УНИВЕРСИТЕТ СИНЕРГИЯ

ОПЕРАТОРЫ BREAK И CONTINE

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 яPythonзыке?
5. По каким компонентам можно классифицировать персональные компьютеры?

Аргументы функции, применимые к спискам

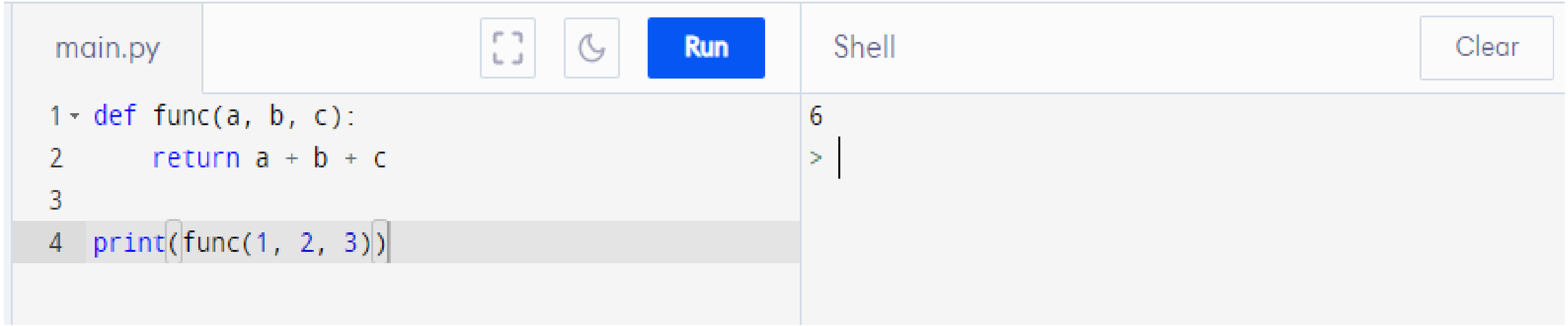


Рисунок 1 – Синтаксис функции, которая работает с тремя аргументами

Аргументы функции, применимые к спискам

main.py 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 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

