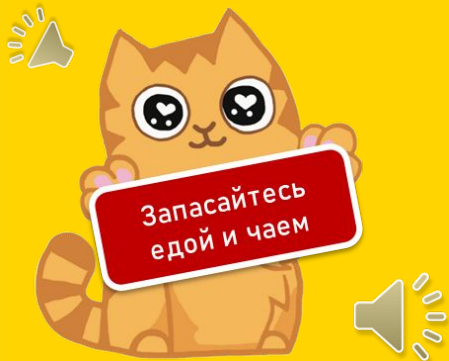


Занятие №1

«Функции. Передача параметров,
возвращение результата»

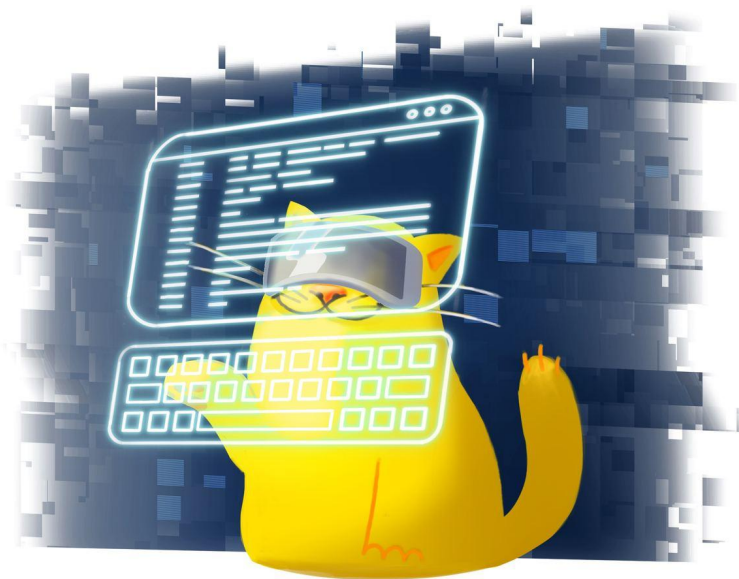




О чем будем говорить сегодня

Тема: функции, передача параметров, возвращение результата

Цель занятия: изучить команды по созданию функций и передаче параметров





Результаты работы

- Научимся создавать свои функции
- Научимся определять аргументы функций
- Научимся разделять глобальные и локальные переменные



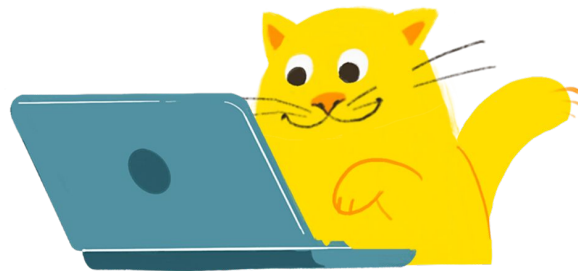
Это нужно знать

Функция – часть программного кода, к которой можно обратиться из другого места программы.

Аргументы функции – компоненты функции.

Самые применяемые функции:

- встроенные
- пользовательские (свои)



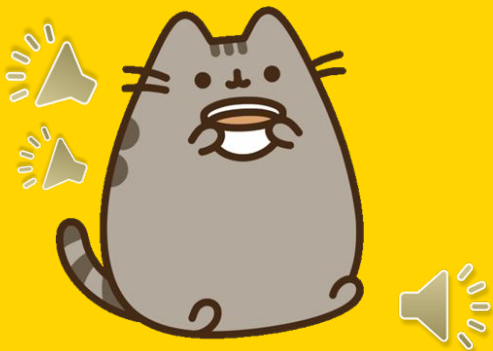


Что помните с прошлого раза?

