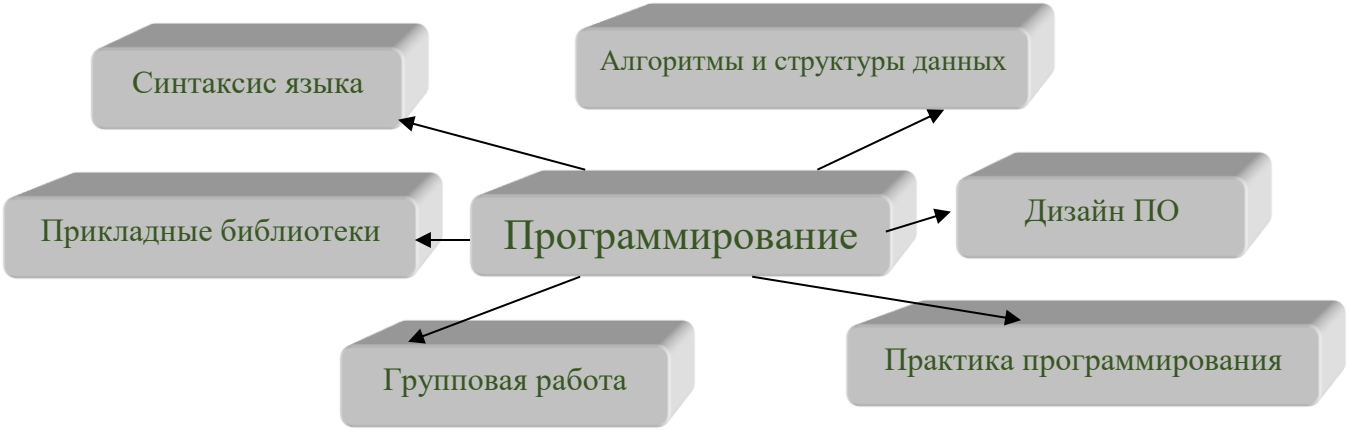
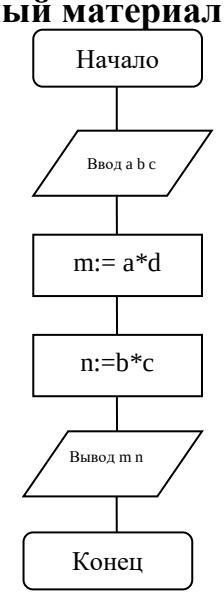
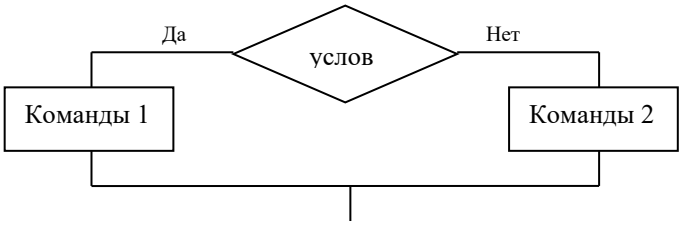
Раздаточный материл № 1



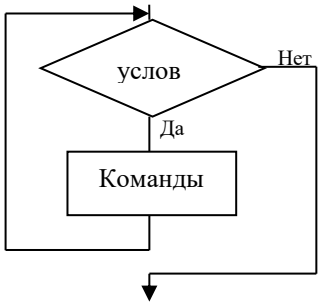
Раздаточный материал № 2



Раздаточный материал № 3



Раздаточный материал № 4



Раздаточный материал № 5

Java: int a = 1;
Python: a = 1

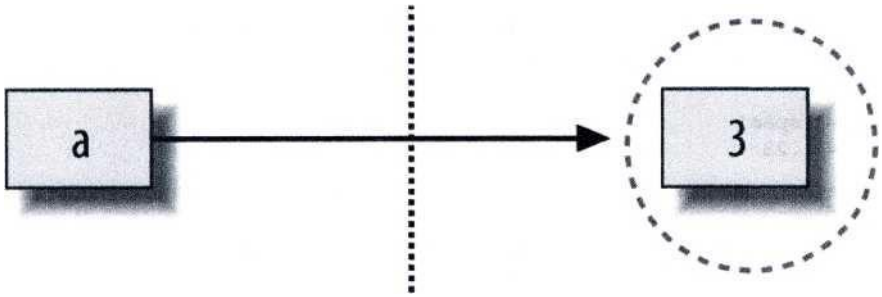
Раздаточный материл № 6

```
Import keyword
…
print(keyword.iskeyword("NumberInt"))
```

False

Раздаточный материл № 7

a = 3



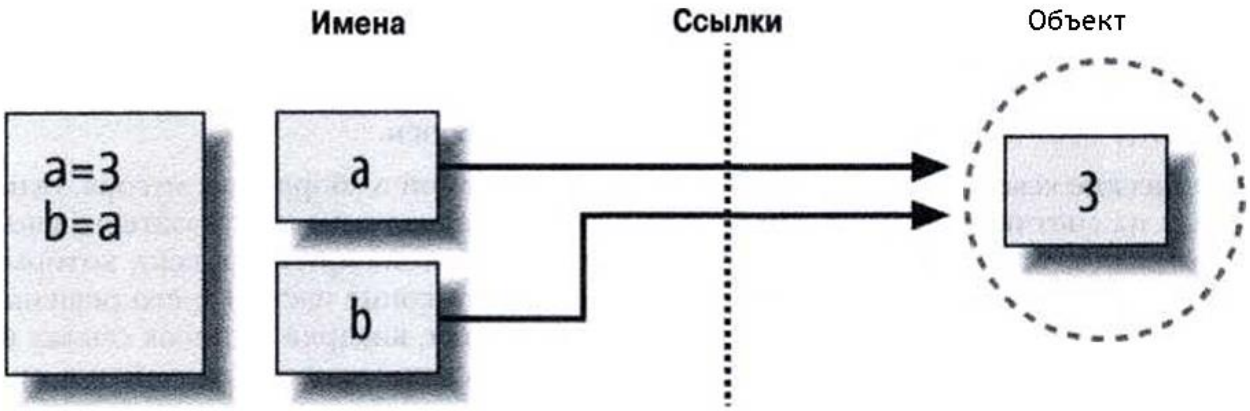
Раздаточный материл № 8

#1

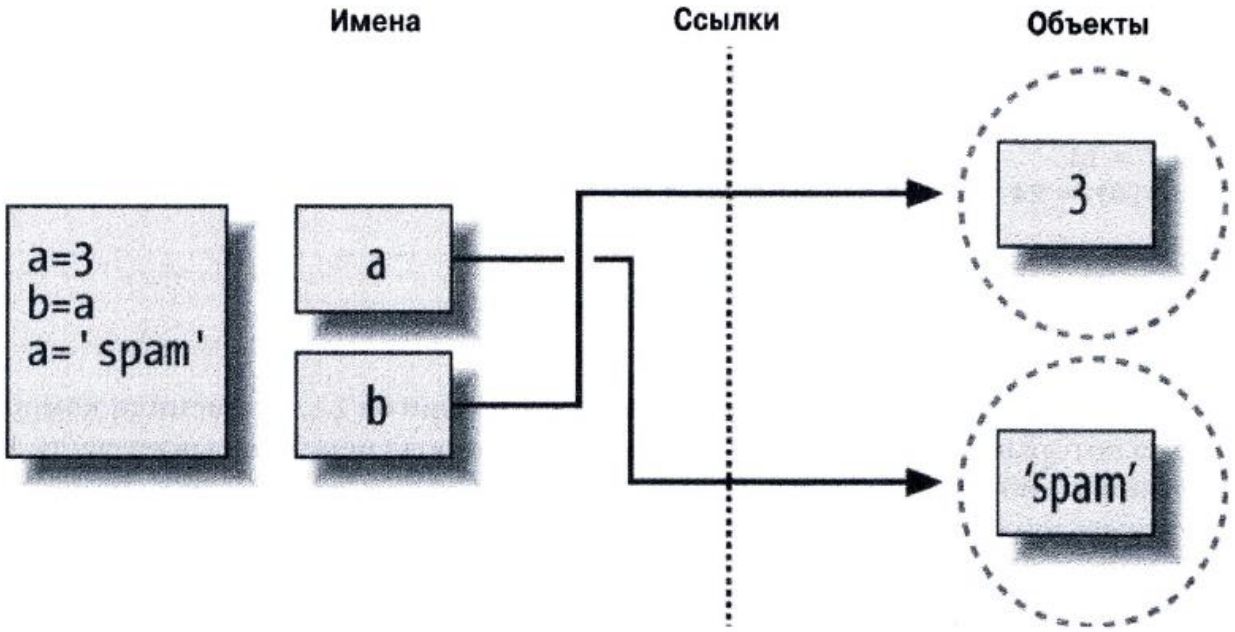
```
print(id(UserName))
```

11083840

Раздаточный материл № 9



Раздаточный материл № 10



Раздаточный материал № 11

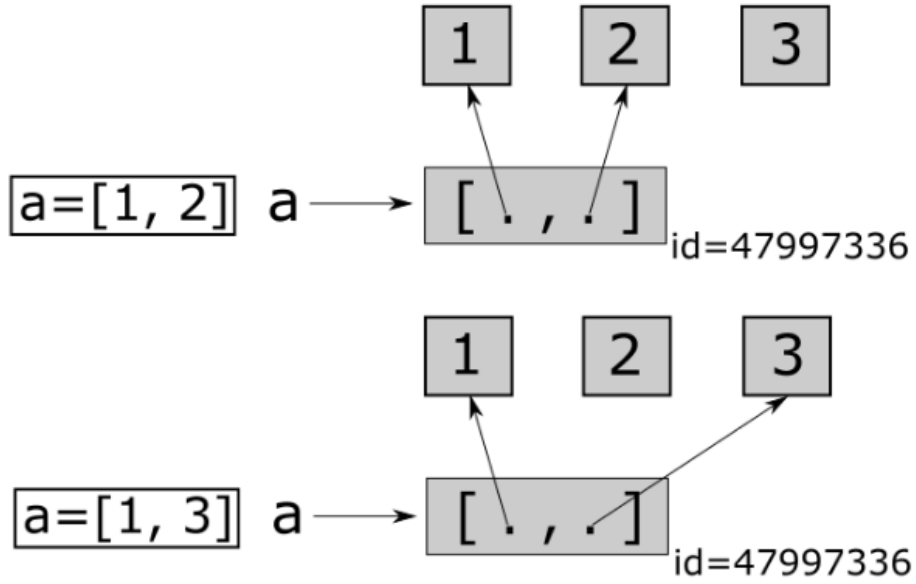
```
r = 15
print(id(r))→ 2012981392
r += 5.1
print(id(r))→ 1719872
```

Раздаточный материал № 12

создадим список [1, 2], а потом заменим второй элемент на 3.

```
>>> a = [1, 2]
>>> id(a)
47997336
>>>a[1] = 3
>>> a
[1, 3]
>>> id(a)
47997336
```

Объект, на который ссылается переменная a, был изменен. Это можно проиллюстрировать следующим рисунком.