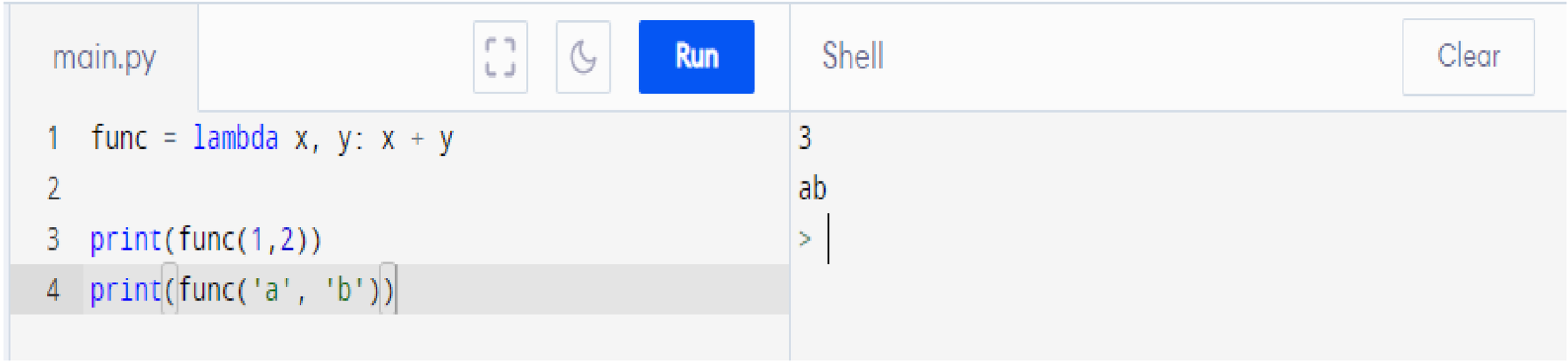


Рисунок 4 – Синтаксис функции с использованием инструкции lambda

Анонимные функции, инструкции lambda

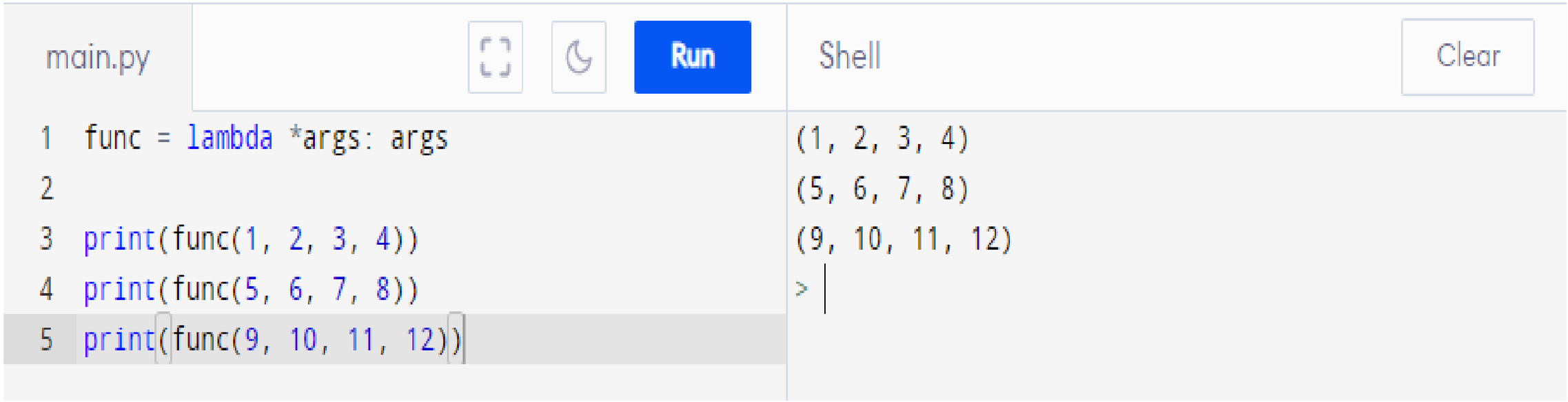


Рисунок 5 – Синтаксис функции lambda с использованием позиционных аргументов

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 def double(x):
2     return x * 2
3
4 print(double(500))
5 print(double(1000))
6 print(double(2000))
```

```
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 типа 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 понадобится возвратить 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

