалить столбец с заданным номером.
25. В матрицу добавить строку в заданную позицию. Позицию и данные для строки спросить у пользователя.
26. В матрицу добавить столбец (состоящий из нулей) в заданную позицию.

Занятие 7. Множества и словари

Часть I. Множества set

1. Создать множество символов из строки, введенной пользователем.
2. Создать множество из целых чисел. Числа вводятся пользователем.
3. Создать множество цифр из строки, введенной пользователем.

---

4. Даны два множества. Найдите их пересечение, объединение, объединение без пересечений и разность с помощью встроенных операций.
5. Даны два множества. Найдите их пересечение, объединение, объединение без пересечений и разность с помощью стандартных методов множеств.
6. Даны три множества A, B, C. Найдите значения выражений: $A \cup B \cup C$, $A \cap B \cap C$ и $(A \cup B) \cap C$.

С помощью генераторов множеств, создайте следующие множества.

7. Множество целых чисел из промежутка $[1, 100]$ делящихся на 3 или 7.
8. Множество натуральных чисел, меньших 1000, которые можно представить в виде $n^2 + m^2$, где $n, m \geq 0$.
9. Множество из 5-ти случайных чисел из промежутка $[-20, 20]$. Используйте модуль random.

Дана строка (вводится пользователем).

10. Подсчитайте, сколько различных английских букв в нее входят.
11. Определите, все ли кириллические буквы в ней есть.
12. Верно ли, что в ней имеются буквы, входящие в слово «шина».
13. Верно ли, что в ней имеются буквы, входящие в слово задаваемое пользователем.
14. Верно ли, что в ней имеются все буквы из слова «шина».
15. Верно ли, что в ней имеются все буквы из слова, задаваемого пользователем.

Дана строка (вводится пользователем).

16. Найдите наибольшее количество цифр, идущих подряд.
17. Найдите символы, входящие в строку не менее двух раз.
18. Найдите символы, входящие в строку только один раз.

Дана строка, состоящая из слов и пробелов.

19. Выведите на экран те буквы, которые встречаются в каждом слове строки только один раз.
20. Выведите на экран гласные буквы, которые входят в каждое слово.
21. Выведите на экран согласные буквы, которые не входят ни в одно слово.
22. Выведите на экран гласные буквы, которые не входят хотя бы в одно слово.
23. Выведите на экран гласные буквы, которые не входят только в одно слово.
24. Выведите на экран гласные буквы, которые не входят более чем в одно слово.

Часть II. Словари dict

1. Создать словарь хранящий данные о человеке, со следующими ключами: ‘имя’, ‘фамилия’, ‘дата’, ‘месяц’ и ‘год’. Значения спросить у пользователя.
2. Создать словарь такого типа: {1: ‘Pascal’, 2: ‘Python’, … }. У пользователя спросить количество элементов словаря. Ключи – это порядковые номера, значения – данные, вводимые пользователем.
3. Создать словарь хранящий данные о студенте, со следующими ключами: ‘ФИО’, ‘курс’, ‘группа’. Значения спросить у пользователя.

---

4. Дан список целых чисел. Создать словарь, в который входят пары (ключ/значение): число из списка и количество повторений числа. Например, для списка L = [1, 2, 1, 2, 1, 2, 1] создается словарь: D = {1:4, 2:3}.
5. Дана строка. Подсчитать сколько раз в строке встречается каждый символ (создавать словарь по строке с парами: символ – количество).
6. Дана строка. Подсчитать сколько раз в строке встречается каждое слово (создавать словарь по строке с парами: слово – количество).

С помощью генераторов словарей, создайте следующие словари.

7. Словарь вида: {1 : ’1’, 2 : ’2’, … n : ‘n’}
8. Словарь, ключами которого являются номера от 1 до n, а значения – случайные числа.
9. Словарь, ключами которого являются строки, а значения – целые числа. Данные для словаря запрашивать у пользователя.

Функции с произвольным количеством именованных аргументов

10. Создайте функцию с произвольным количеством именованных аргументов. Она печатает все переменные, которые ей передаются, находит сумму значений и возвращает ее. Вызовите ее несколько раз с разными данными.
11. Создайте функцию с произвольным количеством именованных аргументов. Она печатает количество переданных ей переменных, собирает все значения в один список и возвращает его в качестве результата. Вызовите ее несколько раз с разными данными.
12. Создайте функцию с произвольным количеством именованных аргументов. Она находит сумму числовых значений (int и float), которые ей передаются, и возвращает ее. Для остальных нечисловых параметров она печатает имена переменных. Вызовите ее несколько раз с разными данными.