Раздаточный материал № 13

Символ	Операция
==	Равно
!=	Не равно
<	Меньше
>	Больше
>=	Больше или равно
<=	Меньше или равно

Например,
x < y; a + b >= c/d; abs (m — n) <= 1.

Примеры вычисления значений отношений:

Отношение	Результат
12 >= 12	True
56 > 10	True
11 <= 6	False

Раздаточный материал № 14

and («и») - логическое умножение,
or («или») - логическое сложение,
not («не») - логическое отрицание.

Действие логических операций and, or и not определяется с помощью таблиц истинности. В этих таблицах показывается, как зависит значение логических выражений A and B, Aor B, not A от значений логических операндов A и B.

Раздаточный материал № 15

Таблица истинности операций and и or

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Таблица истинности операции not

A	not A
True	False
False	True

Логическое выражение	Описание
(x>=5) and (x<=10)	Истинно когда x принимает значения от 5 до 10
(x==3) or (x==5) or (x==7)	Истинно когда x принимает одно их трех значений: 3, 5 или 7
Not (x==5)	Истинно когда x != 5
(x==3) and (x==5)	Всегда ложно

Приоритет операций отношения ниже, чем приоритет логических операций, поэтому скобки являются обязательными.

Раздаточный материал № 16

1. Вычислить значения следующих логических выражений:

а) K % 7 == K // 5 - 1 при K = 15;

б) tAnd (P % 3 == 0) при t = True, P = 10101;

в) (x * y != 0) And (y > x) при x = 2, y = 1;

г) a Or Not b при a = False, b = True.

2. Написать оператор присваивания, в результате выполнения которого логическая переменная t получит значение True, если следующее утверждение истинно, и значение False — в противном случае:

а) из чисел x, y, z только два равны между собой;

б) x— положительное число;

в) каждое из чисел x, y, z положительное;

г) только одно из чисел x, y, z положительное;

д) p делится без остатка на q;

е) цифра 5 входит в десятичную запись трехзначного целого числа k.

Раздаточный материал № 17

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
   - int – целое число
   - float – число с плавающей точкой
   - complex – комплексное число
4. Списки (Sequence Type)
   - list – список
   - tuple – кортеж
   - range – диапазон
5. Строки (Text Sequence Type)
   - str
6. Бинарные списки (Binary Sequence Types)
   - bytes – байты
   - bytearray – массивы байт
   - memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (Set Types)
   - set – множество
   - frozenset – неизменяемое множество

8. Словари (Mapping Types)
    dict – словарь
9. Типы программных единиц
    функции
    модули
    классы
10. Типы, связанные с реализацией
    скомпилированный код
    трассировка стека

Раздаточный материал № 18

- целые числа (тип int) – положительные и отрицательные целые числа, а также 0(например, 4, 687, 45, 0).

– числа с плавающей точкой (тип float) – дробные, они же вещественные, числа (например, 1.45, 3.789654, 0.00453). Примечание: для разделения целой и дробной частей здесь используется точка, а не запятая.

Раздаточный материал № 19(справочно)

Встроенные математические функции Python доступные без подключения модулей.

abs(x) - возвращает модуль числа. Аргумент x может быть целым (int) или вещественным (float) числом.

pow(base, exp[, mod]) - возвращает base в степени exp. Допустима отрицательная и вещественная степень. Если указан третий аргумент mod, функция вернёт остаток по модулю.

divmod(a, b) - для целых аргументов возвращается кортеж с целочисленным результатом деления и остатком от деления.

round(number[, ndigits]) - возвращает число округлённое с точностью ndigits знаков после запятой. Если ndigits пропущено или равно None, функция возвращает ближайшее к number целое число.

oct(x) - конвертирует целое число в строку с восьмеричным числом с префиксом "0o".

bin(x) - конвертирует целое число в строку с двоичным числом с префиксом "0b".

hex(x) - конвертирует целое число в строку с шестнадцатеричным числом с префиксом "0x".

Раздаточный материал № 20

```
# Пример 1
a,b = 5,7 # позиционное присваивание кортежей
>>>a,b
(5, 7)
>>>a,b = b,a # обмен значениями
>>>a,b
(7, 5)
```

```
# Пример 2
x = [10, 11, 22, 83]
>>> i = 0
>>>i,x[i]=2,6
>>>x
[10, 11, 6, 83]
```

Раздаточный материал № 21

```
UserName = input('Введите имя')
```

Раздаточный материал № 22

```
NumberInt = int(input('Введите первое число:   '))

NumberFloat = float(input('Введите второе число:   '))
```

Раздаточный материал № 23

```
print ( [object, …] [,      sep='  '] [, end='\n’][,  file=sys .stdout] [, flush=False] )
```

Раздаточный материал № 24

```
#1
>>> print("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun", sep="-")
Mon-Tue-Wed-Thu-Fri-Sat-Sun
>>>print(1, 2, 3, sep="//")
1//2//3

#2
>>>print(10, end='\n\n')
10

>>>

#3
# \n перенесет каждое слово на новую строку
print('лекция', 'по', 'функции', 'print()', sep='\n')

лекция
по
функции
print()

#4
print('лекция', 'по', 'функции', 'print()', sep=',')

лекция,по,функции,print()
```

#5
print('лекция', 'по', 'функции', 'print()', sep=',+')

лекция,+по,+функции,+print()

Раздаточный материал № 25

# C - стиль
pupil = "Ben"
old = 16
grade = 9.2
print("It ' s %s, %d. Level: %.1f" % (pupil, old, grade))
It's Ben, 16. Level: 9.2

Раздаточныйматериал № 26

метод format()

print("This is a {0}. It's {1}.".format("ball", "red"))
This is a ball. It's red.

print("This is a {0}. It's {1}.".format("cat", "white"))
This is a cat. It's white.

print("This is a {0}. It's {1} {2}.".format(1, "a", "number"))
This is a 1. It'sanumber.

Раздаточный материал № 27

print("Тебя зовут {0}. Твоя фамилия {1}. Ты студент!". format(input(),input(
)))
print(f"Тебязовут {input()}. Твоя фамилия {input()}. Ты студент!")

Раздаточный материал № 28

#1
>>>size1 = "length - {}, width - {}, height - {}"
>>>size1.format(3, 6, 2.3)
'length - 3, width - 6, height — 2.3'
#2
>>> size2 = "height - {2}, length - {0}, width - {1}"
>>>size2.format(3, 6, 2.3)
'height - 2.3, length - 3, width - 6'

#3
>>> n = 20
>>> m = 25
>>> prod = n * m
>>>print(f'Произведение {n} на {m} равно {prod}')

Произведение 20 на 25 равно 500

Раздаточныйматериал № 29

```
>>>info = "This is a {subj}. It's {prop}."
>>>info.format(subj="table", prop="small")
"This is a table. It's small."
```

Раздаточный материал № 30

```
#1
if n < 100:
        b = n + a #отступ является обязательным, т.к. формирует тело
        условного оператора
print(b) # оператор print не является телом условного оператора

#2
tovar1 = 50
tovar2 = 32
if tovar1 + tovar2 > 9 :
        print("99 рублей недостаточно")
else:
        print("Чек оплачен")

#3

a = 5 > 0 # подвыражение 5 > 0 выполнится первым, после чего его
            результат будет присвоен переменной a
if a:
        print(a)

if a > 0 and a < b:
        print(b - a)

if 0 < a < b:
        print(b - a)
```

Раздаточный материал № 31

```
old = int(input('Вашвозраст: '))
print('Рекомендовано:', end=' ')
if 3 <= old < 6:
        print('"Заяцвлабиринте"')
elif 6 <= old < 12:
        print('"Марсианин"')
elif 12 <= old< 16:
        print('"Загадочный остров"')
elif 16 <= old:
        print('"Поток сознания"')
```

Раздаточный материал № 32

aif условие elseb

Раздаточныйматериал № 33

```
a = 50
b = 100
c = 40
max = a if a >b else b
max = c if c >max else max
print(max)
```

Раздаточный материал № 34

```
# Дано целое число. Если оно является положительным,
# то прибавить к нему 20, в противном случае вычесть
из него 5/
# Результат не сохраняется

c = int(input('Введи число:      '))
print('Результат =  ', c + 20 if c >= 0 else c - 5)

# Дано целое число. Если оно является положительным,
# то прибавить к нему 20, в противном случае вычесть
из него 5/
# Результат сохраняется

c = int(input('Введи число:      '))
f = c + 20 if c >= 0 else c - 5
print('Результат =  ', f)
```

Раздаточный материал № 35

```
while проверка:
    операторы
    if проверка: break          # Выход из цикла с пропуском else, если есть
    if проверка: continue           # Переход на проверку в начале цикла
else:
    операторы                   # Выполняется, если не было break
```

Блок else цикла выполняется тогда и только тогда, когда происходит нормальный выход из цикла (т.е. без выполнения оператора break).

Раздаточный материал № 36

```
def fund () : pass # Позже поместить сюда реальный код
def func2(): pass
```

Нельзя оставить тело функции пустым, не получив синтаксической ошибки, поэтому используется pass.

while True: pass # Для прекращения работы нажмите <Ctrl+C>!

