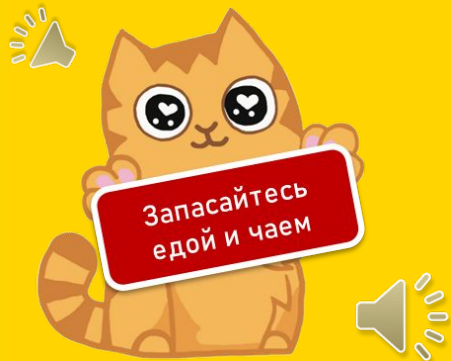


Занятие №1

«Логический тип данных»



Сроки получения материалов для преподавателей

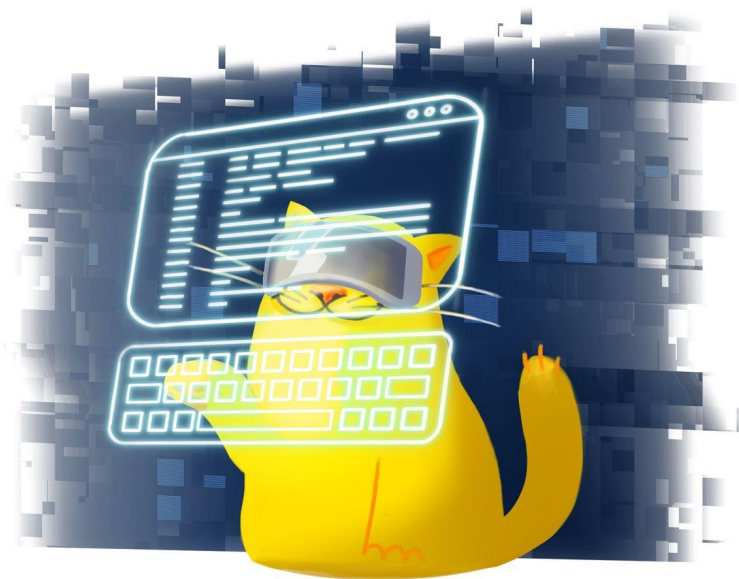
Материал	Срок
Презентация каждого занятия	12.09
Методические пособия ко всем модулям	19.09
Видео теории каждого занятия	26.09
Тесты после каждого урока (с ответами), Итоговый тест после модуля	26.09



О чем будем говорить сегодня

Тема: логический тип данных

Цель занятия: изучить логический тип данных





Результаты работы

- Научимся сравнить переменные между собой;
- Познакомимся с логическими операторами;
- Познакомимся с циклом **while**.



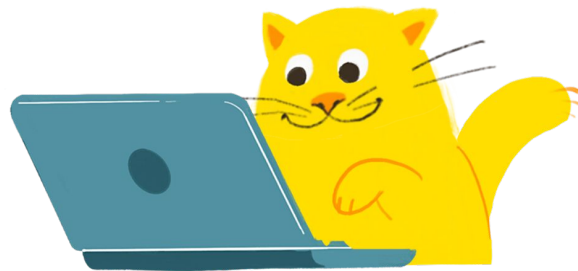
Это нужно знать

Программа – последовательность действий или операций, которая приводит к определенному результату.

Операция – способ записи некоторых действий.

Наиболее часто применяются:

- арифметические
- логические
- строковые операции



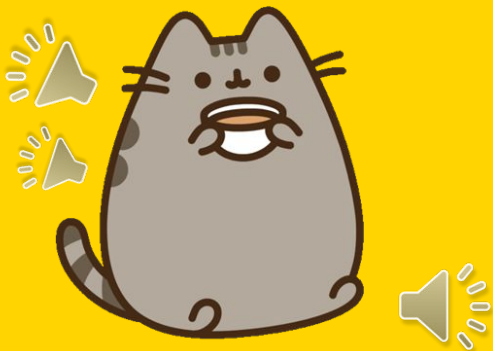


Что помните с прошлого раза?

1. Какую роль выполняет команда str в языке <i>Python</i> ?
2. Что такое арифметическая операция ?
3. Как можно передавать аргументы программе?
4. Какую роль выполняет функция dict() в языке <i>Python</i> ?
5. Какое максимальное количество аргументов можно передать программе на языке <i>Python</i> за один раз?