- | |
|---|
| 1. Какую роль выполняет цикл while в языке <i>Python</i> ? |
| 2. Что такое оператор прерывания цикла ? |
| 3. Как можно передавать аргументы циклу? |
| 4. Какую роль выполняет ключевое слово def в языке <i>Python</i> ? |
| 5. Как можно использовать оператор and в языке <i>Python</i> ? |

Функции

Учимся писать свои функции





Немного теории: функции

```
1  # синтаксис функций
2  # def имя_функции(параметры_функции):
3  # инструкции
```

Функции способны работать с аргументами и возвращать результат. Объявить **функцию** можно при помощи слова **def**.

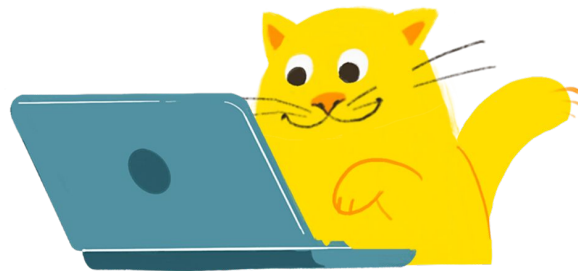


Пример использования функции

Функция вычисляет сумму двух чисел

```
1  # объявление функции
1  usage
2  def sum():
3      # инструкции
4      x = 10
5      y = 15
6      # вывод результата
7      print('Сумма:', x+y)
8  # вызов функции
9  sum()
```

```
07.08 >
python.exe C:\Users\user\PycharmProjects
\pythonProject3\07.08.py
Сумма: 25
```





Начинаем практику: функции

Задание.

Посмотрите на код функции и предположите - что она выведет?

```
1.py third.py 07.08.py
1 1 usage
2  def sum_2():
3      # инструкции
4      x = 10
5      y = 15
6      z = 7
7      # вывод результата
8      print('Результат:', x-y+z)
9  # вызов функции
10 sum_2()
```



Время на выполнение - 5 минут





Посмотрим на код

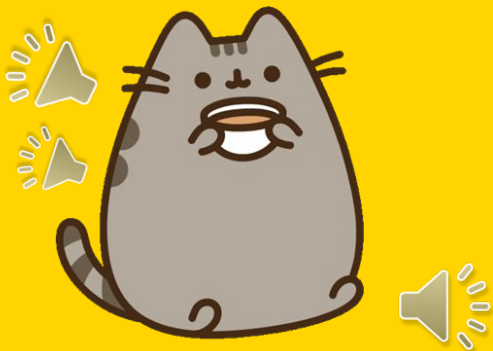
```
1.py x third.py x 07.08.py x
1 1 usage
2 1 def sum_2():
3   # инструкции
4   x = 10
5   y = 15
6   z = 7
7   # вывод результата
8   print('Результат:', x-y+z)
9   # вызов функции
10 sum_2()

07.08 x
python.exe C:\Users\user\PycharmProjects\pythonProject3\07.08.py
Результат: 2
```

Здесь функция работает с переменными **'x'**, **'y'** и **'z'**

Аргументы функции

Знакомимся с аргументами





Немного теории: аргументы функции

Функция может принимать произвольное количество аргументов или не принимать их совсем:

```
1.py x third.py x 07.08.py x
1 usage
1 def func_3(x, s):
2     # вывод результата
3     print('Результат:', x+s)
4     # вызов функции
5     func_3(x: 10, s: 15)
6
C:\Users\user\PycharmProjects
\pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
\pythonProject3\07.08.py
Результат: 25
```

```
1.py x third.py x 07.08.py x
1 usage
1 def func_3():
2     # инструкции
3     x = 10
4     s = 15
5     # вывод результата
6     print('Результат:', x+s)
7     # вызов функции
8     func_3()
9
C:\Users\user\PycharmProjects
\pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
\pythonProject3\07.08.py
Результат: 25
```



Немного теории:

аргументы функции

Схема работы функции:

```
def имя_функции (параметры):
```

```
    инструкции
```

```
    ...
```

```
# вызов функции
```

```
имя_функции (аргументы)
```



Начинаем практику:

аргументы функции

Задание.

Создайте функцию, в которой возвращается произведение переменных **'q'**, **'w'** и **'e'** с типом данных **'int'**.

Значения переменных указываются при вызове функции.