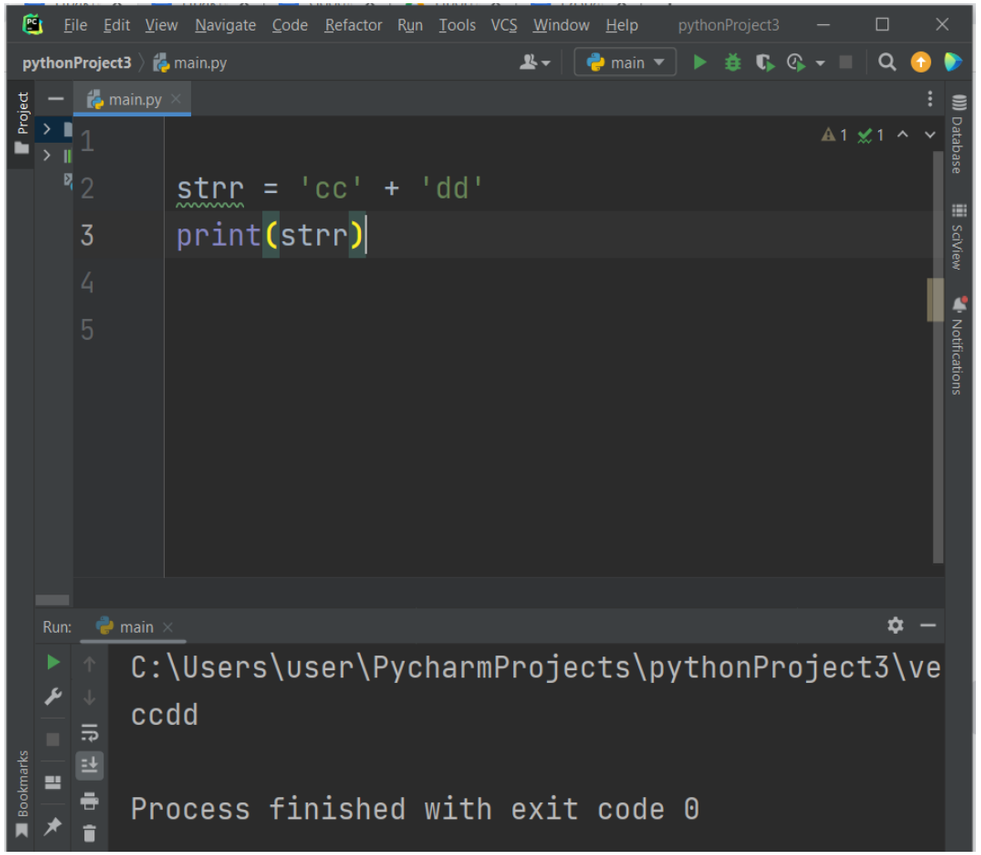


Рисунок 10 – Пример работы программы с склеиванием строк

Склеивание и разделение строк

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

Рисунок 11 – Синтаксис метода join()

Склеивание и разделение строк

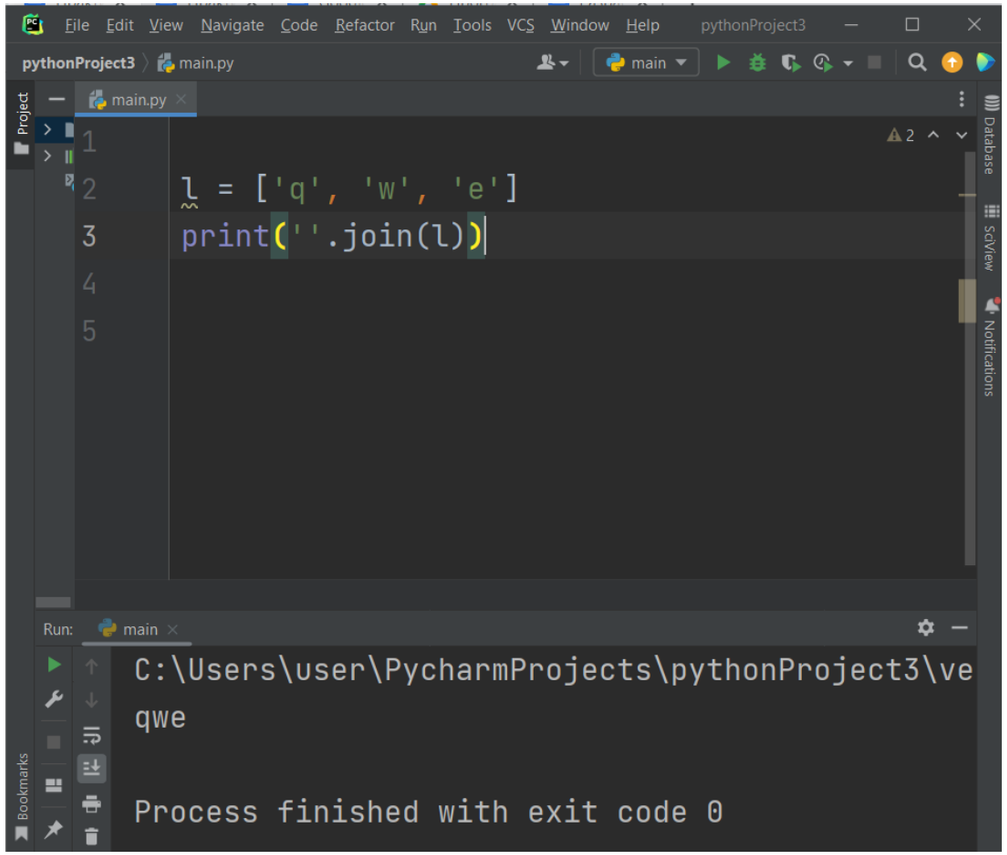


Рисунок 12 – Пример работы программы с использованием метода join()