Операции сравнения

Учимся сравнивать переменные
между собой





Немного теории: **операции сравнения**

Ниже приведена таблица, в которой указаны основные операции сравнения, которые можно применять:

Операция сравнения	Функция
==	Возвращает True , если оба операнда равны. Иначе возвращает False
!=	Возвращает True , если оба операнда НЕ равны. Иначе возвращает False
>	Возвращает True , если первый операнд больше второго
<	Возвращает True , если первый операнд меньше второго
>=	Возвращает True , если первый операнд больше или равен второму
<=	Возвращает True , если первый операнд меньше или равен второму



Пример использования операции сравнения

```
pythonProject3 - 07.08.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject3 07.08.py
1 # использование оператора '=='
2 a = 10
3 b = 8 == 10
4 c = 10 == 10
5
6 # вывод данных
7 print('Переменная "a":', a)
8 print('Переменная "b":', b)
9 print('Переменная "c":', c)
10
Run: 07.08
C:\users\user\python\projects\pythonProject3
\07.08.py
Переменная "a": 10
Переменная "b": False
Переменная "c": True
```

В строке 2 вводится переменная с именем **'a'**, в нее записывается значение **10**. В строке 3 вводится переменная с именем **'b'** и записывается значение **8**, после чего применяется оператор **'=='**, справа от него пишется значение **10**. Значение данной переменной будет **False**.



Вывод при помощи операторов сравнения

Еще вариант исполнения

```
1  # использование оператора '!='
2  a = 15
3  b = 8 != 10
4  c = 12 != 12
5
6  # вывод данных
7  print('Переменная "a":', a)
8  print('Переменная "b":', b)
9  print('Переменная "c":', c)
10
```

```
C:\users\user\python\projects\python\projects
\07.08.py
Переменная "a": 15
Переменная "b": True
Переменная "c": False
```

В данном примере наоборот - если значение после знака '=' будет точно таким же, как и значение после оператора присваивания '!=', то значение переменной будет **False**. Принцип работы такой же, у нас есть три переменных со значениями **15**, **8** и **12**.



Вывод при помощи операторов сравнения

Еще вариант исполнения

```
1  # использование операторов '>' и '<'
2  a = 15
3  b = 8
4
5  # вывод данных
6  print('Переменная "a":', a)
7  print('Переменная "b":', b)
8  print('Переменная "a" больше, чем "b":', a > b)
9  print('Переменная "a" меньше, чем "b":', a < b)
10
```

```
07.08 .py
↑
↓
Переменная "a": 15
Переменная "b": 8
Переменная "a" больше, чем "b": True
Переменная "a" меньше, чем "b": False
```

В строке 8 мы задаем вопрос консоли - значение переменной 'a' больше, чем значение переменной 'b'. Это правда, поэтому в выводе данной строки можно заметить значение **True**. Обратный вопрос задается в строке 9, результатом является значение **False**.



Вывод при помощи операторов сравнения

Еще вариант исполнения

Пример использования других операторов сравнения:

```
1  # использование операторов '>=' и '<='
2  c = 13
3  d = 9
4
5  # вывод данных
6  print('Переменная "c":', c)
7  print('Переменная "d":', d)
8  print('Переменная "c" больше или равна переменной "d":', c >= d)
9  print('Переменная "c" меньше или равна переменной "d":', c <= d)
10
```

```
C:\users\user\python\projects\python\projects\07.08.py
Переменная "c": 13
Переменная "d": 9
Переменная "c" больше или равна переменной "d": True
Переменная "c" меньше или равна переменной "d": False
```



Начинаем практику: процесс именования

Задание.

Посмотрите на код программы и предположите - что выведет данная программа?

```
1  # ввод переменных
2  d = 15
3  e = 10
4
5  # вывод данных
6  print('Переменная "d":', d)
7  print('Переменная "d" больше переменной "e":', d > e)
8  |
```



Время на выполнение - 5 минут





Начинаем практику: решение задания

```
1  # ввод переменных
2  d = 15
3  e = 10
4
5  # вывод данных
6  print('Переменная "d":', d)
7  print('Переменная "d" больше переменной "e":', d > e)
8
```

```
C:\Users\user\PycharmProjects\pythonProject3\venv\scripts
python.exe C:\Users\user\PycharmProjects\pythonProject3\07
.08.py
```

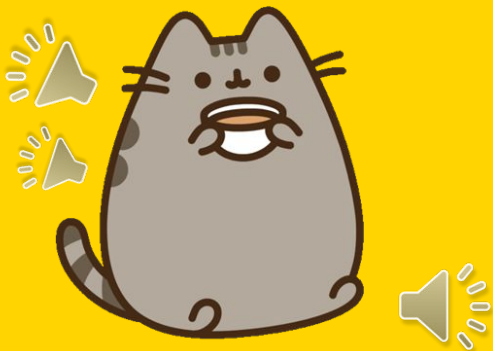
```
Переменная "d": 15
Переменная "d" больше переменной "e": True
```

Данная программа
выводит значение
переменной **'d'** и
значение **True**



Логические операции

Учимся работать с логикой в коде





Немного теории: **логические операции**

Для создания составных условных выражений любой сложности применяются специальные операции, которые называются **логическими операциями**.