Раздаточный материал № 37

```
deffunc1(): ...
func1()       # При вызове ничего не делает
```

Раздаточный материал № 38

```
t = 10
whilet:
t -= 1
ift % 2 != 0: continue # пропуск нечетных чисел
print(t, end=' ')
```

Раздаточныйматериал № 39

```
while True:
    name = input('Enter name: ')
if name == 'stop': break# привводestop -
выходизцикла
age = input ('Enter age: ')
print('Hello', name, '=>', int(age) ** 2)
```

Раздаточный материал № 40

```
try:
    n = int(input("Введите целое число: "))
    print("Удачно")
except:
    print("Что-то пошло не так")
```

Раздаточный материал № 41

```
try:
    n = int(input("Введите целое число: "))
    print("Удачно")
except ValueError:
    print("Что-то пошло не так")
```

Раздаточный материал № 42

```
n = input("Введите целое число: ")
while type(n) != int:
    try:
        n = int(n)
    except ValueError:
      print("Неправильно ввели!")
      n = input("Введите целое число: ")
```

```
if  not(math.fmod(n, 2)) :
        print("Четное")
else:
        print("Нечетное")
```

Раздаточныйматериал № 43

1. Даны три целых числа. Найти количество положительных чисел в исходном наборе.

```
# Даны три целых числа. Найти количество положительных
чисел в исходном наборе.
a, b, c = input("Введите первое число: "),
input("Введите второе число: "), input("Введите третье
число: ")

while type(a) != int:  # обработка исключений
try:
        a = int(a)
except ValueError:
print("Неправильно ввели!")
        a = input("Введите первое число: ")

while type(b) != int:  # обработка исключений
try:
        b = int(b)
except ValueError:
print("Неправильно ввели!")
        b = input("Введите второе число: ")

while type(c) != int:  # обработка исключений
try:
        c = int(c)
except ValueError:
print("Неправильно ввели!")
        c = input("Введите третье число: ")
k = 0
if a >0: k += 1
if b >0: k += 1
if c >0: k += 1
print('Количество положительных чисел =  ', k)
```

2. Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.

```
# Даны три переменные вещественного типа: A, B, C.
# Если их значения упорядочены по возрастанию или убыванию,
то удвоить их;
# в противном случае заменить значение каждой переменной на
противоположное.
# Вывести новые значения переменных A, B, C.

a, b, c = input("Введите первое число: "), input("Введите
второе число: "), input("Введите третье число: ")

while type(a) != float:  # обработка исключений
try:
        a = float(a)
except ValueError:
print("Неправильно ввели!")
        a = input("Введите первое число: ")

while type(b) != float:  # обработка исключений
try:
        b = float(b)
except ValueError:
print("Неправильно ввели!")
        b = input("Введите второе число: ")

while type(c) != float:  # обработка исключений
try:
        c = float(c)
except ValueError:
print("Неправильно ввели!")
c = input("Введитетретьечисло: ")

if (a>b>c) or (a<b<c):
    a *= 2;b *= 2;c *= 2;
print(a,b,c)
else:
    a = -a; b = -b; c = -c;
print(a, b, c)
```

3. Дано целое число K. Вывести строку-описание оценки, соответствующей числу K (1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»). Если K не лежит в диапазоне 1-5, то вывести строку «ошибка».

```
# Дано целое число K. Вывести строку-описание оценки,
соответствующей числу K
# (1 — «плохо», 2 — «неудовлетворительно», 3 —
«удовлетворительно»,
```

```
# 4 — «хорошо», 5 — «отлично»). Если К не лежит в диапазоне
1-5, то вывести строку «ошибка».

k = input("Введите число: ")

while type(k) != int:  # обработка исключений
try:
        k = int(k)
except ValueError:
print("Неправильно ввели!")
        k = input("Введите число: ")

if k == 1: print('Плохо')
elif k == 2: print('Неудовлетворительно')
elif k == 3: print('Удовлетворительно')
elif k == 4: print('Хорошо')
elif k == 5: print('Отлично')
else: print('Нет такой оценки!')
```

Раздаточный материал № 44

- Ввести 2 числа. Если их произведение отрицательно, умножить его на 8, в противном случае увеличить его в 1.5 раза.
- Вести число. Если оно четное, разделить его на 4, если нечетное - умножить на 5.
- Ввести двухзначное число. Если сумма цифр числа четная, то увеличить число на 2, в противном случае уменьшить на 2.
- Дано целое число. Если оно является положительным, то прибавить к нему 20, в противном случае вычесть из него 5.
- Дано два числа. Если их сумма кратна 5, то прибавить 1, иначе вычесть 2.

Раздаточный материал № 45

Даны два целых числа A и B (A < B). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество этих чисел (использовать оператор цикла)

```
# Даны два целых числа A и B (A < B). Вывести в порядке
возрастания все целые числа,
# расположенные между A и B (включая сами числа A и B), а
также количество этих чисел
# (использовать оператор цикла)

a, b = input("Введите первое число: "), input("Введите
второе число: ")

while type(a) != int:  # обработка исключений
try:
        a = int(a)
except ValueError:
print("Неправильно ввели!")
        a = input("Введите первое число: ")
```

```
while type(b) != int:  # обработка исключений
try:
        b = int(b)
except ValueError:
print("Неправильно ввели!")
        b = input("Введите второе число: ")
k = 0
while a <= b:
print(a)
    a += 1
k += 1
print('Количество чисел:  ', k)
```

Получить и вывести следующую арифметическую прогрессию: a_1=1, a_2=4, a_3=7, a_4=10, a_5=13, …

```
# Получить и вывести следующую арифметическую прогрессию:
# a1=1, a2=4, a3=7, a4=10, a5=13, …

k = input("Введите количество чисел арифметической
прогрессии: ")

while type(k) != int:  # обработка исключений
try:
        k = int(k)
except ValueError:
print("Неправильно ввели!")
k = input("Введитечисло: ")

l = 1; s = 1
while l <= k:
print (s)
    l += 1
s += 3
```

Найти факториал произвольного целого числа.

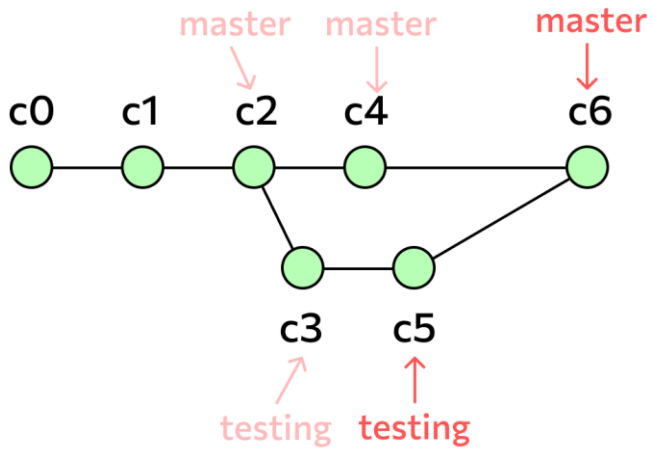
```
# Найти факториал произвольного целого числа.
k = input("Введите число для расчета факториала: ")

while type(k) != int:  # обработка исключений
try:
        k = int(k)
except ValueError:
print("Неправильно ввели!")
k = input("Введитечисло: ")

s = 1
while k:
```

```
    s *= k
    k -= 1
print (s)
```

Раздаточный материал № 46

1. Ввести 4 числа. Найти и вывести на экран сумму и количество отрицательных чисел.
2. Ввести 4 числа. Найти и вывести на экран количество четных чисел.
3. Найти и вывести на экран квадраты и кубы чисел от 2 до 5.
4. Найти и вывести на экран S=1!+2!+3!+4!+…+n! (n>1).
5. Ввести N чисел. Найти и вывести их среднее арифметическое.
6. Ввести N чисел. Посчитать и вывести количество чисел равных нулю.
7. Даны два целых числа A и B (A < B). Вывести в порядке убывания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество этих чисел (использовать оператор цикла).
8. Даны два целых числа A и B (A < B). Найти сумму всех целых чисел от A до B включительно (использовать оператор цикла).
9. Посчитать и вывести количество элементов арифметической прогрессии, удовлетворяющих условию $10<a_i<30$.
10. Вывести первые N (N≥3) чисел Фибоначчи и посчитать количество четных чисел.
11. Дана арифметическая прогрессия $a_1=1$, $a_2=4$, $a_3=7$, $a_4=10$, $a_5=13$, … Составить программу, которая каждый элемент прогрессии разделит на 2 и результат округлит до ближайшего целого.

Раздаточный материал № 47



Раздаточныйматериал № 48

```
def countFish():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Раздаточныйматериал № 49

```
    # программавыводитSOS

    defCharS():
    print('S', end='')


    defCharO():
    print('O', end='')


    CharS()
    CharO()
    CharS()
```

Раздаточныйматериал № 50

```
def rectangle():
    a = float(input("Ширина %s: " % figure)) #
обращениекглобальной
b = float(input("Высота %s: " % figure)) # переменной
figure
print("Площадь: %.2f" % (a*b))
def triangle():
    a = float(input("Основание %s: " % figure))
    h = float(input("Высота %s: " % figure))
print("Площадь: %.2f" % (0.5 * a * h))
figure = input("1-прямоугольник, 2-треугольник: ")
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()
```

Раздаточный материал № 51

```
# В основной ветке программы вызывается функция
cylinder(), которая вычисляет площадь
# цилиндра. В теле cylinder() определена функция
circle(), вычисляющая площадь круга по
# формуле пr2. В теле cylinder() у пользователя
спрашивается, хочет ли он получить только
# площадь боковой поверхности цилиндра, которая
вычисляется по формуле 2пrh, или полную
# площадь цилиндра. В последнем случае к площади
боковой поверхности цилиндра должен
# добавляться удвоенный результат вычислений
функции circle().

