

ОПЕРАТОРЫ BREAK И CONTINUE

 Модуль 2 Занятие №1

Знакомство со строками



Цель занятия: Изучение понятия строк и применения строк на языке Python.



Глоссарий

- 1. Программа** – последовательность действия или операций, которая приводит к решению конкретно поставленной задачи.

- 2. Строка** – последовательность символов ASCII, отдельный тип данных, который входит в систему базовых типов языка.



1. Что такое **среда разработки**?

2. Для чего используется тип переменных **int** в языке Python?

3. Что такое команда **print()** и как ее можно применять в языке Python?

4. Как можно работать с несколькими операторами одновременно в языке Python?

5. По каким компонентам можно классифицировать персональные компьютеры?

Аргументы функции, применимые к спискам



```
main.py [Full Screen] [Dark Mode] [Run] Shell [Clear]
1 def func(a, b, c):
2     return a + b + c
3
4 print(func(1, 2, 3))
```

```
6
> |
```

Рисунок 1 – Синтаксис функции, которая работает с тремя аргументами

Аргументы функции, применимые к спискам



```
main.py [Full Screen] [Dark Mode] [Run] Shell [Clear]
1 def func(*args):
2     return args
3
4 print(func(1, 2, 3, 'abc'))
5 print(func())
6 print(func(1))
```

```
(1, 2, 3, 'abc')
()
(1,)
> |
```

Рисунок 2 – Синтаксис функции с использованием позиционных аргументов

Аргументы функции, применимые к спискам



```
main.py [Copy] [Refresh] [Run] Shell [Clear]
1 def func(**kwargs):
2     return kwargs
3
4 print(func(a=1, b=2, c=3))
5 print(func())
6 print(func(a='I love Python'))
```

```
{'a': 1, 'b': 2, 'c': 3}
{}
{'a': 'I love Python'}
> |
```

Рисунок 3 – Синтаксис функции с использованием именованных аргументов

Анонимные функции, инструкции lambda



```
main.py ⌂ 🌙 Run Shell Clear  
1 func = lambda x, y: x + y  
2  
3 print(func(1,2))  
4 print(func('a', 'b'))
```

3
ab
> |

Рисунок 4 – Синтаксис функции с использованием инструкции lambda

Анонимные функции, инструкции lambda



```
main.py ⏏ 🌙 Run Shell Clear  
1 func = lambda *args: args  
2  
3 print(func(1, 2, 3, 4))  
4 print(func(5, 6, 7, 8))  
5 print(func(9, 10, 11, 12))
```

```
(1, 2, 3, 4)  
(5, 6, 7, 8)  
(9, 10, 11, 12)  
> |
```

Рисунок 5 – Синтаксис функции lambda с использованием
позиционных аргументов

Анонимные функции, инструкции lambda



```
main.py ⌵ 🌙 Run Shell Clear  
1 double = lambda x: x * 2  
2  
3 print('Удвоенное значение числа 5: ', double(5))  
4 print('Удвоенное значение числа 25: ', double(25))  
5 print('Удвоенное значение числа 100: ', double(100))
```

Удвоенное значение числа 5: 10
Удвоенное значение числа 25: 50
Удвоенное значение числа 100: 200
> |

Рисунок 6 – Синтаксис функции-lambda, удваивающей вводимое значение

Анонимные функции, инструкции lambda



main.py				Run	Shell	Clear
1 <code>def double(x):</code> 2 <code> return x * 2</code> 3 4 <code>print(double(500))</code> 5 <code>print(double(1000))</code> 6 <code>print(double(2000))</code>					1000 2000 4000 >	

Рисунок 7 – Синтаксис функции, удваивающей вводимое значение в переменную x



Анонимные функции, инструкции lambda

1. Функция на рисунке 5 может иметь любое количество аргументов, но вычисляет и возвращает только одно значение;
2. Лямбда-функция применимы везде, где требуются объекты-функции;
3. Нужно помнить о том, что, синтаксически, лямбда-функция ограничена, она позволяет представить всего одно выражение;
4. Данные выражения имеют множество вариантов применения в конкретных областях программирования, наряду с другими типами выражений, используемых в функциях

Анонимные функции, инструкции lambda



```
main.py ⌂ ☾ Run Shell Clear  
1 def defined_cube(y):  
2     return y * y * y  
3  
4 lambda_cube = lambda y: y * y * y  
5  
6 print(defined_cube(2))  
7 print(lambda_cube(2))
```

```
8  
8  
> |
```

Рисунок 8 – Синтаксис функции, возвращающей заданное значение,
возведенное в третью степень

Анонимные функции, инструкции `lambda`



Отметим также основные особенности данных функций

- **Без использования лямбда-функции.** Здесь обе функции возвращают заданное значение, возведенное в куб, но при использовании ключевого слова **def**, приходится определить функцию с именем и **defined_cube()** и дать ей входную величину. После выполнения, разработчики также понадобятся вернуть результат из того места, откуда была вызвана функция (например, используя ключевое слово **def**)
- **С применением лямбды-функции.** Определение лямбды не включает оператор *return*, а всегда содержит возвращаемое выражение. Также можно поместить определение лямбды в любое место, где ожидается функция, а также не нужно присваивать его переменной. Так выглядят простые лямбда-функции