Склеивание и разделение строк

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

Рисунок 13 – Синтаксис метода split()

Склеивание и разделение строк

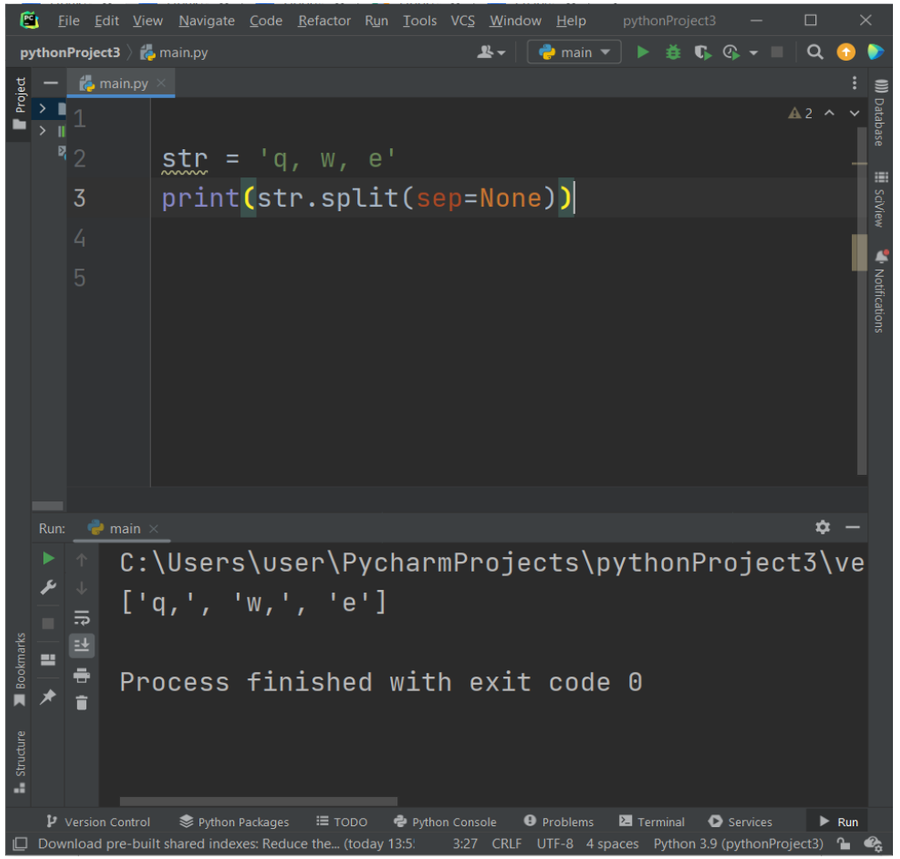


Рисунок 14 – Пример работы программы с использованием метода split()