Оператор **and** (логическое **умножение**) применяется для двух операндов

```
1  # ввод данных
2  first = 23
3  second = 59
4
5  # использование лог. оператора 'and'
6  res = first > 20 and second == 59
7
8  # вывод данных
9  print('Переменная "first":', first)
10 print('Переменная "second":', second)
11 print('Переменная "res":', res)
12
13
```

```
Переменная "first": 23
Переменная "second": 59
Переменная "res": True
```




Пример использования логические операции

Оператор **or** (логическое сложение), также может применяться к двум операндам с одинаковым типом

```
1  # Ввод данных
2  first = 23
3  second = False
4
5  # использование лог. оператора 'or'
6  res = first > 22 or second
7
8  # вывод данных
9  print('Переменная "first":', first)
10 print('Переменная "second":', second)
11 print('Переменная "res":', res)
12
13
```

```
↑
↓
Переменная "first": 23
Переменная "second": False
Переменная "res": True
```



Пример использования логические операции

Оператор **not** (логическое отрицание), который возвращает значение **True**, если выражение равно **False**:

```
1  # ввод данных
2  first = 23
3  second = False
4
5  # использование лог. оператора 'not'
6  # вывод данных
7  print('Переменная "first":', first)
8  print('Переменная "second":', second)
9  print('Выражение "not 0":', not 0)
10
11
```

```
↑  Переменная "first": 23
↓  Переменная "second": False
↑  Выражение "not 0": True
```



Пример использования: логические операции

С помощью оператора **in** возможно проверить - есть ли в строке подстрока (набор из меньшего числа символов, чем вся строка:

```
1 # Ввод данных
2 mes = 'Python language'
3 first_op = 'Python'
4 second_op = 'SQL'
5
6 # использование оператора 'in'
7 # Вывод данных
8 print('Строка "mes":', mes)
9 print('Входит ли подстрока "first_op":', first_op in mes)
10 print('Входит ли подстрока "second_op":', second_op in mes)
11
```

Строка "mes": Python language
Входит ли подстрока "first_op": True
Входит ли подстрока "second_op": False

```
1 # Ввод данных
2 mes = 'Python language'
3 first_op = 'Pascal'
4 second_op = 'SQL'
5
6 # использование оператора 'in'
7 # Вывод данных
8 print('Строка "mes":', mes)
9 print('Входит ли подстрока "first_op":', first_op in mes)
10 print('Входит ли подстрока "second_op":', second_op in mes)
11
```

Строка "mes": Python language
Входит ли подстрока "first_op": False
Входит ли подстрока "second_op": False



Начинаем практику! **логические операции**

Задание.

Посмотрите на код программы и предположите - что выведет данная программа?

```
1  # ВВОД ДАННЫХ
2  first = 'This is Python language'
3  first_op = 'Pascal'
4  second_op = 'SQL'
5
6  # использование оператора 'in'
7  # ВЫВОД ДАННЫХ
8  print('Строка "mes":', first)
9  print('Входит ли подстрока "first_op":', first_op in first)
10 print('Входит ли подстрока "second_op":', second_op in first)
11
```



Время на выполнение - 5 минут





Начинаем практику: логические операции

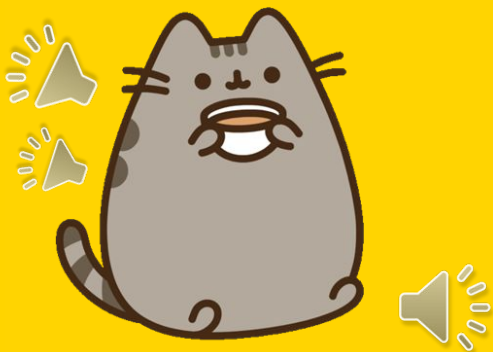
```
1  # ввод данных
2  first = 'This is Python language'
3  first_op = 'Pascal'
4  second_op = 'SQL'
5
6  # использование оператора 'in'
7  # вывод данных
8  print('Строка "mes":', first)
9  print('Входит ли подстрока "first_op":', first_op in first)
10 print('Входит ли подстрока "second_op":', second_op in first)
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
Строка "mes": This is Python language
Входит ли подстрока "first_op": False
Входит ли подстрока "second_op": False
```