    Замечание. Для определения типа можно использовать функцию isinstance.

Лабораторная работа № 8
Часть I. Работа с текстовыми файлами

Замечание. Удобно работать с файлами, находящимися в одном каталоге с программой. Тогда не нужно указывать полный путь к файлу, достаточно только имя файла. Для этого нужно, чтобы каталог программы был текущим рабочим каталогом. В IDLE они совпадают. В Pyzo по умолчанию нет. Но можно настроить в меню Pyzo: “Run (Запуск) → Change directory when executing file”.

Создание файла, чтение данных из файла и вывод их на экран

1. Создайте произвольный текстовый файл Input.txt (с помощью приложения Блокнот или WordPad).
2. Создать файл из строк, введенных пользователем. Пользователь задает количество строк и вводит эти строки в программу. Программа из них создает новый текстовый файл.
3. Прочитать файл и вывести его содержимое на экран.

Чтение и обработка строк в файлах

4. Прочитать текст из файла и вывести на экран нечетные строки.
5. Прочитать текст из файла и вывести на экран те строки, в которых нет цифр.
6. Проверить, есть ли в файле строки, начинающиеся и оканчивающиеся одним и тем же символом.
7. Проверить, есть ли в файле строки, состоящие только из цифр.
8. Посчитать количество строк в файле.
9. Посчитать количество непустых строк в файле (в пустой строке могут быть только пробелы и символы перехода на следующую строку).

Работа с числами в текстовых файлах

10. Создать файл, содержащий целые числа. В файле создается случайное количество строк (от 1 до 5), в каждой строке случайное количество целых чисел (от 0 до 5). Числа в каждой строке, разделены пробелами.
11. Создать файл, содержащий все двузначные натуральные числа, делящиеся на заданное число m (каждое число в новой строке).
12. Создайте файл, содержащий случайное (не менее $n1$ и не более $n2$) количество случайных целых чисел из промежутка $[-m, m]$. Числа в файле разделены пробелами.

Создайте текстовый файл, содержащий целые числа, разделенные пробелами в одной строке (с помощью приложения Блокнот или WordPad).

13. Чтение чисел из файла и вывод их на экран в одну строку.
14. Читать числа из файла и вывести их на экран каждое с новой строки.

15. Найти в файле максимальное число и вывести его на экран.
16. Подсчитать произведение положительных чисел из файла.
17. Чтение чисел из файла и вычисление суммы двузначных чисел.
18. Проверить, есть ли в файле числа, оканчивающиеся цифрой 3.

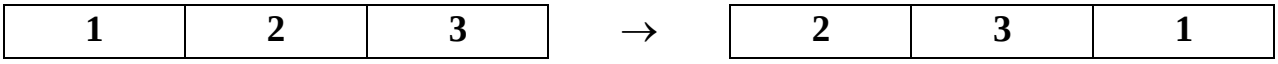
Создайте текстовый файл с целыми числами, находящимися в нескольких строках файла, разделенные пробелами внутри строк (с помощью приложения Блокнот или WordPad).

19. Чтение чисел из существующего файла и создание двух новых файлов: в один записываются четные числа, в другой – нечетные (каждое число с новой строки).
20. Чтение чисел из существующего файла и создание нового файла, в который записываются те числа, в десятичном представлении которых использованы только нечетные цифры (числа разделены пробелами в одной строке).
21. Создать два новых файла: в первый поместить положительные компоненты файла, а во второй все остальные (числа разделены пробелами в одной строке).
22. Создать новый файл, в который поместить только четные числа из файла (каждое число остается в строке с тем же номером; если в строке не было четных чисел, то в новом файле эта строка пустая).

Изменение текстового файла

В этих задачах можно использовать методы для работы с файлами на диске, например, erase и rename.

23. Дан текстовый файл. Удалите строку с заданным номером. Если строки с указанным номером нет, сообщите об этом.
24. Дан текстовый файл. Вставьте строку с заданным номером. Если строки с указанным номером нет, сообщите об этом.
25. Дан текстовый файл. Замените его на новый файл, в котором по отношению к исходному файлу меняются местами строки: первая и вторая, третья и четвертая, пятая и шестая и т.д.
26. Дан текстовый файл. Замените его на новый файл, в котором по отношению к исходному файлу меняются местами фрагменты по следующей схеме:



Количество строк в каждом фрагменте отличается не более чем на 1. Если при разделении на фрагменты количество строк в них разное, то фрагменты упорядочены по убыванию. Например, в файле с 8-ю строками: 3, 3 и 2 строки в фрагментах 1, 2 и 3, соответственно.

