УНИВЕРСИТЕТ СИНЕРГИЯ

ПОНЯТИЕ ЦИКЛА. ВИДЫ ЦИКЛОВ

Модуль 1. Занятие 8.

Тема и цель

Тема:

Знакомство с циклами.
Различия циклов.

Цель занятия:

Изучение основ по работе
с циклами на языке Python.

Глоссарий

1. **Цикл** - разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.
2. **Программа** - данные, которые используются процессором как инструкции по управлению компьютерной системой.

Подтемы

1. Какую роль выполняет функция **lambda** в языке *Python*?
2. Что такое **функция**?
3. Как можно передавать аргументы функции?
4. Какую роль выполняет функция **filter()** в языке *Python*?
5. Какое максимальное количество аргументов можно передать функции на языке *Python* за один раз?

ТИПЫ
ПЕРЕМЕННЫХ

Типы переменных

Переменная хранит данные одного из типов данных:

- **int** (integer) – число
- **float** (плавающая точка) – дробное число
- **str** (string) – строка
- **bool** (булева функция) – True или False (правда или ложь / из двоичной логики)

Преобразование типа данных

Для изменения типа данных используется следующая конструкция:
Python поддерживает все распространенные арифметические операции:

- `+` сложение
- `-` вычитание
- `=` присваивание
- `*` умножение
- `/` деление
- `**` возведение в степень
- `//` целочисленное деление
- `%` остаток от деления

Преобразование типа данных

Оператор	Описание	Пример
==	Возвращает True, если оба операнда равны	a == b
!=	Возвращает True, если оба операнда не равны	a != b
>	Возвращает True, если первый операнд больше второго	a > b
<	Возвращает True, если первый операнд меньше второго	a < b
>=	Возвращает True, если первый операнд больше или равен второму	a >= b
<=	Возвращает True, если первый операнд меньше или равен второму	a <= b

Преобразование типа данных

Для создания составных условных выражений применяются логические операции.
В Python имеются следующие логические операторы:

and (логическое умножение)

or (логическое сложение)

not (логическое отрицание)

Оператор in

Преобразование типа данных

Отладкой кода называют процесс последовательного приведения кода в рабочее состояние после нахождения ошибки. Существуют различные методы отладки:

- **Debug**
- **Проверки выводом**

Циклы

- for
- while

Цикл while

Конструкция цикла while

```
1  while условное_выражение:
2      инструкции
```

Цикл while

Пример работы цикла while

```
1  number = 1
2
3  while number < 5:
4      print(f"number = {number}")
5      number += 1
6  print("Работа программы завершена")
```

Цикл while

Конструкции цикла

```
1 print(f"number = {number}")
2 number += 1
```

Процесс цикла

Процесс цикла можно представить следующим образом:

1. Сначала проверяется значение переменной **number** - больше ли оно 5.
   И поскольку вначале переменная равна 1, то это условие возвращает True,
   и поэтому выполняются инструкции цикла

2. Инструкции цикла выводят на консоль строку number = 1. И далее значение
   переменной number увеличивается на единицу - теперь она равна 2.
   Однократное выполнение блока инструкций цикла называется **итерацией**.
   То есть таким образом, в цикле выполняется первая **итерация**

3. Снова проверяется условие number < 5. Оно по прежнему равно True,
   так как number = 2, поэтому выполняются инструкции цикла

Процесс цикла

4. Инструкции цикла выводят на консоль строку number = 2. И далее значение переменной number опять увеличивается на единицу - теперь она равна 3. Таким образом, выполняется вторая итерация.

5. Опять проверяется условие number < 5. Оно по прежнему равно True, так как number = 3, поэтому выполняются инструкции цикла

6. Инструкции цикла выводят на консоль строку number = 3. И далее значение переменной number опять увеличивается на единицу - теперь она равна 4. То есть выполняется третья итерация.