Данная программа выведет значение переменной **'first'** и дважды значение **False**

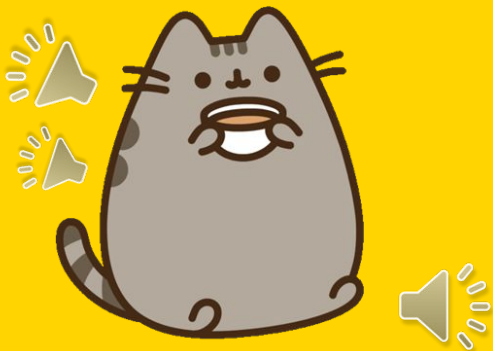


Перерыв **10** минут



Цикл **while**

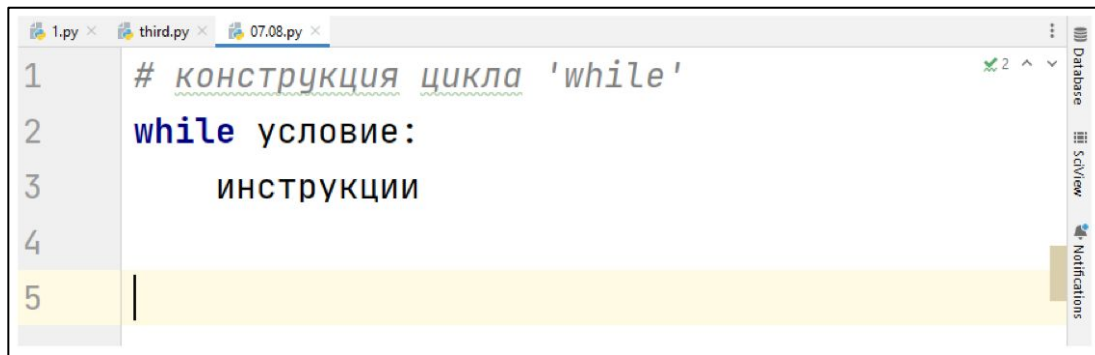
Учимся использовать цикл 'while'





Немного теории: цикл **while**

Цикл **while** проверяет истинность некоторого условия, и если условие истинно, то выполняются инструкции цикла



```
1  # конструкция цикла 'while'
2  while условие:
3      инструкции
4
5  |
```

После ключевого слова **while** указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций



Пример использования цикл **while**

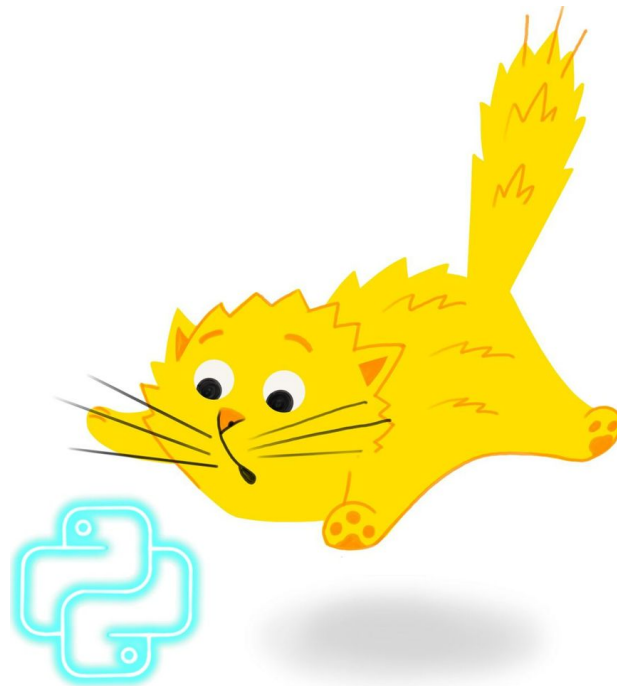
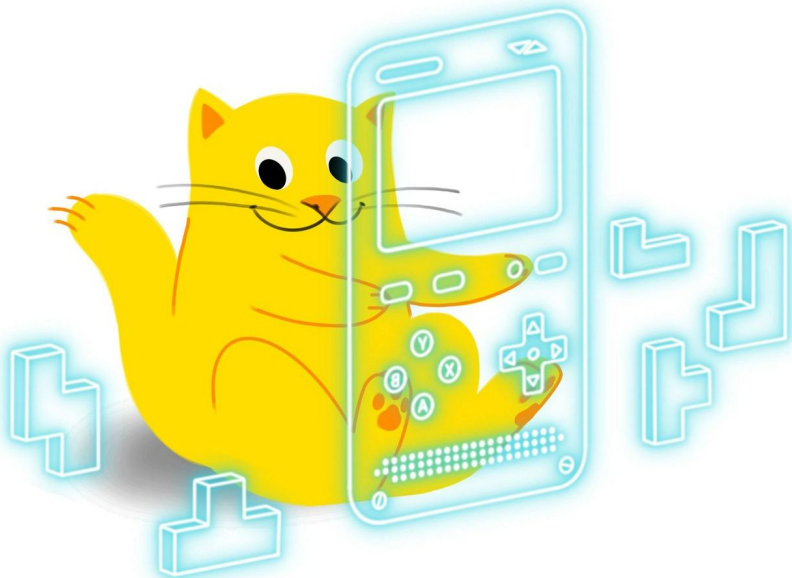
```
1  # применение цикла 'while'
2  number = 1
3
4  while number < 4:
5      print(f'number = {number}')
6      number += 1
7  print('Работа программы завершена!')
```

```
number = 1
number = 2
number = 3
Работа программы завершена!
```

В данном примере цикл **while**
будет выполняться, пока
переменная **'number'** меньше **5**.



ПРАКТИЧЕСКИЕ ЗАДАЧИ





Практика. Задача №1

Создать программу, в которой используется следующая строка: **'mes'** = **'Python'**. Кроме того, в программе используется подстрока: **'first_op'**. В данной программе необходимо реализовать оператор **in** и проверить - входит ли подстрока с именем **'first_op'** в строку **'mes'**. Результаты продемонстрировать на консоли.





Решение задачи №1

Напишем код для решения данной практической задачи:

```
1  # Ввод строк
2  mes = 'Python'
3  first_op = 'Pascal'
4
5  # вывод данных
6  print('Строка "mes":', mes)
7  print('Входит подстрока "first_op" в строку "mes"?',
8        first_op in mes)
9
10
```

C:\Users\user\PycharmProjects\pythonProject3\07.08.py
Строка "mes": Python
Входит подстрока "first_op" в строку "mes"? False



Практика. Задача №2

Создать программу, в которой используются переменные с именами 'а', 'с' и 'е'. Необходимо использовать специальный оператор '!=', относительно двух пар переменных - 'а' и 'с', а затем 'а' и 'е'. Результаты продемонстрировать на консоли.





Решение задачи №2

Напишем код для решения данной практической задачи:

```
1  # ввод данных
2  a = 17
3  c = 8 != a
4  e = 17 != a
5
6  # вывод данных
7  print('Переменная "a"', a)
8  print('Оператор "!=" к переменным "a" и "c":', c)
9  print('Оператор "!=" к переменным "a" и "e":', e)
10
11 |
```

```
07.08
Переменная "a" 17
Оператор "!=" к переменным "a" и "c": True
Оператор "!=" к переменным "a" и "e": False
```



Практика. Задача №3

Создать программу, в которой пользователь вводит три строки из символов. Необходимо вывести данные строки отдельно на консоль.





Решение задачи №3

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:

```
1  # Ввод данных
2  a = str(input('Введите "a": '))
3  c = str(input('Введите "c": '))
4  e = str(input('Введите "e": '))
5
6  # вывод данных
7  print(a)
8  print(c)
9  print(e)
10
11
```

```
↑
↓
Введите "a": Это наше
Введите "c": четвертое занятие
Введите "e": по языку Python!
Это наше
четвертое занятие
по языку Python!
```




Практика. Задача №4

Создать программу с использованием цикла **while**. Цикл **while** работает с переменной **'z'**, значение которой изначально равно 3. Данный цикл выводит значение переменной **'z'** во второй степени. При каждой итерации значение переменной **'z'** увеличивается на единицу и выводит квадратные значения переменной **'z'** до 6 включительно.