Склеивание и разделение строк

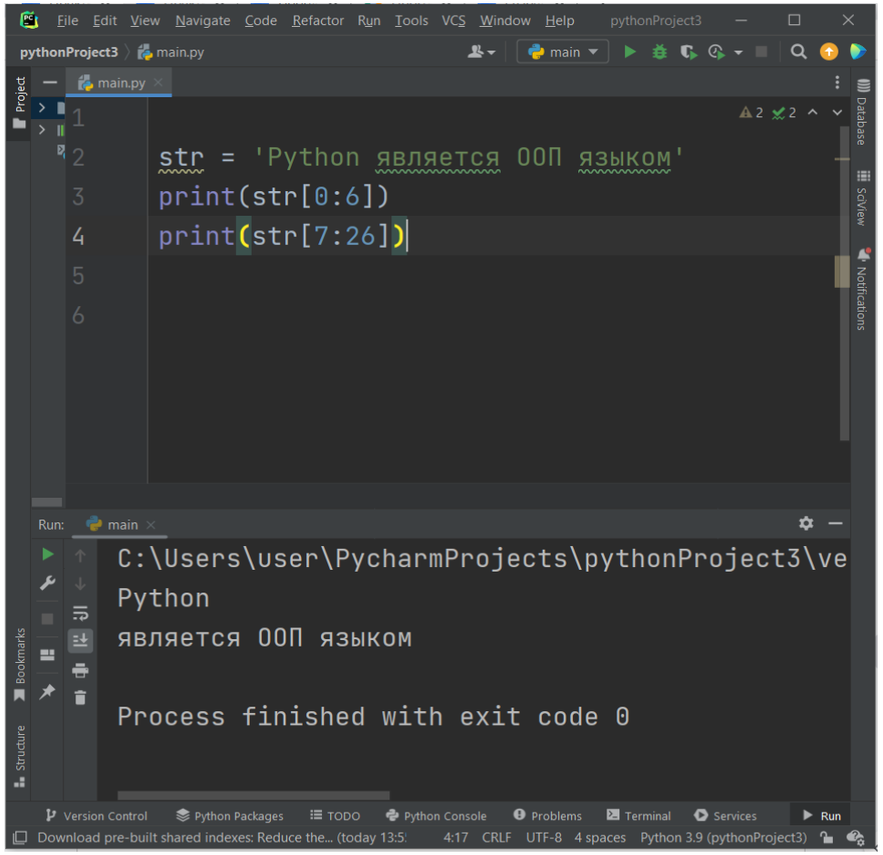


Рисунок 15 – Пример работы программы с использованием среза

Склеивание и разделение строк

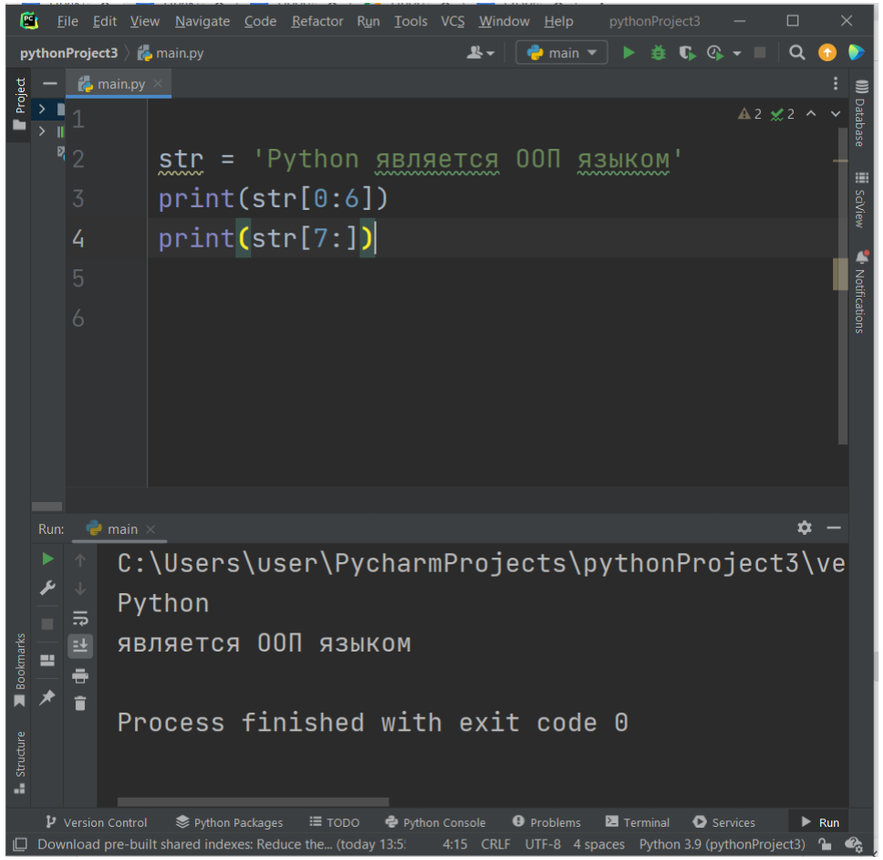


Рисунок 16 – Пример работы программы с использованием среза и выводом до последнего элемента массива включительно