Время на выполнение - 5 минут



Решение задания

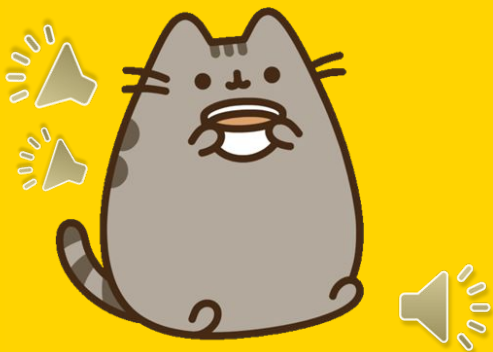
```
1.py x third.py x 07.08.py x
1 usage
2 def func_5(q, w, e):
3     # вывод результата
4     print('Результат:', q*w*e)
5     # вызов функции
6     func_5(q: 5, w: 6, e: 7)

C:\Users\user\PycharmProjects
pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
pythonProject3\07.08.py
Результат: 210
```



Глобальные и локальные переменные

**Знакомимся с новыми типами
переменных**





Немного теории:

глобальные и локальные переменные

```
1.py x third.py x 07.08.py x
2  # объявление глобальной переменной
3  n = "Python"
4  1 usage
5  def first():
6      print('This is', n)
7  1 usage
8  def second():
9      print('This was', n)
10 first()
    second()
```

```
07.08 x
\pythonProject3\07.08.py
This is Python
This was Python
```

Глобальные переменные
могут быть объявлены вне
функции, но использовать ей



Пример использования глобальные и локальные переменные

```
1.py x third.py x 07.08.py x
1
2 # объявление глобальной переменной
3 x = 23
4
5 1 usage
6
7 def first():
8     print('Значение:', x)
9
10 first()
```

```
07.08 x
C:\Users\user\PycharmProjects
pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
pythonProject3\07.08.py
Значение: 23
```

Здесь **глобальная**
переменная 'x' подставляется
в функцию **'first'**



Начинаем практику:

глобальные и локальные переменные

Задание.

Создайте программу, в которой есть две глобальные переменные и одна функция. В данной функции происходит сложение переменных и возвращение суммы.



Время на выполнение - 5 минут



Решение задания

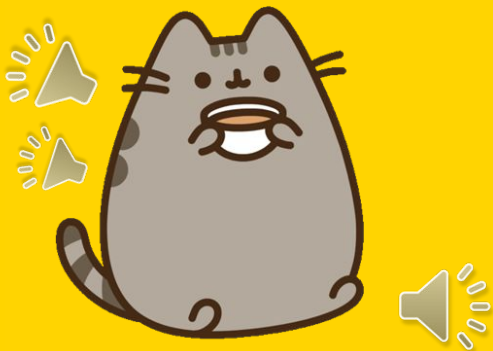
```
1  global a, b
2  a = 12
3  b = 15
   1 usage
4  def func():
5      print('Результат:', a+b)
6
7  func()
```

C:\Users\user\PycharmProjects
pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
pythonProject3\07.08.py