Решение задачи №4

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:

```
1.py x third.py x 07.08.py x
1  # ввод данных
2  z = 3
3
4  # использование цикла 'while'
5  while z < 7:
6      print('Результат:', z**2)
7      z += 1
8  print('Работа программы завершена!')
9
10 |

07.08 x
↑ ↓
Результат: 9
Результат: 16
Результат: 25
Результат: 36
Работа программы завершена!
```



Практика. Задача №5

Создать программу, в которой используются переменные с именами 'с' и 'd'. Необходимо использовать специальный оператор '>=', относительно данных переменных. Результаты продемонстрировать на консоли.





Решение задачи №5

Напишем код для решения данной практической задачи:

```
1  # ввод данных
2  c = 13
3  d = 10
4
5  # вывод данных
6  print('Переменная "c" больше или равна переменной "d":', c >= d)
7  print('Переменная "d" больше или равна переменной "c":', d >= c)
8
9  |
```

C:\Users\user\PycharmProjects\pythonProject3\07.08.py

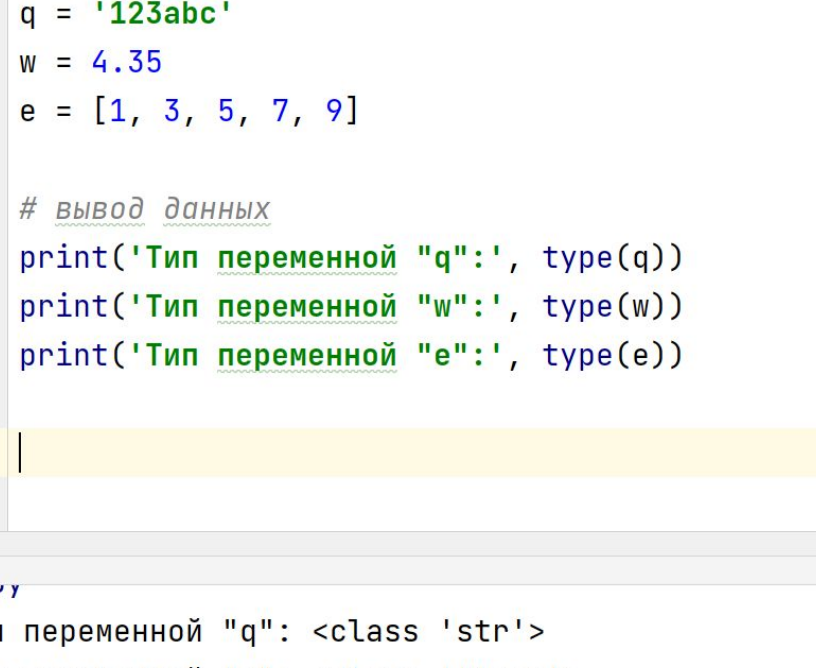
Переменная "c" больше или равна переменной "d": True
Переменная "d" больше или равна переменной "c": False



Практика. Задача №6

Создать программу, в которой используются: переменная **'q'** с типом **str**, переменная **'w'** с типом **float** и переменная **'e'** с типом **list**. Необходимо при помощи функции **type()** вывести на консоль типы данных переменных.





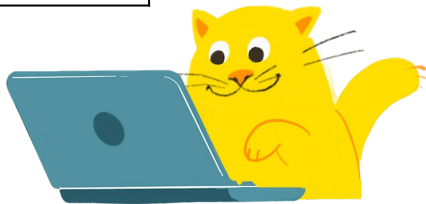
```
1 # ввод переменных
2 q = '123abc'
3 w = 4.35
4 e = [1, 3, 5, 7, 9]
5
6 # вывод данных
7 print('Тип переменной "q":', type(q))
8 print('Тип переменной "w":', type(w))
9 print('Тип переменной "e":', type(e))
10
11
```

```
07.08
Тип переменной "q": <class 'str'>
Тип переменной "w": <class 'float'>
Тип переменной "e": <class 'list'>
```



Закрепление знаний

1. Какую роль выполняет команда float в языке <i>Python</i> ?
2. Что такое логическая операция ?
3. Как можно передавать аргументы программе?
4. Какую роль выполняет функция boolean() в языке <i>Python</i> ?
5. Какое минимальное количество аргументов можно передать программе на языке <i>Python</i> за один раз?



Спасибо за внимание!

**Следующее занятие в
субботу в 12:00 (по Москве)**