SC = 0
SQ = 0
```

```
def cylinder():
    r = float(input('Введи радиус:  '))

def circle():
        SC = 3.14 * r * 2
return SC

    c = input('1 - площадь боковой поверхности
цилиндра, 2 - полная площадь цилиндра:  ')
if c == '1':
print(circle())
elif c == '2':
        h = float(input('Введи высоту:  '))
        SQ = 2 * 3.14 * r * h + 2 * circle()
print(SQ)


cylinder()
```

Раздаточный материал № 52

Port = `cylinder()`

Раздаточный материал № 53

```
def duble():
    width = float(input('Введи ширину:   '))
    height = float(input('Введи высоту:   '))
    ploch = width * height
    perim  = 2 * (width + height)
return ploch, perim

g_ploch, g_perim = duble()
print('Полощадь прямоугольника:  ', g_ploch)
print('Периметр прямоугольника:  ', g_perim)
```

Раздаточный материал № 54

```
print(duble())

Введи ширину:   10
Введи высоту:   20
(200.0, 60.0) – скобки говорят, что выводится
```
кортеж

Раздаточный материал № 55

```
def duble(a, b):
    ploch = a * b
    perim  = 2 * (a + b)
```

```
    return ploch, perim

    width = float(input('Введи ширину:    '))
    height = float(input('Введи высоту:    '))
    g_ploch, g_perim = duble(width, height)
    print('Площадь прямоугольника:  ', g_ploch)
    print('Периметр прямоугольника:  ', g_perim)
```

Раздаточный материал № 56

```
# 1
defduble(a, b=20):
ploch = a * b
perim  = 2 * (a + b)
returnploch, perim

width = float(input('Введи ширину:    '))
g_ploch, g_perim = duble(width)
print('Площадь прямоугольника:  ', g_ploch)
print('Периметр прямоугольника:  ', g_perim)

# 2
defduble(a, b=20):
ploch = a * b
perim  = 2 * (a + b)
returnploch, perim

width = float(input('Введи ширину:    '))
height = float(input('Введи высоту:    '))
g_ploch, g_perim = duble(width, height)
print('Площадь прямоугольника:  ', g_ploch)
print('Периметр прямоугольника:  ', g_perim)
```

Раздаточный материал № 57

```
defduble(a, b):
    ploch = a * b
    perim = 2 * (a + b)
return ploch, perim


g_ploch, g_perim = duble(b=20, a=10)
print('Площадь прямоугольника:  ', g_ploch)
print('Периметр прямоугольника:  ', g_perim)
```

Раздаточныйматериал № 58

```
defoneOrMany(*a):
print(a)

oneOrMany(1)
```

```
oneOrMany('1', 1, 2, 'abc')
oneOrMany()
```

Результат:
(1,)
('1', 1, 2, 'abc')
()

Раздаточный материал № 59

```
# 1
defmax2(max,min):
ifmax>min:
returnmax
returnmin

defmax3(a,b,c):
returnmax2(a, max2(b,c))

a,b,c = int(input('Введи первое число:    ')),
int(input('Введи второе число:    ')),
int(input('Введи третье число:    '))

print('Максимальное число:    ',max3(a,b,c))
```

#2

```
# Составить функцию определения количества цифр в
целом числе

defcountInt(k):
t = 0
whilek>0:
k //= 10
t += 1
returnt


Int_Nuber = input("Введи целое число: ")

whiletype(Int_Nuber) != int:  # обработка исключений
try:
Int_Nuber = int(Int_Nuber)
exceptValueError:
print("Неправильно ввели!")
Int_Nuber = input("Введите целое число: ")
```

```
print('Количество цифр в цисле:    ',
countInt(Int_Nuber))

#3

# Найти все двузначные числа, в которых есть
заданная цифра. Реализовать с применением функции.

def search_figure(k):
    d = 10
while d != 100:
        a, b = divmod(d, 10)
if (a == k) or (b == k):
print(d, -d)
        d += 1


figure = input("Введи целое число от 0 до +- 9: ")

while type(figure) != int:  # обработка исключений
try:
        figure = int(figure)
except ValueError:
print("Неправильно ввели!")
        figure = input("Введите целое число: ")

search_figure(figure)
```

Раздаточный материал № 60(справочно)

abs() - возвращает абсолютное значение числа. Если это комплексное число, то абсолютным значением будет величина целой и мнимой частей.

chr() - возвращает строку, представляющую символ Unicode для переданного числа. Она является противоположностью ord(), которая принимает символ и возвращает его числовой код.

callable() - сообщает, является ли объект вызываемым. Если да, то возвращает True, а в противном случае — False. Вызываемый объект — это объект, который можно вызвать.

```
>>>callable(5)
False
```

round() - округляет вещественное число до определенного знака после запятой. Если второй аргумент не задан, то округление идет до целого числа. Второй аргумент может быть отрицательным числом. В этом случае округляться начинают единицы, десятки, сотни и т. д., то есть целая часть.

```
>>> a = 10/3
>>>a
3.3333333333333335
```

```
>>>round(a,2)
3.33
>>>round(a)
3

>>>round(5321, -1)
5320
>>>round(5321, -3)
5000
>>>round(5321, -4)
10000
```

divmod() - выполняет одновременно деление нацело и нахождение остатка от деления. Возвращает кортеж.

```
>>>divmod(10, 3)
(3, 1)
>>>divmod(20, 7)
(2, 6)
```

pow() - возводит в степень. Первое число – основание, второе – показатель. Может принимать третий необязательный аргумент - это число, на которое делится по модулю результат возведения в степень.

```
>>>pow(3, 2)
9
>>>pow(2, 4)
16

>>>pow(2, 4, 4)
0
>>> 2**4 % 4
0
```

dict() - используется для создания словарей. Это же можно делать и вручную, но функция предоставляет большую гибкость и дополнительные возможности. Например, ей в качестве параметра можно передать несколько словарей, объединив их в один большой.

```
>>>dict({"a":1, "b":2}, c = 3)
{'a': 1, 'b': 2, 'c': 3}

>>>list = [["a",1],["b",2]]
>>>dict(list)
{'a': 1, 'b': 2}
```

dir() - получает список всех атрибутов и методов объекта. Если объект не передать, то функция вернет все имена модулей в локальном пространстве имен.

```
>>>x = ["Яблоко", "Апельсин", "Гранат"]
```

```
>>>print(dir(x))
['__add__', '__class__', '__contains__',....]
```

enumerate() - в качестве параметра эта функция принимает последовательность. После этого она перебирает каждый элемент и возвращает его вместе со счетчиком в виде перечисляемого объекта. Основная особенность таких объектов — возможность размещать их в цикле для перебора.

```
>>> x = "Строка"
>>>list(enumerate(x))
[(0, 'С'), (1, 'т'), (2, 'р'), (3, 'о'), (4, 'к'), (5, 'а')]
```

eval() - обрабатывает переданное в нее выражение и исполняет его как выражение Python. После этого возвращается значение. Чаще всего эта функция используется для выполнения математических функций.

```
>>>eval('2+2')
4
>>>eval('2*7')
14
>>>eval('5/2')
2.5
```

filter() - функция используется для перебора итерируемых объектов и последовательностей, таких как списки, кортежи и словари. Но перед ее использованием нужно также иметь подходящую функцию, которая бы проверяла каждый элемент на валидность. Если элемент подходит, он будет возвращаться в вывод.

```
list1 = [3, 5, 4, 8, 6, 33, 22, 18, 76, 1]
result = list(filter(lambdax: (x%2 != 0) , list1))
print(result)
```

float() - конвертирует число или строку в число с плавающей точкой и возвращает результат. Если из-за некорректного ввода конвертация не проходит, возвращаются ValueError или TypeError.

hash() - у большинства объектов в Python есть хэш-номер. Функция hash() возвращает значение хэша переданного объекта. Объекты с __hash__() — это те, у которых есть соответствующее значение.

```
>>>hash('Hello World')
-2864993036154377761
>>>hash(True)
1
```

help() - предоставляет простой способ получения доступа к документации Python без интернета для любой функции, ключевого слова или модуля.

```
>>>help(print)
```

Help on built-in function print in module builtins:

int() - функция возвращает целое число из объекта, переданного в параметра. Она может конвертировать числа с разным основанием (шестнадцатеричные, двоичные и так далее) в целые.

```
>>>int(5.6)
5

>>>int('0101', 2)
5
```

iter() - принимает объект и возвращает итерируемый объект. Сам по себе он бесполезен, но оказывается крайне эффективным при использовании в циклах for и while. Благодаря этому объект можно перебирать по одному свойству за раз.

```
>>>lis = ['a', 'b', 'c', 'd', 'e']
>>> x = iter(lis)
>>>next(x)
'a'
>>>next(x)
'b'
>>>next(x)
'c'
>>>next(x)
'd'
```

max() - функция используется для нахождения «максимального» значения в последовательности, итерируемом объекте и так далее. В параметрах можно менять способ вычисления максимального значения.

```
>>>max('a', 'A')
'a'

>>>x = [5, 7, 8, 2, 5]
>>>max(x)
8

>>>x = ["Яблоко", "Апельсин", "Автомобиль"]
>>>max(x, key = len)
'Яблоко'
```

min() - функция используется для нахождения «минимального» значения в последовательности, итерируемом объекте и так далее. В параметрах можно менять способ вычисления минимального значения.

```
>>>min('a','A')
'A'

>>>x = [5, 7, 8, 2, 5]
```

```
>>>min(x)
2

>>>x = ["Виноград", "Манго", "Фрукты", "Клубника"]
>>>min(x)
'Виноград'
```

len() - функция используется для вычисления длины последовательности или итерируемого объекта.

```
>>> x = (2, 3, 1, 6, 7)
>>>len(x)
5
>>>len("Строка")
6
```

list() - в качестве параметра функция list() принимает итерируемый объект и возвращает список. Она обеспечивает большие гибкость и скорость при создании списков по сравнению с обычным способом.

```
>>>list("Привет")
['П', 'р', 'и', 'в', 'е', 'т']

>>>list({1:"a", 2:"b", 3:"c"})
[1, 2, 3]
```

map() - используется для применения определенной функции к итерируемому объекту. Она возвращает результат в виде итерируемого объекта (списки, кортежи, множества). Можно передать и несколько объектов, но в таком случае нужно будет и соответствующее количество функций.

```
>>>def inc(x):
        x = x + 1
return x

>>>lis = [1,2,3,4,5]
>>>result = map(inc,lis)

>>>for x in result:
print(x)

2
3
4
5
6
```

next() - используется для итерируемых объектов. Умеет получать следующий (next) элемент в последовательности. Добравшись до конца, выводит значение по умолчанию.

```
>>>lis = ['a', 'b', 'c', 'd', 'e']
>>> x = iter(lis)
>>>next(x)
'a'
>>>next(x)
'b'
>>>next(x)
'c'
>>>next(x)
'd'
```

ord() - принимает один символ или строку длиной в один символ и возвращает соответствующее значение Unicode. Например, ord("a") вернет 97, а 97 — a.