Склеивание и разделение строк

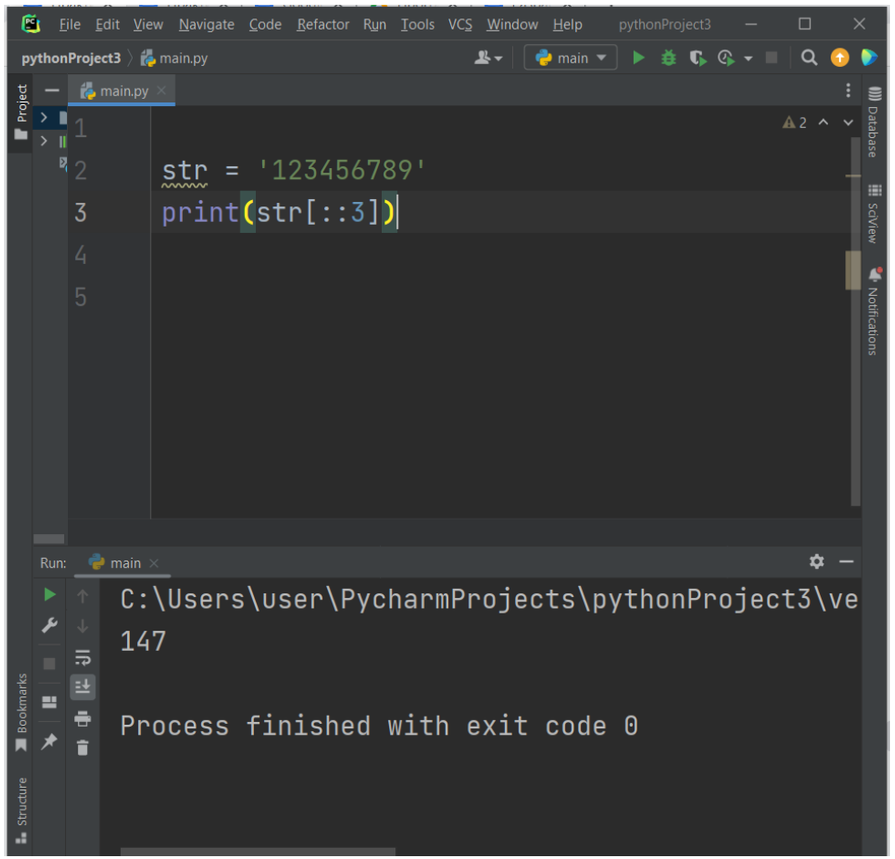


Рисунок 17 – Пример работы программы с особым типом среза строки

УНИВЕРСИТЕТ СИНЕРГИЯ

ПРАКТИЧЕСКИЕ ЗАДАЧИ

Задача 1

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

Решение

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

main.py

Run

```
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
```

Shell

Clear

```
Нечетные числа:  [1, 15, 17, 19]
>
```

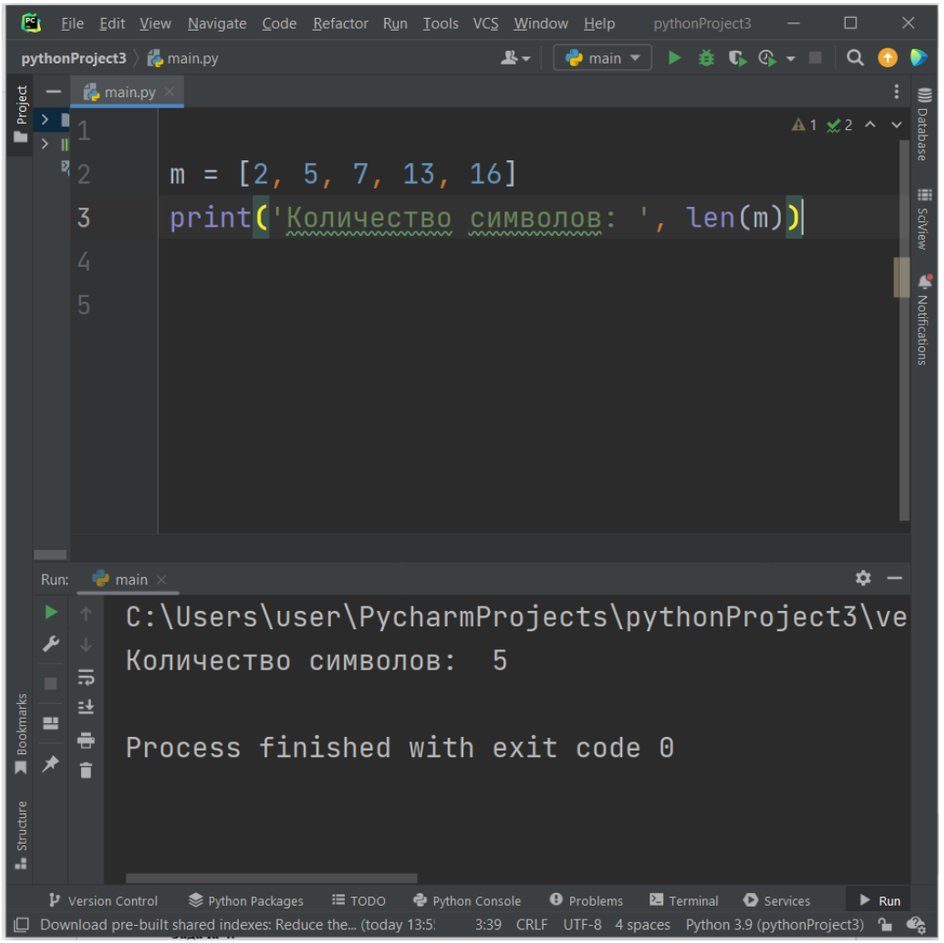
Задача 2

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:

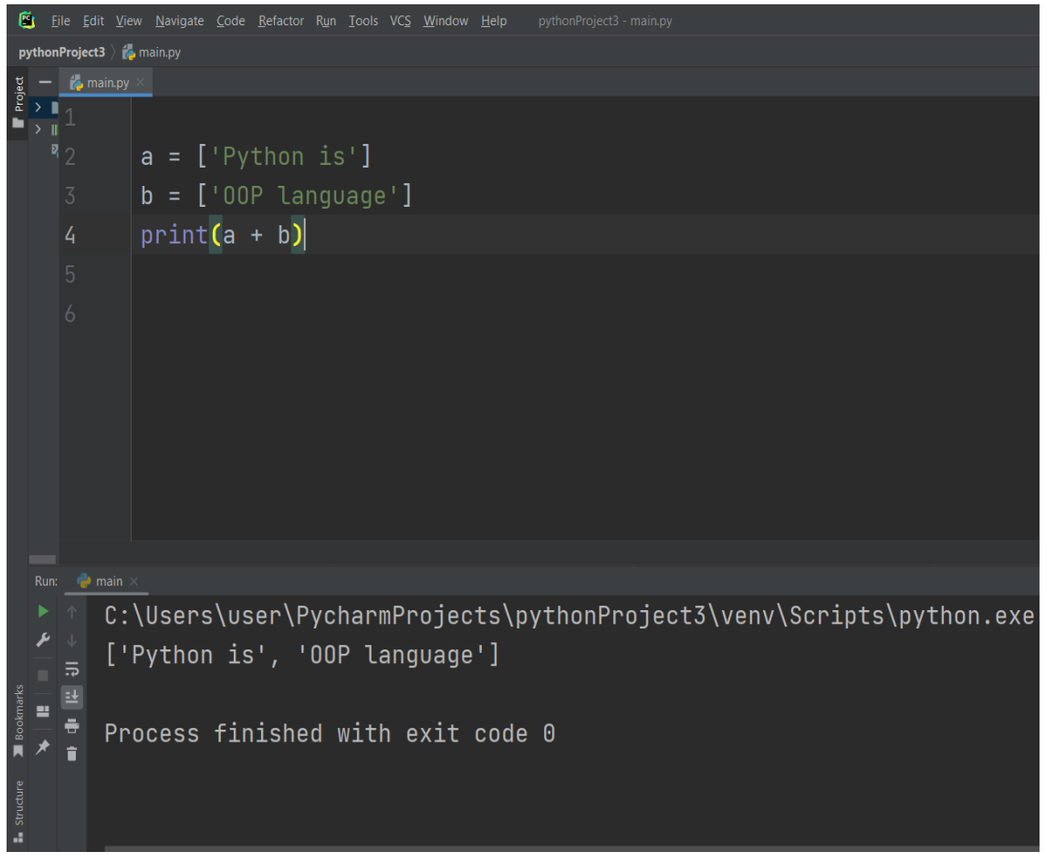


Задача 3

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:

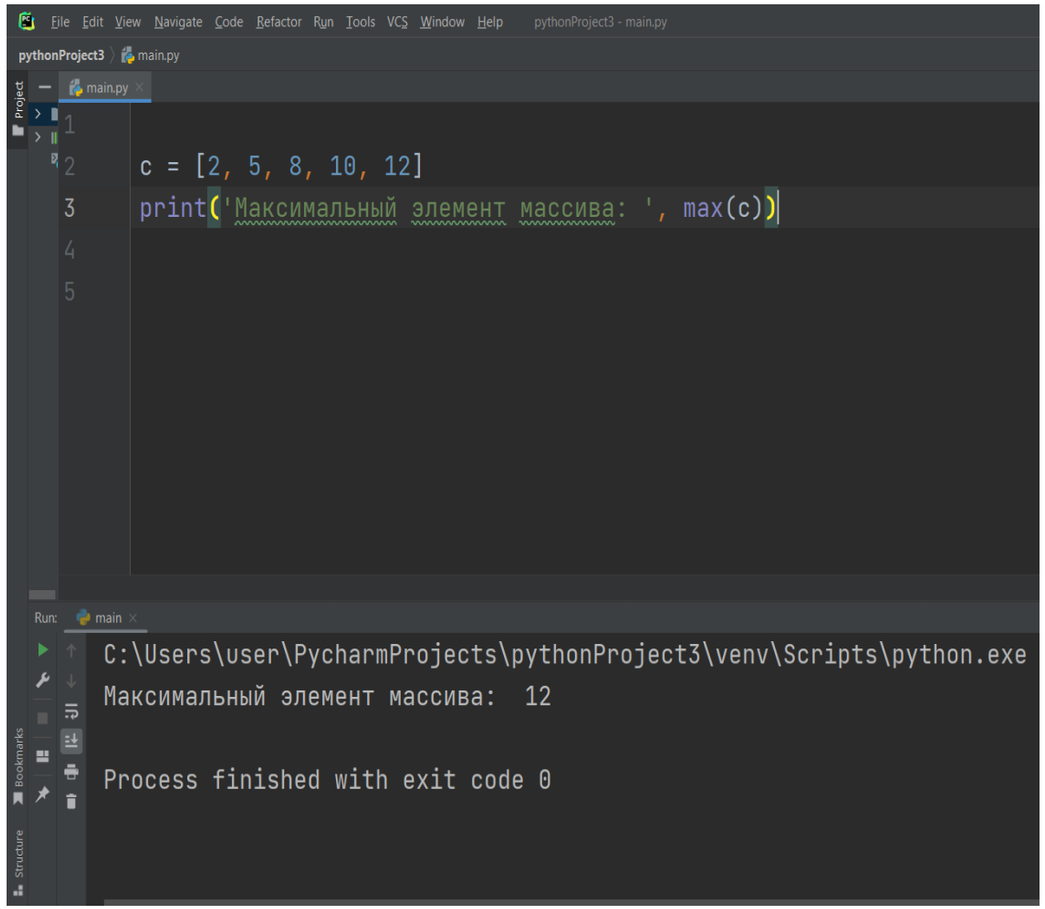


Задача 4

Создать программу, в которой используется список: **‘c’** = [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:



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?