Результат: 27

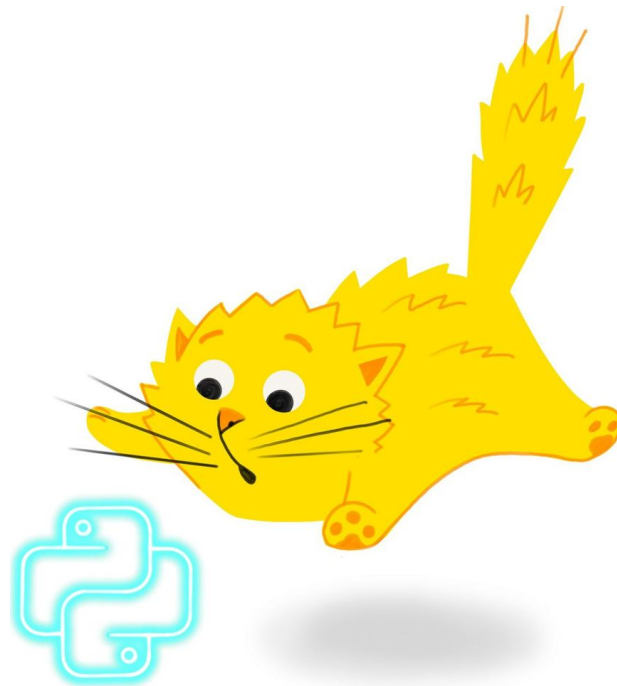
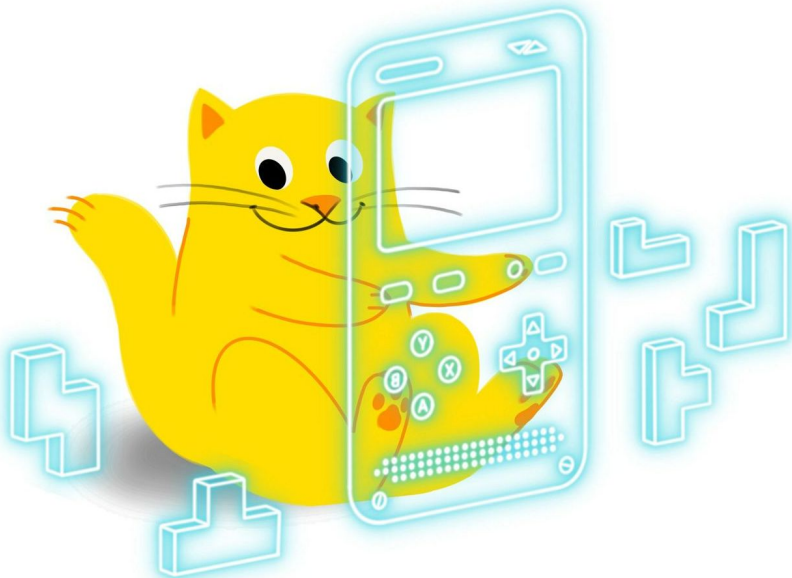


Перерыв **10** минут





ПРАКТИЧЕСКИЕ ЗАДАЧИ





Практика. Задача №1

Написать программу с одной глобальной переменной `c`, и функцией, которая считает произведение трех целых чисел (включая переменную `c`).





Решение задачи №1

Напишем код для решения
и посмотрим на вывод:

```
1 global c
2 c = 123
   1 usage
3 def task_1():
4     a = 12
5     b = 20
6     print('Результат:', a+b+c)
7
8 task_1()
```

C:\Users\user\PycharmProjects
\pythonProject3\venv\Scripts\python
.exe C:\Users\user\PycharmProjects
\pythonProject3\07.08.py
Результат: 155



Практика. Задача №2

Написать программу с двумя глобальными переменными (имена задаются произвольно), которая вычисляет разность этих переменных.





Решение задачи №2

Напишем код для решения
и посмотрим на вывод:

```
1  global c, x
2  c = 123
3  x = 56
   1 usage
4  def task_2():
5      print('Разность:', c-x)
6
7  task_2()
```

task_20

07.08 ×

↑
↓
↺
↻
🗑

Разность: 67

Process finished with exit code 0



Практика. Задача №3

Создать программу с использованием цикла **while** и переменной 'y', значение которой, изначально, равно 2. Данный цикл выводит значения переменной 'y' в третьей степени до тех пор, пока значение переменной 'y' не станет равным 10. При каждой итерации значение переменной 'y' увеличивается на 2.