```
>>>ord('a')
97

>>>ord('A')
65
```

reversed() - предоставляет простой и быстрый способ развернуть порядок элементов в последовательности. В качестве параметра она принимает валидную последовательность, например, список, а возвращает итерируемый объект.

```
>>> x = [3,4,5]
>>> b = reversed(x)
>>>list(b)
[5, 4, 3]
```

range() - используется для создания последовательности чисел с заданными значениями от и до, а также интервалом. Такая последовательность часто используется в циклах, особенно в цикле for.

```
>>>list(range(10,20,2))
[10, 12, 14, 16, 18]
```

reduce() - выполняет переданную в качестве аргумента функцию для каждого элемента последовательности. Она является частью functools, поэтому перед ее использованием соответствующий модуль нужно импортировать.

```
>>> list1 = [2, 5, 3, 1, 8]
>>>functools.reduce(operator.add,list1)
19
```

```
>>> list1 = [2, 5, 3, 1, 8]
>>>functools.reduce(operator.mul,list1)
240

>>> list1 = [2, 5, 3, 1, 8]
>>>functools.reduce(operator.truediv,list1)
0.016666666666666666
```

sorted() - используется для сортировки последовательностей значений разных типов. Например, может отсортировать список строк в алфавитном порядке или список числовых значений по возрастанию или убыванию.

```
>>>X = [4, 5, 7, 3, 1]
>>>sorted(X)
[1, 3, 4, 5, 7]
```

str() - используется для создания строковых представлений объектов, но не меняет сам объект, а возвращает новый. У нее есть встроенные механизмы кодировки и обработки ошибок, которые помогают при конвертации.

```
>>>str(5)
'5'

>>> X = [5,6,7]
>>>str(X)
'[5, 6, 7]'
```

set() - используется для создания наборов данных, которые передаются в качестве параметра. Обычно это последовательность, например, строка или список, которая затем преобразуется в множество уникальных значений.

```
>>>set()
set()

>>>set("Hello")
{'e', 'l', 'o', 'H'}

>>>set((1,2,3,4,5))
{1, 2, 3, 4, 5}
```

sum() - автоматически суммирует все элементы и возвращает сумму.

```
>>>x = [1, 2, 5, 3, 6, 7]
>>>sum(x)
24
```

tuple() - принимает один аргумент (итерируемый объект), которым может быть, например, список или словарь, последовательность или итератор и возвращает его в форме кортежа. Если не передать объект, то вернется пустой кортеж.

```
>>>tuple("Привет")
('П', 'р', 'и', 'в', 'е', 'т')

>>>tuple([1, 2, 3, 4, 5])
(1, 2, 3, 4, 5)
```

type() - применяется в двух сценариях. Если передать один параметр, то она вернет тип этого объекта. Если же передать три параметра, то можно создать объект type.

```
>>>type(5)
<class 'int'>

>>>type([5])
<class 'list'>
```

Раздаточный материал № 61

1. Даны три целых числа. Определить у какого числа больше сумма цифр. Вывод результата предусмотреть в основной программе. Расчет суммы цифр оформить в функции.
2. Рассчитать и вывести периметр и площадь прямоугольника. Расчеты оформить в функции.
3. Написать программу, подсчитывающую количество цифр числа, используя для этого функцию.

Раздаточный материал № 62

```
>>> import math
```

Раздаточныйматериал № 63

```
>>>dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite',
'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Раздаточный материал № 64

```
>>>math.pow(2, 2)
4.0
>>> math.pi
3.141592653589793
```

Раздаточный материал № 65

```
>>>help(math.gcd)
```

Раздаточный материал № 66

>>>frommathimportgcd, sqrt, hypot

Раздаточный материал № 67

>>> from math import *

Раздаточныйматериал № 68

```
>>>pi = 3.14
>>>from math import pi
>>>pi
3.141592653589793
```

Раздаточныйматериал № 69

```
>>>from math import pi as P
>>> P
3.141592653589793
>>> pi
3.14
```

Раздаточный материал № 70(справочно)

Модуль Math

math.ceil(X) – округление до ближайшего большего числа.

math.copysign(X, Y) - возвращает число, имеющее модуль такой же, как и у числа X, а знак - как у числа Y.

math.fabs(X) - модуль X.

math.factorial(X) - факториал числа X.

math.floor(X) - округление вниз.

math.fmod(X, Y) - остаток от деления X на Y.

math.frexp(X) - возвращает мантиссу и экспоненту числа.

math.ldexp(X, I) - X * 2i. Функция, обратная функции math.frexp().

math.fsum(последовательность) - сумма всех членов последовательности. Эквивалент встроенной функции sum(), но math.fsum() более точна для чисел с плавающей точкой.

math.isfinite(X) - является ли X числом.

math.isinf(X) - является ли X бесконечностью.

math.isnan(X) - является ли X NaN (Not a Number - не число).

math.modf(X) - возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X.

math.trunc(X) - усекает значение X до целого.

math.exp(X) - eX.

math.expm1(X) - eX - 1. При X → 0 точнее, чем math.exp(X)-1.

math.log(X, [base]) - логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.

math.log1p(X) - натуральный логарифм (1 + X). При X → 0 точнее, чем math.log(1+X).

math.log10(X) - логарифм X по основанию 10.

math.log2(X) - логарифм X по основанию 2. Новое в Python 3.3.

math.pow(X, Y) - X^Y.

math.sqrt(X) - квадратный корень из X.

math.acos(X) - арккосинус X. В радианах.

math.asin(X) - арксинус X. В радианах.

math.atan(X) - арктангенс X. В радианах.

math.atan2(Y, X) - арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y).

math.cos(X) - косинус X (X указывается в радианах).

math.sin(X) - синус X (X указывается в радианах).

math.tan(X) - тангенс X (X указывается в радианах).

math.hypot(X, Y) - вычисляет гипотенузу треугольника с катетами X и Y (math.sqrt(x * x + y * y)).

math.degrees(X) - конвертирует радианы в градусы.

math.radians(X) - конвертирует градусы в радианы.

math.cosh(X) - вычисляет гиперболический косинус.

math.sinh(X) - вычисляет гиперболический синус.

math.tanh(X) - вычисляет гиперболический тангенс.

math.acosh(X) - вычисляет обратный гиперболический косинус.

math.asinh(X) - вычисляет обратный гиперболический синус.

math.atanh(X) - вычисляет обратный гиперболический тангенс.

math.erf(X) - функция ошибок.

math.erfc(X) - дополнительная функция ошибок (1 - math.erf(X)).

math.gamma(X) - гамма-функция X.

math.lgamma(X) - натуральный логарифм гамма-функции X.

math.pi - pi = 3,1415926...

math.e - e = 2,718281...

Раздаточныйматериал № 71

```
>>>import random
>>>from random import random, randrange, randint
```

Раздаточный материал № 72

```
>>>random.randint(0, 10)
10
```

или (если импортировались отдельные функции):

```
>>>randint(-100, 200)
-10
```

Раздаточный материал № 73

randrange(10, 20, 3) → "случайное" число будет выбираться из чисел 10, 13, 16, 19

Раздаточный материал № 74

```
>>>random.random()
0.17855729241927576
или
>>>random()
0.025328854415995194
```

Раздаточный материал № 75

```
>>> round(random.random(), 3)
0.629
```

Раздаточный материал № 76

```
>>>random.random() * 10
2.510618091637596
>>>random.random() * (1 + 1) - 1
-0.673382618351051
```

Раздаточный материал № 77

```
>>> a = [12, 3.85, "black", -4]
>>> a
[12, 3.85, 'black', -4]
```

Раздаточный материал № 78

```
>>>a[0]
12
>>>a[3]
-4
```

Раздаточный материал № 79

```
>>>a[0:2]
[12, 3.85]
```

Раздаточный материал № 80

```
>>>a[:3]
[12, 3.85, 'black']
>>>a[2:]
['black', -4]
>>>a[:]
```

[12, 3.85, 'black', -4]

Раздаточный материал № 81

```
>>>a[1] = 4
>>> a
[12, 4, 'black', -4]
```

Раздаточный материал № 82

```
>>> a = [1, 3, 5, 7]
>>> b = a[:]
>>>print(a)
[1, 3, 5, 7]
>>>print(b)
[1, 3, 5, 7]
```

Раздаточный материал № 83(справочно)

list.append(значение) – добавление нового значения в конец списка.

list.insert(позиция, значение) - добавление нового значения в указанную позицию списка.

list.remove(значение) – удаляет указанное значение из списка, не привязываясь к индексу.

list.pop() – удаляет последний элемент списка и возвращает значение.

list.pop(индекс) – удаляет элемент списка с указанным индексом и возвращает значение.

dela[индекс] - удаляет элемент списка с указанным индексом.

dela[индекс : индекс] - удаляет срез элементов списка с указанными индексами.

list.clear() – удаляет все элементы из списка.

list.index - Возвращает индекс элемента

list.count(x) Возвращает количество вхождений элемента x в список

list.sort(key=None, reverse=False) - сортирует элементы в списке по возрастанию. Для сортировки в обратном порядке используйте флаг reverse=True.

ist.reverse() - изменяет порядок расположения элементов в списке на обратный.

list.copy() - возвращает копию списка.