Анонимные функции, инструкции lambda



```
main.py ⌵ 🌙 Run Shell Clear  
1 my_list = [1, 2, 3, 4, 6, 10, 11, 15, 12, 14]  
2 new_list = list(filter(lambda x: (x%2 == 0), my_list))  
3 print('Четные числа: ', new_list)  
4 |
```

Четные числа: [2, 4, 6, 10, 12, 14]
> |

Рисунок 9 – Синтаксис функции *filter()*, которая отбирает четные числа из списка

Склеивание и разделение строк



```
pythonProject3
main.py
1
2 strr = 'cc' + 'dd'
3 print(strr)
4
5

Run: main
C:\Users\user\PycharmProjects\pythonProject3\ve
ccdd

Process finished with exit code 0
```

Рисунок 10 – Пример работы программы с склеиванием строк

Склеивание и разделение строк



```
l = ['q', 'w', 'e']  
print(''.join(l))
```

Рисунок 11 – Синтаксис метода join()

Склеивание и разделение строк



The screenshot shows an IDE window for a project named 'pythonProject3'. The main editor displays the following Python code in 'main.py':

```
1  
2 l = ['q', 'w', 'e']  
3 print(''.join(l))  
4  
5
```

Below the editor, the 'Run' console shows the output of the program:

```
Run: main x  
C:\Users\user\PycharmProjects\pythonProject3\ve  
qwe  
Process finished with exit code 0
```

Рисунок 12 – Пример работы программы с использованием
метода join()

Склеивание и разделение строк



```
str = 'q, w, e'  
print(str.split(sep=None))
```

Рисунок 13 – Синтаксис метода split()

Склеивание и разделение строк



The screenshot shows an IDE window for a Python project named 'pythonProject3'. The main editor displays the following code in 'main.py':

```
1  
2 str = 'q, w, e'  
3 print(str.split(sep=None))  
4  
5
```

Below the editor, the 'Run' console shows the execution output:

```
C:\Users\user\PycharmProjects\pythonProject3\ve  
['q,', 'w,', 'e']  
  
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 with 4 spaces and the Python version is 3.9.

Рисунок 14 – Пример работы программы с использованием
метода split()

Склеивание и разделение строк



```
pythonProject3
main.py
1
2 str = 'Python является ООП языком'
3 print(str[0:6])
4 print(str[7:26])
5
6

Run: main
C:\Users\user\PycharmProjects\pythonProject3\ve
Python
является ООП языком

Process finished with exit code 0
```

Рисунок 15 – Пример работы программы с использованием среза

Склеивание и разделение строк



```
pythonProject3
main.py
1
2 str = 'Python является ООП языком'
3 print(str[0:6])
4 print(str[7:])
5
6

Run: main
C:\Users\user\PycharmProjects\pythonProject3\ve
Python
является ООП языком