Лабораторная работа № 8
Часть II. Работа с файлами
Примеры
Хранение целых чисел в двоичных файлах

Пример 1. Перевод целого числа int в тип bytes и обратно (int ↔ bytes)

Двоичные файлы хранят данные в виде строк bytes. Целое число можно перевести в bytes с помощью метода to_bytes:

int.to_bytes(length, byteorder, *, signed = False)

Параметры	Описание
length	количество байт выделяемое на число (вызывается исключение, если нельзя записать число в указанное число байт)
byteorder	определяет порядок байтов, используемый для представления целого числа ('big' или 'little')
signed	число со знаком (True) или нет (False)

Наоборот, данные из bytes в int можно перевести с помощью метода from_bytes.

int.from_bytes(bytes, byteorder, *, signed = False)

Параметры	Описание
bytes	данные типа bytes
byteorder	порядок байтов представления целого числа ('big' или 'little')
signed	число со знаком (True) или нет (False)

Родной порядок байт системы возвращает sys.byteorder.

```
import sys
# Родной порядок байт:
print(sys.byteorder)                    # little
# Перевод числа 1024 в bytes (2 байта со знаком):
bx = int.to_bytes(1024, 2, byteorder = 'little',
signed = True)
print(bx)                               # b'\x00\x04'
# Обратный перевод числа 1024 из bytes (2 байта со знаком):
x = int.from_bytes(bx, byteorder = 'little', signed = True)
print(x)                                # 1024
```

Пример 2. Создание файла, содержащего случайное количество (от 3 до 10) случайных целых чисел из промежутка [–32768, 32767].

```
import random
random.seed()
```

```
n = random.randint(3, 10)
# Создаем/открываем битовый файл "numbers1.dat" для записи:
f = open("numbers1.dat", "wb")
for i in range(n):
 # Будем использовать формат числа 2 байта со знаком,
 # Максимально допустимый диапазон: [-32768, 32767].
   x = random.randint(-32768, 32767)
 # Перевод в bytes:
   bx = x.to_bytes(2, byteorder = 'little', signed = True)
 # Сохранение 2 байт в файл:
   f.write(bx)
f.close( )
```

Пример 3. Чтение целых чисел из битового файла и вывод их на экран

```
# Открываем битовый файл "numbers1.dat" для чтения:
f = open("numbers1.dat", "rb")
# читаем из него по 2 байта:
bx = f.read(2)
# Проверка на пустую строку bytes b'':
while bx != b'':
 # Перевод из bytes в число int:
   x = int.from_bytes(bx, byteorder='big', signed=True)
 # выводим числа на экран:
   print(x)
 # читаем из файла по 2 байта:
   bx = f.read(2)
f.close()
```

Задания

Чтение и запись текстовых файлов в разных кодировках

1. Создайте произвольный текстовый файл “text_encoding_1.txt” (с помощью приложения Блокнот или WordPad). Используйте в этом файле как латинские, так и кириллические буквы.
2. Посмотрите кодировку, установленную по умолчанию для работы с текстовыми файлами (f.encoding). Прочитайте созданный файл, используя разные кодировки (utf-8, cp1251 или другие), и выведите его содержимое на экран.

   Замечание. Для проверки в программу добавьте комментарий, например, “кодировка по умолчанию …, корректно все данные прочитались в кодировке …, в другой кодировке …”.
3. Создайте файл, содержащий разные символы Unicode (например, английские, русские и иероглифы). Попробуйте сохранить его, используя разные кодировки (в файлы “text_encoding_utf8.txt” и т. п.). Откройте созданные файлы в Блокноте.

   Замечание. Для проверки в программу добавьте комментарий, например, “удалось сохранить в кодировке …, корректно отображаются в Блокноте …”.

Работа с файловой системой

Напишите программы, которые выполняют следующие действия

4. Создайте новый каталог. Создайте в этом каталоге файл.
5. Переименуйте существующий файл. Проверьте существование файла.
6. Удалите существующий файл. Проверьте существование файла.
7. Покажите содержимое текущей папки (вывести на экран имена файлов).

Хранение целых чисел в двоичных файлах

В примере 1 (см. примеры выше) демонстрируется перевод целого числа int в тип bytes и обратно. В примере 2 создается битовый файл, содержащий целые числа. В примере 3 этот файл выводится на экран.

8. Создать файл, содержащий случайные числа из промежутка [100, 200]. Проверку.
9. Создать файл, содержащий случайное количество (от 10 до 15) случайных натуральных чисел.