Раздаточный материал № 84

```
for цель in объект:          # Присваивает цели элементы объекта
    операторы          # Повторяемое тело цикла: использует цель
else:                  # Необязательная часть else
    операторы          # Если не встречался оператор break
```

```
spisok = [10, 40, 20, 30]
>>>for element in spisok:
        print(element + 2)
12
42
22
32
```

Раздаточный материал № 85

```
for цель in объект:                    #Присваивает цели элементы объекта
    операторы if проверка: break       #Выход из цикла с пропуском else
    if проверка: continue              #Переход в начало цикла
else:
    операторы                          #Если не встречался оператор break
```

Раздаточный материал № 86(справочно)

```
#1    изменить срез:
>>>mylist = ['ab','ra','ka','da','bra']
>>>mylist[0:2] = [10,20]
>>>mylist
[10, 20, 'ka', 'da', 'bra']
```

#2 пример создания пустого списка с последующим заполнением его в цикле случайными числами:

```
>>>import random
>>> c = []
>>> i = 0
>>> while i < 10:
... c.append(random.randint(0,100))
... i += 1
...
>>> c
74
[30, 44, 35, 77, 53, 44, 49, 17, 61, 82]
```

```
#3 сортировка списка
>>>a = [1, 4, 2, 8, 1]
>>>a.sort()
>>>print(a)
[1, 1, 2, 4, 8]
```

```
#4 изменения порядка расположения на обратный
>>> a = [1, 3, 5, 7]
>>>a.reverse()
>>>print(a)
[7, 5, 3, 1]
```

#5

```
# Программа запрашивает с клавиатуры пять чисел,
добавляет их в список.
# На экран выводит их сумму, максимальное и
минимальное из них.

ListAppend = []
i = 0
while i <5:
    ListAppend.append(int(input('Введи значение
списка:  ')))
    i += 1
print(ListAppend)
print('Сумма элементов списка:   ', sum(ListAppend))
print('Минимальный элемент списка:   ',
min(ListAppend))
print('Максимальный элемент списка:   ',
max(ListAppend))
```

#6

```
# Программа генерирует сто случайных вещественных
чисел и заполняет ими список.
# Выводит получившийся список на экран по десять
элементов в ряд.
# Далее сортирует список с помощью метода sort() и
# снова выводит его на экран по десять элементов в
строке.
# Для вывода списка использована отдельная функция,
которая в качестве аргумента принимает список.
import random


def PrintList(d):
    i = 0
while i <100:
if (i % 10 != 0) or (i == 0):
print(d[i], end=' ')
else:
print()
print(d[i], end=' ')
        i += 1
print()


ListAppend = []
t = 0
while t <100:
    ListAppend.append(round(random.random(), 2))
    t += 1
```

```
print('Исходныйсписок', sep='\n')
PrintList(ListAppend)
ListAppend.sort()
print('Отсортированныйсписок', sep='\n')
PrintList(ListAppend)
```

#7

```
# Дан целочисленный массив размера N. Увеличить все
четные числа,
# содержащиеся в массиве, на исходное значение
последнего четного числа.
# Если четные числа в массиве отсутствуют, то
оставить массив без изменений.

import random


def PrintList(ListAppend):
    for element in ListAppend:
print(element, end=' ')
print()


d = int(input('Введиразмермассива:   '))
ListAppend = []
t = 0
while t <d:
ListAppend.append(random.randint(-100, 100))
if ListAppend[t] % 2 == 0:
k = ListAppend[t]
t += 1

print('Исходныймассив', sep='\n')
PrintList(ListAppend)
print('Последнеечетноечисло:  ', k, sep='\n')
for i in range(len(ListAppend)):
    if ListAppend[i] % 2 == 0:
ListAppend[i] += k

print('Полученныйсписок', sep='\n')
PrintList(ListAppend)
```

#8

```
# Даны два массива A и B размера 5, элементы которых
упорядочены по возрастанию.
# Объединить эти массивы так, чтобы результирующий
массив C (размера 10)
# остался упорядоченным по возрастанию.
```

```
import random
a,b,c = [],[],[]
i = 0
while i <5:
a.append(random.randrange(0,20))
b.append(random.randrange(0, 20))
i += 1

a.sort()
b.sort()
print('МассивА:  ',a)
print('МассивВ:  ',b)
c = a + b
c.sort()
print('Массив С:  ',c)
```

#9

```
# Данмассив А размера N. Сформироватьновыймассив В
тогожеразмерапоследующемуправилу:
# элемент
ВКравенсреднемуарифметическомуэлементовмассива А
сномерамиот К до N

import random

n = int(input('Введиразмермассива'))
a, b = [], []
t = 0
while t <n:
a.append(random.randint(1, 2))
t += 1
print('Исходныймассив', a, sep='\n')
t = 0
i = 0
while t <n:
s = 0
while i <n:
s += a[i]
i += 1
b.append(s / (n - t))
t += 1
i = t
print(b)
```

Раздаточный материал № 87

```
>>>s = "Hello, World!"
>>>s[0]
'H'
```

```
>>>s[7:]
'World!'
>>>s[::2]   # здесь извлечение идет с шагом = 2
'Hlool!'
```

Раздаточный материал № 88

```
>>>s = s[0:-1] + '.'
>>>s
'Hello, World.'    # старое значение s теряется
```

Раздаточный материал № 89

obj.foo(<args>)

Раздаточный материал № 90(справочно)

Строковые операторы	

+ - конкатенация строк	>>> s = 'py'
	>>> t = 'th'
	>>> u = 'on'
	>>> s + t + u
	'python'
	>>> print('Привет, ' + 'Мир!')

* - умножение строк. Значение множителя должно быть целым положительным числом	>>> s = 'py.'
	>>> s * 4
	'py.py.py.py.'

in - оператор принадлежности подстроки, возвращает True, если подстрока входит в строку, и False, если нет. Есть также оператор not in, у которого обратная логика	>>> s = 'Python' >>>s in 'I love Python.' True >>>s in 'I love Java.' False
Встроенные функции строк	

split() позволяет разбить строку по пробелам. В результате получается список слов.
Может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение
	>>> s = input()
	red blue orange white
	>>> s
	'red blue orange white'
	>>> sl = s.split()
	>>> sl
	['red', 'blue', 'orange', 'white']
	>>> s
	'red blue orange white'
	
	>>> s.split('e')
	['r', 'd blu', ' orang', ' whit', '']
	>>> '40030023'.split('00')
	['4', '3', '23']

Метод строк join() выполняет обратное действие. Он формирует из списка	>>> '-'.join(sl) 'red-blue-orange-white'

строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список. Если разделитель не нужен, то метод применяется к пустой строке	>>> ''.join(sl) 'redblueorangewhite'

find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.	>>> s
	'red blue orange white'
	>>> s.find('blue')
	4
	>>> s.find('green')
	-1
	
	>>> letters = 'ABCDACFDA'
	>>> letters.find('A', 3)
	4
	>>>letters.find('DA', 0, 6)
	3
	# Поиск идет с третьего индекса и до конца, а также с первого и до шестого

replace() заменяет одну подстроку на другую	>>>letters.replace('DA', 'NET')
	'ABCNETCFNET'
	
	Исходная строка не меняется:
	>>> letters
	'ABCDACFDA'
	
	если результат надо сохранить, то его надо присвоить переменной
	>>> new_letters = letters.replace('DA', 'NET')
	>>> new_letters
	'ABCNETCFNET'

ord(c) возвращает числовое значение для заданного символа	>>> ord('a')
	97
	>>> ord('#')
	35

chr(n) возвращает символьное значение для данного целого числа.	>>> chr(8364)
	'€'
	>>> chr(8721)
	'∑'

len(s) возвращает длину строки	>>> s = 'Простая строка.'
	>>> len(s)
	15

str(obj) возвращает строковое представление объекта	>>> str(49.2)
	'49.2'
	>>> str(3+4j)
	'(3+4j)'
	>>> str(3 + 29)

	'32' >>> str('py') 'py'
Встроенные методы строк	

string.capitalize() приводит первую букву в верхний регистр, остальные в нижний.	>>> s = 'everyTHing yoU Can IMaGine is rEAl'
	>>> s.capitalize()
	'Everything you can imagine is real'

string.lower() преобразует все буквенные символы в строчные.	>>> 'everyTHing yoU Can IMaGine is rEAl'.lower() 'everything you can imagine is real'

string.swapcase() меняет регистр буквенных символов на противоположный.	>>> 'the sun also rises'.title()
	'The Sun Also Rises'
	
	>>> 'follow us @PYTHON'.title()
	'Follow Us @Python'

string.upper() преобразует все буквенные символы в заглавные.	>>> 'follow us @PYTHON'.upper() 'FOLLOW US @PYTHON'

string.count(<sub>[, <start>[, <end>]]) подсчитывает количество вхождений подстроки в строку.
s.count(<sub>) возвращает количество точных вхождений подстроки <sub> в s:
Количество вхождений изменится, если указать <start> и <end>
	>>> 'foo goo moo'.count('oo')
	3
	>>> 'foo goo moo'.count('oo', 0, 8)
	2

string.endswith(<suffix>[, <start>[, <end>]]) определяет, заканчивается ли строка заданной подстрокой
s.endswith(<suffix>) возвращает, True если s заканчивается указанным <suffix> и False если нет
Сравнение ограничено подстрокой, между <start> и <end>, если они указаны
	>>> 'python'.endswith('on')
	True
	>>> 'python'.endswith('or')
	False
	>>> 'python'.endswith('yt', 0, 4)
	True
	>>> 'python'.endswith('yt', 2, 4)
	False

string.find(<sub>[, <start>[, <end>]]) ищет в строке заданную подстроку
s.find(<sub>) возвращает первый индекс в s который соответствует началу строки <sub>
Этот метод возвращает, -1 если указанная подстрока не найдена
Поиск в строке ограничивается подстрокой, между <start> и <end>, если они указаны
	>>> 'Follow Us @Python'.find('Us')
	7
	>>> 'Follow Us @Python'.find('you')
	-1
	>>> 'Follow Us @Python'.find('Us', 4)
	7
	>>> 'Follow Us @Python'.find('Us', 4, 7)
	-1

s.rfind(<sub>) возвращает индекс последнего вхождения подстроки <sub> в s, который соответствует	>>> 'Follow Us @Python'.rfind('o') 15

началу <sub>. Как и в .find(), если подстрока не найдена, возвращается -1. Поиск в строке ограничивается подстрокой, между <start> и <end>, если они указаны.	