Решение задачи №3

Напишем код для решения и посмотрим на вывод:

```
1  # цикл 'while'
2  y = 2
3  while y < 10:
4      print('Значение:', y**3)
5      y += 2
```

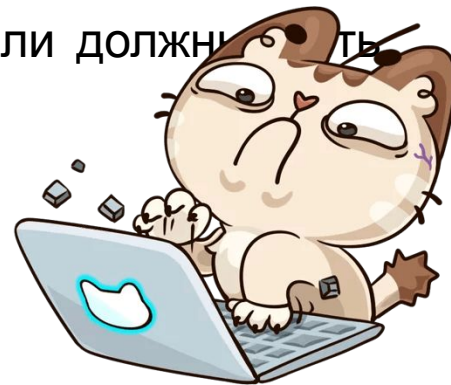
07.08 x

Значение: 8
Значение: 64
Значение: 216
Значение: 512



Практика. Задача №4

Создать программу, в которой реализован оператор выхода из цикла **break**. Изначально вводится переменная 's' со значением 0. При каждой итерации цикла значение переменной 's' увеличивается на единицу. Когда значение переменной 's' становится равным 5, оператор **break** завершает работу цикла (на консоли должны быть выведены значения с 1 по 4 включительно).





Решение задачи №4

Напишем код для решения
и посмотрим на вывод:

```
1  s = 0
2  while s < 9:
3      s += 1
4      if s == 5:
5          break
6      print(f's = {s}')
7
```

g: 07.08 x

Debugger Console

```
s = 1
s = 2
s = 3
s = 4
```



Практика. Задача №5

Создать программу с использованием цикла **for**. Данный цикл перебирает значения переменной 'a' от 3 до 7 и переменной 'b' от 10 до 13 и выводит соответствующие значения на консоль.





Решение задачи №5

Напишем код для решения
и посмотрим на вывод:

```
1  for a in range(3, 8):
2      print('a =', a)
3  for b in range(10, 14):
4      print('b =', b)
```



```
↑
↓
a = 3
a = 4
a = 5
a = 6
a = 7
b = 10
b = 11
b = 12
b = 13
```




Практика. Задача №6

Создать программу, в которой вводятся списки **'x' = [1, 2, 3, 4, 5, 6]** и **'a' = [1, 3, 5, 7, 9]**. Необходимо найти максимальный элемент в списке **'a'** и минимальный элемент в списке **'x'**.

Данные значения вывести на консоль.





Решение задачи №6

Напишем код для решения и посмотрим на вывод:

```
1 x = [1, 2, 3, 4, 5, 6]
2 a = [1, 3, 5, 7, 9]
3
4 print('Список "x":', x)
5 print('Список "a":', a)
6 print('Минимальный элемент в списке "x":', min(x))
7 print('Максимальный элемент в списке "x":', max(a))
8
9
```

```
↑
↓
Список "x": [1, 2, 3, 4, 5, 6]
Список "a": [1, 3, 5, 7, 9]
Минимальный элемент в списке "x": 1
Максимальный элемент в списке "x": 9
```



Практика. Задача №7

Создать программу, в которой реализован оператор выхода из цикла **break**. Изначально вводится переменная '**num**' со значением **0**. При каждой итерации цикла значение переменной '**num**' увеличивается на единицу. Когда значение переменной '**num**' становится равным **3**, оператор **break** завершает работу цикла (на консоли должны быть выведены оба значения: **1** и **2**).





Решение задачи №7

Напишем код для решения и посмотрим на вывод:

The screenshot shows an IDE with two windows. The top window, titled '07.08.py', contains the following Python code:

```
1 num = 0
2 # реализация оператора 'break'
3 while num < 5:
4     num += 1
5     if num == 3:
6         break
7     print(f'num = {num}')
```

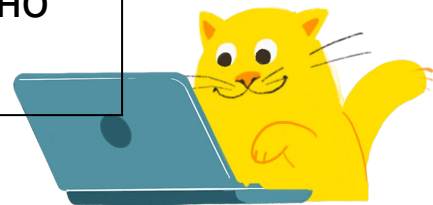
The bottom window, titled '\07.08.py', shows the output of the code:

```
num = 1
num = 2
```



Закрепление знаний

1. Какую роль выполняет ключевое слово def в языке <i>Python</i> ?
2. Что такое функция ?
3. Как можно передавать аргументы функции?
4. Какую роль выполняют функции в языке <i>Python</i> ?
5. Какое максимальное количество аргументов можно передать функции на языке <i>Python</i> за один раз?



Спасибо за внимание!
До новых встреч!