10. Из файла целых чисел создать новый файл по правилу: все подряд идущие значения одного знака суммируются, группа нулей заменяется одним нулем.
11. Из двух файлов целых чисел, упорядоченных по возрастанию, создать новый файл чисел, упорядоченных по возрастанию.

---

12. Найти в файле максимальное и минимальное число и поменять их местами.
13. Найти в файле максимальное число и удалить его из файла.

    Замечание. Алгоритм удаления: в компоненту, содержащую максимальное число, скопировать последнюю компоненту файла, а затем удалить последнюю компоненту с помощью truncate.

14. Удалить из файла наименьшее нечетное число (см. замечание к предыдущей задаче).
15. Создать еще один файл, в который поместить числа из файла в обратном порядке.
16. Изменить в файле порядок следования элементов на обратный, меняя местами равноудаленные от концов элементы файла.
17. Проверить, есть ли в файле простые числа, вывести на экран сообщение.
18. Поменять местами в файле вторую и предпоследнюю компоненты.
19. Поменять местами компоненты файла по схеме:

1	**2**	**3**	**4**	**5**	**6**	→	**4**	**5**	**6**	**1**	**2**	**3**

(если файл содержит нечетное количество компонент, то предварительно удалить последнюю).

Лабораторная работа № 9
Пакет numpy, алгебра

Создание одномерных массивов (векторов) и двумерных массивов (матриц)

1. Создайте 2 вектора с помощью функции array.
2. Создайте 2 вектора с помощью функции arrange.
3. Создайте 2 вектора с помощью функции linspace.
4. Создайте 2 вектора с помощью функции random.
5. Создайте 2 вектора с помощью функции randint.

Атрибуты массивов, операции с векторами

6. Выдайте на экран информацию о созданном выше массиве: тип (dtype), размерность (ndim, shape), количество элементов (size), itemsize, nbytes.
7. С помощью срезов выделите несколько элементов в массиве и поменяйте их значения.
8. Найти результаты арифметических операций для 2 массивов (или числа и массива). Используйте операции или функции numpy (т.е., например, “+” или “np.add”). Вычислите еще любые две функции от массивов.

Расстояние, норма и скалярное произведение
(определения см. в лекции по алгебре)

9. Вычислите расстояние ρ_1 между двумя векторами.
10. Вычислите расстояние ρ_2 между двумя векторами.
11. Вычислите расстояние ρ_∞ между двумя векторами.
12. Вычислите норму $\|a\|_1$ вектора.
13. Вычислите норму $\|a\|_2$ вектора.
14. Вычислите норму $\|a\|_\infty$ вектора.
15. Вычислите скалярное произведение двух векторов.

Определение матрицы и операции с матрицами

16. Создайте 2 матрицы одинакового размера. Найти результаты арифметических операций для 2 матриц. Кроме поэлементного умножения “*”, вычислите еще произведение матриц, определяемое в алгебре, с помощью функции “np.dot”.
17. Найдите определитель квадратной матрицы.
18. Найдите собственные значения квадратной матрицы.
19. Найдите собственные значения и собственные вектора квадратной матрицы.
20. Найдите обратную матрицу к квадратной матрице.
21. Найдите норму квадратной матрицы.

Агрегирование

Определите произвольный массив. Найдите для него следующие величины (можно по одной из осей или для всего массива):

22. Сумму, среднее и минимальное значения.
23. Произведение, стандартное отклонение и индекс минимального значения.
24. Среднее значение, дисперсию и медиану.

Сравнения, маски и булева логика

Определите произвольный массив.

25. Найдите количество элементов из промежутка [–5, 5].
26. Определите, есть ли среди его элементов числа большие 100.
27. Определите, все ли они положительные.
28. Выбрать из массива все элементы из промежутка [0, 10].
29. Заменить в массиве все отрицательные элементы на 0.

Лабораторная работа № 10

Пакет pandas

1. Прочитайте текст из User Guide “10 minutes to pandas” (текст лекции) https://pandas.pydata.org/docs/user_guide/10min.html и выполните команды из него в Jupiter.

2. Выберите пункт из User Guide https://pandas.pydata.org/docs/user_guide/index.html (любой, например, Series, DataFrame или др.). Изучите его и выполните команды из этого пункта в Jupiter.

Лабораторная работа № 11

Пакет Matplotlib

1. Прочитайте текст из User's Guide "Pyplot tutorial" https://matplotlib.org/stable/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py и выполните команды из него в Jupiter.

2. Выберите пункт из User's Guide https://matplotlib.org/stable/users/index.html. Изучите его и выполните команды из него.