string.isalnum() определяет, состоит ли строка из букв и цифр, возвращает True, если строка s не пустая, а все ее символы буквенно-цифровые (либо буква, либо цифра). В другом случае False	>>> 'abc123'.isalnum()
	True
	>>> 'abc$123'.isalnum()
	False
	>>> ''.isalnum()
	False

string.isalpha() определяет, состоит ли строка только из букв, возвращает True, если строка s не пустая, а все ее символы буквенные. В другом случае False	>>> 'ABCabc'.isalpha()
	True
	>>> 'abc123'.isalpha()
	False

string.isdigit() определяет, состоит ли строка из цифр (проверка на число), возвращает True когда строка s не пустая и все ее символы являются цифрами, а в False если нет	>>> '123'.isdigit()
	True
	>>> '123abc'.isdigit()
	False

string.isidentifier() определяет, является ли строка допустимым идентификатором Python, возвращает True, если s валидный идентификатор (название переменной, функции, класса и т.д.) python, а в False если нет. Вернет True для строки, которая соответствует зарезервированному ключевому слову python, даже если его нельзя использовать	>>> 'foo32'.isidentifier()
	True
	>>> '32foo'.isidentifier()
	False
	>>> 'foo$32'.isidentifier()
	False

string.islower() определяет, являются ли буквенные символы строки строчными, возвращает True, если строка s не пустая, и все содержащиеся в нем буквенные символы строчные, а False если нет. Не алфавитные символы игнорируются	>>> 'abc'.islower()
	True
	>>> 'abc1$d'.islower()
	True
	>>> 'Abc1$D'.islower()
	False

string.isprintable() определяет, состоит ли строка только из печатаемых символов, возвращает, True если строка s пустая или все буквенные символы которые она содержит можно вывести на экран. Возвращает, False если s содержит хотя бы один специальный символ. Не алфавитные символы игнорируются. Это единственный метод, который возвращает True, если s	>>> 'a\tb'.isprintable() # \t - символтабуляции
	False
	>>> 'a b'.isprintable()
	True
	>>> ''.isprintable()
	True
	>>> 'a\nb'.isprintable() # \n - символ перевода строки
	False

пустая строка. Все остальные возвращаются False	

string.isspace() определяет, состоит ли строка только из пробельных символов, возвращает True, если s не пустая строка, и все символы являются пробельными, а False, если нет. Наиболее часто встречающиеся пробельные символы — это пробел ' ', табуляция '\t' и новая строка '\n'	>>> ' \t \n '.isspace()
	True
	>>> ' a '.isspace()
	False

string.istitle() определяет, начинаются ли слова строки с заглавной буквы, возвращает True когда s не пустая строка и первый алфавитный символ каждого слова в верхнем регистре, а все остальные буквенные символы в каждом слове строчные. Возвращает False, еслинет	>>> 'This Is A Title'.istitle()
	True
	>>> 'This is a title'.istitle()
	False
	>>> 'Give Me The #$#@ Ball!'.istitle()
	True

string.isupper() определяет, являются ли буквенные символы строки заглавными, возвращает True, если строка s не пустая, и все содержащиеся в ней буквенные символы являются заглавными, и в False, если нет. Не алфавитные символы игнорируются	>>> 'ABC'.isupper()
	True
	>>> 'ABC1$D'.isupper()
	True
	>>> 'Abc1$D'.isupper()
	False

Раздаточный материал № 91

```
# Дан список размера N, состоящий из символов. Привести символы букв
# из верхнего регистра в нижний. Определить количество букв.

import random
import List_def as l1

List_4 = []
d = int(input('Введи размер массива:    '))

t = 0
while t <d:
List_4.append(chr(random.randint(45, 100)))
t += 1

print('Исходный массив:    ')
l1.PrintList(List_4)

h = 0
for i in List_4:
    if i.isalpha():
```

```
h += 1

print('Количество букв:    ', h)


New_String = ''.join(List_4) # преобразование списка в
строку
print("Переводим буквы в нижний регистр:    ",
New_String.lower())
```

Раздаточныйматериал № 92
storm_1 = ('Lightning')
Union = (' and ')
storm_2 = ('Thunder')
print(storm_1 + Union + storm_2)
Результат: Lightning and Thunder

dog_do = ('woof!',)
print(dog_do * 3)
Результат: ('woof!', 'woof!', 'woof!')

Раздаточный материал № 93

>>>a[3]
89
>>>a[1:3]
(2.13, 'square')

Раздаточныйматериал № 94
>>> a = ()
>>>print(type(a))
<class 'tuple'>

Раздаточныйматериал № 95
>>> a = (1, 2, 3, 4, 5)
>>>print(type(a))
<class 'tuple'>
>>>print(a)
(1, 2, 3, 4, 5)
>>>print(*a)
1 2 3 4 5
>>> a = tuple('hello, world!')
>>>a
('h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd', '!')

Раздаточный материал № 96

```
>>> a = tuple((1, 2, 3, 4))
>>> print(a)
(1, 2, 3, 4)
```

Раздаточный материал № 97

```
>>>a = (1, 2, 3, 4, 5)
>>>print(a[0])
1
>>>print(a[1:3])
(2, 3)
```

Раздаточныйматериал № 98

```
>>>del a
>>>print(a)
Traceback (most recent call last):
File "<pyshell#28>", line 1, in <module>
print(a)
NameError: name 'a' is not defined
```

Раздаточный материал № 99

```
>>>lst = [1, 2, 3, 4, 5]
>>>print(type(lst))
<class 'list'>
>>>print(lst)
[1, 2, 3, 4, 5]
>>>tpl = tuple(lst)
>>>print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(1, 2, 3, 4, 5)
```

Обратная операция также является корректной:

```
>>>tpl = (2, 4, 6, 8, 10)
>>>print(type(tpl))
<class 'tuple'>
>>>print(tpl)
(2, 4, 6, 8, 10)
>>>lst = list(tpl)
>>>print(type(lst))
<class 'list'>
>>>print(lst)
[2, 4, 6, 8, 10]
```

Раздаточныйматериал № 100

```
>>> nested = (1, "do", ["param", 10, 20])
```

```
>>>nested[2][1] = 15
>>> nested
(1, 'do', ['param', 15, 20])
```
Выражения типа nested[2][1] используются для обращения к вложенным объектам. Первый индекс указывает на позицию вложенного объекта, второй – индекс элемента внутри вложенного объекта.

Раздаточный материал № 101

```
>>> T = (1, 2, 3, 2, 4, 2)          # Методы кортежей в Python 2.6, 3.0
                                    # и последующих версиях
>>>T.index(2)                       # Смещение первого появления элемента 2
1
>>>T.index(2, 2)                    # Смещение появления элемента 2 после смещения 2
3
>>>T.count(2)                       # Сколько всего элементов 2?
3
```

Раздаточный материал № 102

```
tuplex = (4, 6, 2, 8, 3, 1)
a, b, *c = tuplex
Результат: 4 6 [2, 8, 3, 1, 9]
Переменные a и b содержат целочисленные переменные, в c
помещается список. Исходный кортеж tuplex остается
неизменным.
```

Раздаточный материал № 103

```
>>>d1 = dict()
>>>print(type(d1))
<class 'dict'>

>>> d2 = {}
>>>print(type(d2))
<class 'dict'>
```

Раздаточныйматериал № 104

```
>>>d1 = dict(Ivan="менеджер", Mark="инженер")
>>>print(d1)
{'Mark': ‘инженер’, 'Ivan': ‘менеджер’}

>>> d2 = {"A1":"123", "A2":"456"}
>>>print(d2)
{'A2': '456', 'A1': '123'}

>>> a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}
```

Созданиесловарячерезвложенныйсписок

```
a = [['cat','кошка'], ['dog','собака'], ['bird','птица'],
['mouse','мышь']]
```

```
s = dict(a)
print(s)
```
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}

Раздаточный материал № 105

```
>>>a['cat']
'кошка'

>>>a['bird']
'птица'
```

Раздаточный материал № 106

```
>>> a['elephant'] = 'бегемот'    # добавляем
>>> a['table'] = 'стол'          # добавляем
>>> a
```
{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь', 'bird': 'птица', 'table': 'стол', 'elephant': 'бегемот'}

```
>>>a['elephant'] = 'слон'              # изменяем
>>>del a['table']                     # удаляем
>>>a
```
{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь', 'bird': 'птица', 'elephant':'слон'}

Раздаточный материал № 107

```
d2 = {"A1":"123", "A2":"456"}
"A1" in d2
True
"A3" in d2
False
```

Раздаточныйматериал № 108

```
nums = {1: 'one', 2: 'two', 3: 'three'}
person = {'name': 'Tom', 1: [30, 15, 16], 2: 2.34, ('ab', 100): 'no'}
```

Раздаточный материал № 109

```
# извлекаются ключи
foriinnums:
    print(i)
```

Результат

1
2
3

```
# извлекаются значения:
for i in nums:
        print(nums[i])
```

Результат
one
two
three

Раздаточный материал № 110 (справочно)

clear() -удаляет все элементы словаря, но не удаляет сам словарь	>>> d2 = {"A1":"123", "A2":"456"}
	>>> print(d2)
	{'A2': '456', 'A1': '123'}
	>>> d2.clear()
	>>> print(d2)
	{}

copy() - создает новую копию словаря	>>> d2 = {"A1":"123", "A2":"456"}
	>>> d3 = d2.copy()
	>>> print(d3)
	{'A1': '123', 'A2': '456'}
	>>> d3["A1"]="789"
	>>> print(d2)
	{'A2': '456', 'A1': '123'}
	>>> print(d3)
	{'A1': '789', 'A2': '456'}

fromkeys(seq[, value]) - создает новый словарь с ключами из seq и значениями из value. По умолчанию value присваивается значение None.	`t = dict.fromkeys(['a','b','c'],15)`
	`print(t)`
	{'a': 15, 'b': 15, 'c': 15}

get(key) - возвращает значение из словаря по ключу key	>>> d = {"A1":"123", "A2":"456"}
	>>> d.get("A1")
	'123'

items() - возвращает (в виде кортежа) элементы словаря (ключ, значение) в отформатированном виде	>>> d = {"A1":"123", "A2":"456"}
	>>> d.items()
	dict_items([('A2', '456'), ('A1', '123')])

keys() - возвращает ключи словаря	>>> d = {"A1":"123", "A2":"456"} >>> d.keys()

	dict_keys(['A2', 'A1'])

pop(key[, default]) - если ключ key есть в словаре, то данный элемент удаляется из словаря и возвращается значение по этому ключу, иначе будет возвращено значение default. Если default не указан и запрашиваемый ключ отсутствует в словаре, то будет вызвано исключение KeyError	>>> d = {"A1":"123", "A2":"456"}
	>>> d.pop("A1")
	'123'
	>>> print(d)
	{'A2': '456'}

popitem() - удаляет и возвращает последнюю пару (ключ, значение) из словаря. Если словарь пуст, то будет вызвано исключение KeyError	>>> d = {"A1":"123", "A2":"456"}
	>>> d.popitem()
	('A2', '456')
	>>> print(d)
	{'A1': '123'}

setdefault(key[, default]) - если ключ key есть в словаре, то возвращается значение по ключу. Если такого ключа нет, то в словарь вставляется элемент с ключом key и значением default, если default не определен, то по умолчанию присваивается None	>>> d = {"A1":"123", "A2":"456"}
	>>> d.setdefault("A3", "777")
	'777'
	>>> print(d)
	{'A2': '456', 'A3': '777', 'A1': '123'}
	>>> d.setdefault("A1")
	'123'
	>>> print(d)
	{'A2': '456', 'A3': '777', 'A1': '123'}

update([other]) - обновляет словарь парами (key/value) из other, если ключи уже существуют, то обновляет их значения	>>> d = {"A1":"123", "A2":"456"}
	>>> d.update({"A1":"333", "A3":"789"})
	>>> print(d)
	{'A2': '456', 'A3': '789', 'A1': '333'}

values() - возвращает значения элементов словаря	>>> d = {"A1":"123", "A2":"456"}
	>>> d.values()
	dict_values(['456', '123'])