Process finished with exit code 0
```

Рисунок 16 – Пример работы программы с использованием среза и выводом до последнего элемента массива включительно

Склеивание и разделение строк



The screenshot shows an IDE window for a Python project named 'pythonProject3'. The main editor displays the following code in 'main.py':

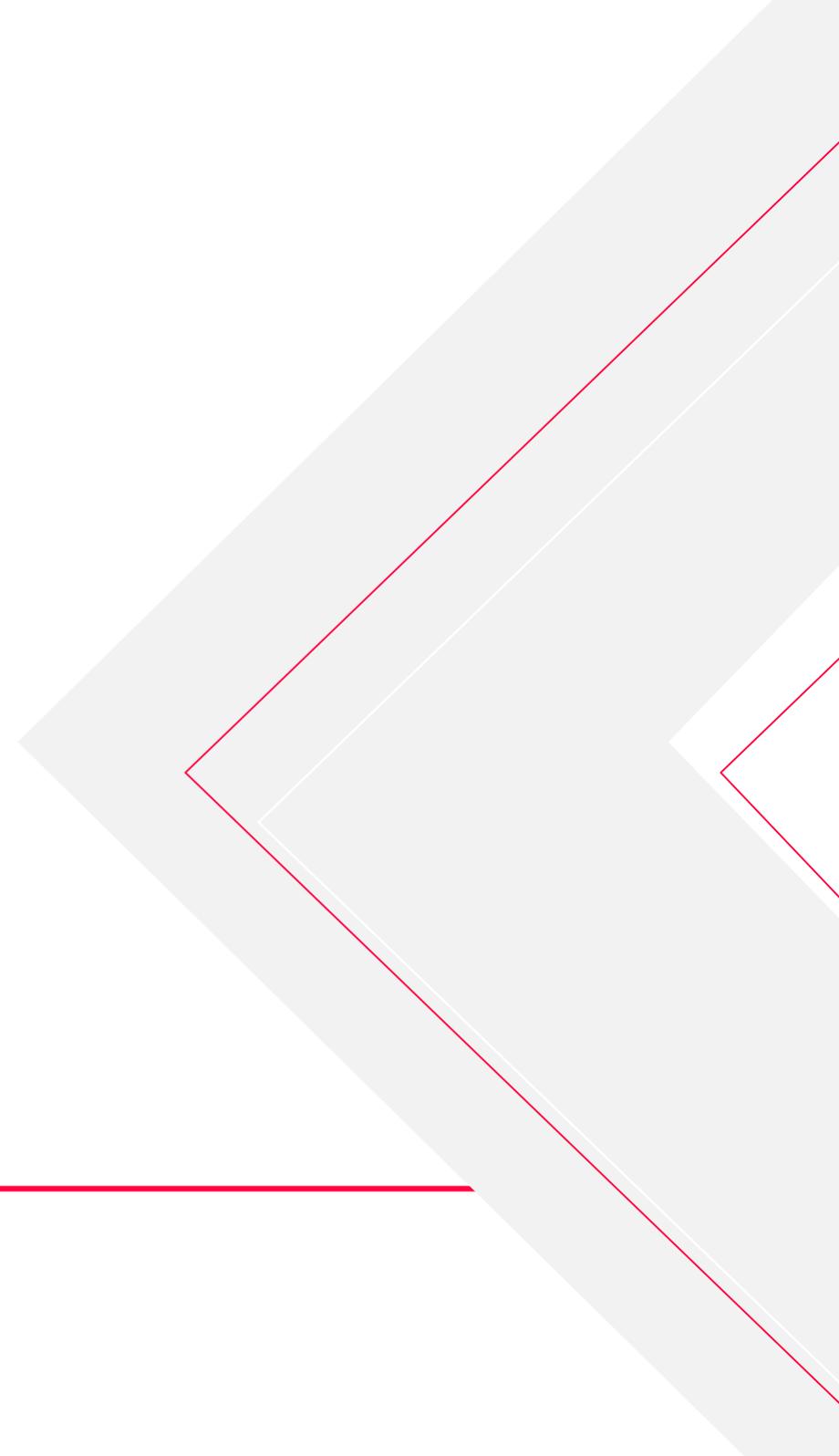
```
1  
2 str = '123456789'  
3 print(str[::-3])  
4  
5
```

Below the editor, the 'Run' console shows the execution output:

```
Run: main x  
C:\Users\user\PycharmProjects\pythonProject3\ve  
147  
Process finished with exit code 0
```

Рисунок 17 – Пример работы программы с особым типом среза строки

ПРАКТИЧЕСКИЕ ЗАДАЧИ





Задача 1

Дан следующий список чисел: **my_list** = [1, 4, 6, 10, 12, 15, 17, 19]. Используя функцию **filter()**, отобразить из данного списка только нечетные числа.



Решение

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

<pre>main.py</pre> <div style="text-align: right;">  Run</div>	<pre>Shell</pre> <div style="text-align: right;">Clear</div>
<pre>1 my_list = [1, 4, 6, 10, 12, 15, 17, 19] 2 new_list = list(filter(lambda x: (x%2 == 1), my_list)) 3 print('Нечетные числа: ', new_list) 4</pre>	<pre>Нечетные числа: [1, 15, 17, 19] > </pre>



Задача 2

Дан следующий список чисел: **'m'** = [2, 5, 8, 13, 16].

Используя функцию **len()**, посчитать количество элементов в списке.



Решение

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

The screenshot shows the PyCharm IDE interface. The main editor window displays the following Python code in `main.py`:

```
1  
2 m = [2, 5, 7, 13, 16]  
3 print('Количество символов: ', len(m))  
4  
5
```

Below the editor, the Run console shows the output of the code execution:

```
Run: main x  
C:\Users\user\PycharmProjects\pythonProject3\ve  
Количество символов: 5  
  
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 with 4 spaces and the Python version is 3.9.



Задача 3

Создать программу, в которой используются две строки: **'a'** = [Python is] и **'b'** = [OOP language]. Данные строки выводятся вместе в одной строке консоли



Решение

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

```
pythonProject3 - main.py
pythonProject3 main.py
1
2 a = ['Python is']
3 b = ['OOP language']
4 print(a + b)
5
6

Run: main x
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
['Python is', 'OOP language']

Process finished with exit code 0
```



Задача 4

Создать программу, в которой
используется список: **'с'** = [2, 5, 8, 10, 12].

Необходимо найти максимальный
элемент массива и вывести его на

КОНСОЛЬ



Решение

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

```
pythonProject3 - main.py  
pythonProject3 main.py  
1  
2 c = [2, 5, 8, 10, 12]  
3 print('Максимальный элемент массива: ', max(c))  
4  
5  
Run: main x  
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe  
Максимальный элемент массива: 12  
Process finished with exit code 0
```



1. Что такое **строка**?

2. Как можно использовать строки в языке Python?

3. Что такое **оператор присваивания** и как его можно использовать в языке Python?

4. Как можно работать с несколькими операторами одновременно в языке Python?

5. Для чего служит ключевое слово **if** в языке Python?