Раздаточныйматериал № 111

```
for key, value in nums.items():
    print(key, 'is', value)
```

Результат

1 is one
2 istwo
3 isthree

Раздаточныйматериал № 112

```
# Программаформируетсловарь,
состоящийизуникальныхсимволовиихповторенийв
# заданномпредложении
s_n = {}
for i in 'ИзучаемязыкпрограммированияПитон':
    if i not in s_n:
```

```
        s_n[i] = 1
    else:
        s_n[i] += 1
print (s_n)
for i in s_n.items():
    print(i)
```

Результат
{'И': 1, 'з': 2, 'у': 1, 'ч': 1, 'а': 3, 'е': 1, 'м': 3, ' ': 3, 'я': 2, 'ы': 1, 'к': 1, 'п': 1, 'р': 3, 'о': 3, 'г': 1, 'и': 3, 'в': 1, 'н': 2, 'П': 1, 'т': 1}
('И', 1)
('з', 2)
('у', 1)
('ч', 1)
('а', 3)
('е', 1)
('м', 3)
(' ', 3)
…

```
# Программапреобразуетвсловарьзначенияизстроки
student = {}
inf = 'ИвановИванИвановичПОКС-29 5 3 5 5 4'
inf = inf.split()
student['Фамилия'] = inf[0]
student['Имя'] = inf[1]
student['Отчество'] = inf[2]
student['Группа'] = inf[3]
student['Оценки'] = []
for i in inf[4:]:
    student['Оценки'].append(int(i))
print(student)
```

Результат
{'Фамилия': 'Иванов', 'Имя': 'Иван', 'Отчество': 'Иванович', 'Группа': 'ПОКС-29', 'Оценки': [5, 3, 5, 5, 4]}

```
# Добавить в словарь новый элемент elephant = слон, если
его еще нет в словаре
a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица',
'mouse': 'мышь'}
if 'elephant' not in a:
    a['elephant'] = 'слон'
print(a)
```

Результат
{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь', 'elephant': 'слон'}

Раздаточный материал № 113

```
 # целые числа
```

```
b = {1, 3, 5, 7, 1, 3, 5, 7}
print(b)
{1, 3, 5, 7}

# строки
c = {'ok', 'no', 'yes', 'ok', 'no', 'yes'}
print(c)
{'ok', 'no', 'yes'}
```

Раздаточный материал № 114

```
 d = set('множество')
print(d)
{'о', 'ж', 'т', 'н', 'с', 'м', 'е', 'в'}
```

Раздаточный материал № 115

```
f = set([11, 12, 13, 14, 12, 13])
print(f)
{11, 12, 13, 14}

e = set(['list', 'set', 'and', 'set'])
print(e)
{'and', 'list', 'set'}
```

Раздаточный материал № 116

```
q = set()
```

Раздаточный материал № 117

```
w = {3, 4, 9}
w.add(11)
print(w)
{11, 9, 3, 4}
```

Раздаточный материал № 118

```
t = {5, 6, 10}
t.update([12, 15, 17])
print(t)
{5, 6, 10, 12, 15, 17}
```

Раздаточный материал № 119

```
y = {20, 21, 22}
y.discard(21)
print(y)
{20, 22}
```

Метод remove делает тоже самое.

```
p = {27, 28, 29}
```

p.remove(27)
print(p)
{28, 29}

Раздаточный материал № 120

g = {27, 28, 29}
g.pop()
print(g)
{28, 29}

Раздаточный материал № 121

h = {31, 32, 33}
h.clear()
print(h)
set()

Раздаточный материал № 122

k = {34, 35, 36, 37}
print(len(k))
4

Раздаточный материал № 123

l = {38, 39, 40, 41}
z = {42, 39, 40, 43}
print(l&z)
{40, 39}

x = {44, 45, 46, 47}
c = {48, 49, 50, 51}
print(x & c)
set()

Раздаточный материал № 124

v = {52, 53, 54, 55}
b = {55, 56, 57, 58}
print(v | b)
{52, 53, 54, 55, 56, 57, 58}

Метод union является аналогичным способом объединения множеств.

n = {59, 60, 61}
m = {62, 63, 64}

print (n.union(m))
{64, 59, 60, 61, 62, 63}

Раздаточныйматериал № 125

one = {71, 72, 73}
two = {71, 72, 75}
print(one - two)
{73}

Раздаточныйматериал № 126

one = {71, 72, 73}
two = {71, 72, 73}
print(one == two)
True

q = {65, 66, 67}
z = {68, 69, 70}
print(q == z)
False

Раздаточныйматериал № 127

colors = {"red", "green", "blue"}
for color in colors:
print(color)

Результат (порядок может быть другим):
red
green
blue

Раздаточный материал № 128

```
# Существует набор продуктов, продаваемых в нескольких
магазинах города.
# Определить: какие продукты есть во всех магазинах города;
полный набор продуктов в городе

magnit = {'хлеб', 'молоко'}
troyka = {'хлеб','сыр','масло'}
karusel ={'хлеб','сыр'}

print ('продукты есть во всех магазинах города:  ',
magnit&troyka&karusel)
print ('полный набор продуктов в городе:  ',
magnit|troyka|karusel)
```

```
# добавить название цвета если его нет во множестве

colors = {"red", "green", "blue"}
if "yellow" not in colors:
    colors.add("yellow")
print(colors)
```

Раздаточный материал № 129

```
f1 = open('new_file_1.txt')
print('Читаем и выводим на экран первые 10 байт или символов')
print(f1.read(10))
f1.close()

f1 = open('new_file_1.txt')
print('Читаем и выводим на экран весь файл')
print(f1.read())
f1.close()

f1 = open('new_file_1.txt')
print('Читаем и выводим на экран весь файл с помощью for')
for line in f1:
    print(line, end='')
print(type(f1.read()))
f1.close()
```

Раздаточный материал № 130

Метод	Описание
readline()	Чтение файла построчно
readlines()	Считывает сразу все строки и создает список

Раздаточный материал № 131

```
print('Запишем в файл структуру данных - список')
l = [' tree', ' four']
f2 = open('data.txt', 'w')
f2.write('one')
f2.write(' two')
f2.writelines(l)
f2.close()
f2 = open('data.txt')
print(f2.read())
```

```
print(type(f2.read())) # получаем тип - строка
f2.close()
```

Раздаточный материал № 132

```
# содержимое файла data_2.txt:
# зима
# весна
# лето
# осень

nums = []
for i in open('data_2.txt', encoding='UTF-8'):
    nums.append(i[:-1])
print(nums)
print('получаем тип', type(nums))
```

Результат
['зима', 'весна', 'лето', 'осень']

Раздаточный материал № 133

```
# 1
# Средствами языка Python сформировать текстовый файл
(.txt) содержащий
# последовательность из целых положительных и отрицательных
чисел.
# Сформировать новый текстовый файл (.txt) следующего вида,
# предварительно выполнив требуемую обработку элементов:
# Исходные данные:
# Количество элементов:
# Максимальный элемент:
# Количество отрицательных элементов:

# Запишем в файл data_3.txt структуру данных - список
l = ['-99 6 12 -36 20 45 100 -15']
f3 = open('data_3.txt', 'w')
f3.writelines(l)
f3.close()
```

```
# Дублируем список в новый файл data_4.txt
f4 = open('data_4.txt', 'w')
f4.write('Исходные данные:  ')
f4.write('\n')
f4.writelines(l)
f4.close()

# разбиваем строку и ее значения преобразуем в числа
f3 = open('data_3.txt')
k = f3.read()
k = k.split()
for i in range(len(k)):
    k[i] = int(k[i])
f3.close()

# Ищем максимальный элемент и количество отрицательных
элементов
# в файле data_3.txt и записываем в файл data_4.txt
f3 = open('data_3.txt')
max, t = 0, 0
for i in range(len(k)):
    max = max if max > k[i] else k[i]
    if k[i] < 0:
        t += 1

f4 = open('data_4.txt', 'a')  # открываем файл для дозаписи
f4.write('\n')
print('Количество элементов:  ', len(k), 'Максимальный
элемент:  ', max, file=f4)
f4.close()


# 2
# Из предложенного текстового файла (text18-1.txt) вывести
на экран его содержимое,
# посчитать и вывести количество строк и количество букв
'ж'.
# Сформировать новый файл, в который поместить текст,
предварительно поменяв
# местами первую и четвертую строки.

t = 0
d = 0
for i in open('text18-1.txt', encoding='UTF-8'):
    print(i, end='')
    t += 1
    for j in i:
        if j == 'ж':
            d += 1
```

```
print(end='\n')
print('Количествострок:    ', t, end='\n')
print('Количествобукв "ж" :    ', d, end='\n')


f1 = open('text18-1.txt', encoding='UTF-8')
l = f1.readlines()
l[0], l[3] = l[3], l[0]
f1.close()

f2 = open('text18-2.txt', 'w')
f2.writelines(l)
f2.close()
```