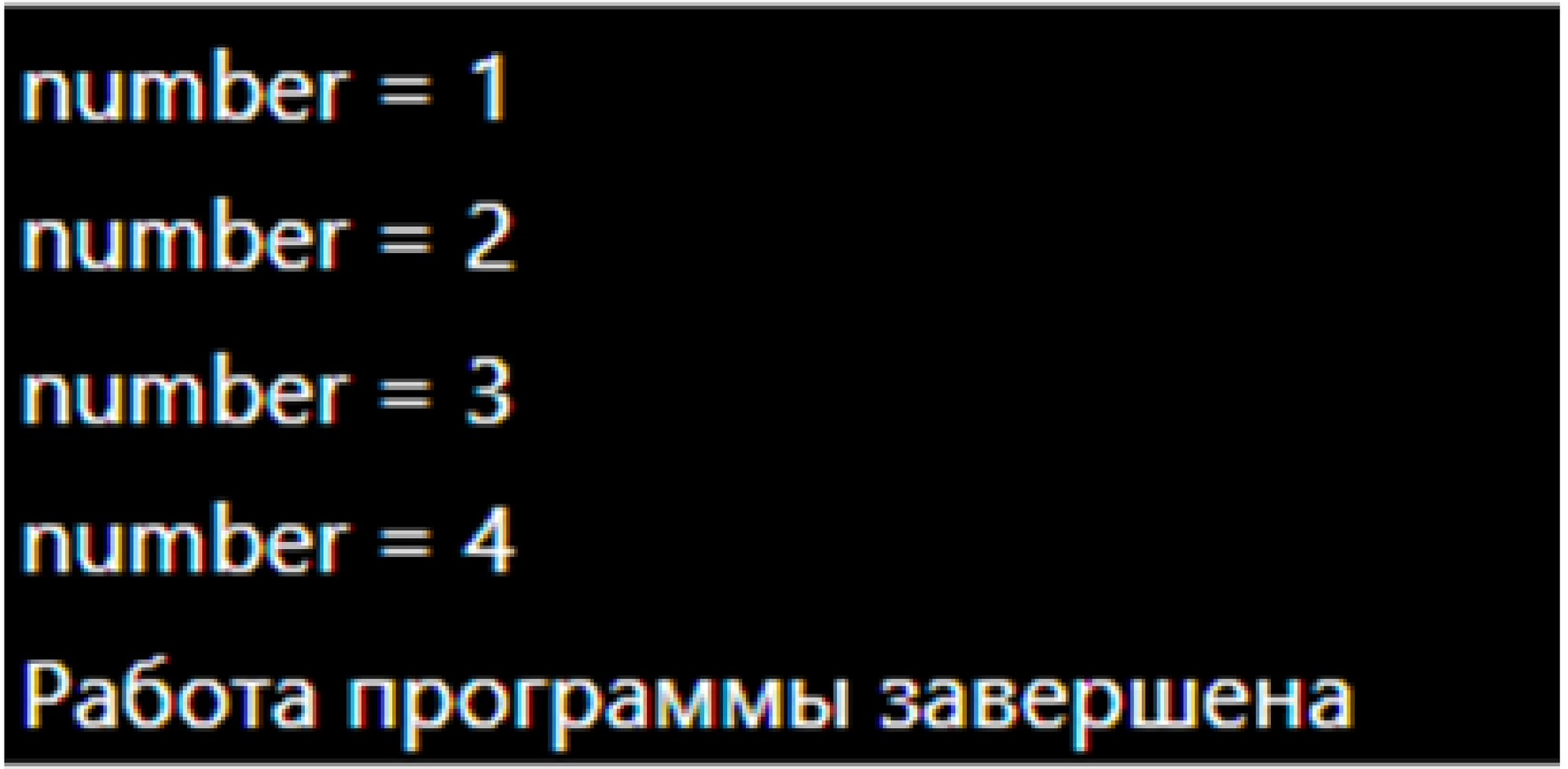
Процесс цикла

7. Снова проверяется условие number < 5. Оно по прежнему равно True, так как number = 4, поэтому выполняются инструкции цикла

8. Инструкции цикла выводят на консоль строку number = 4. И далее значение переменной number опять увеличивается на единицу - теперь она равна 5. То есть выполняется четвертая итерация.

9. И вновь проверяется условие number < 5. Но теперь оно равно False, так как number = 5, поэтому выполняются выход из цикла. Все цикл - завершился. Дальше уже выполняются действия, которые определены после цикла. Таким образом, данный цикл произведет четыре прохода или четыре итерации

Консольный вывод при использовании цикла while



Пример программы с циклом while

```
1  number = 1
2
3  while number < 5:
4      print(f"number = {number}")
5      number += 1
6  else:
7      print(f"number = {number}. Работа цикла завершена")
8  print("Работа программы завершена")
```

Консольный вывод при работе с циклом while

```
number = 1
number = 2
number = 3
number = 4
number =5. Работа цикла завершена
Работа программы завершена
```

Пример программы
с блоком else

```
1  number = 10
2
3  while number < 5:
4      print(f"number = {number}")
5      number += 1
6  else:
7      print(f"number = {number}. Работа цикла завершена")
8  print("Работа программы завершена")
```

Цикл for

Конструкция
цикла for

```
1 for переменная in набор_значений:
2     инструкции
```

Пример программы с циклом for

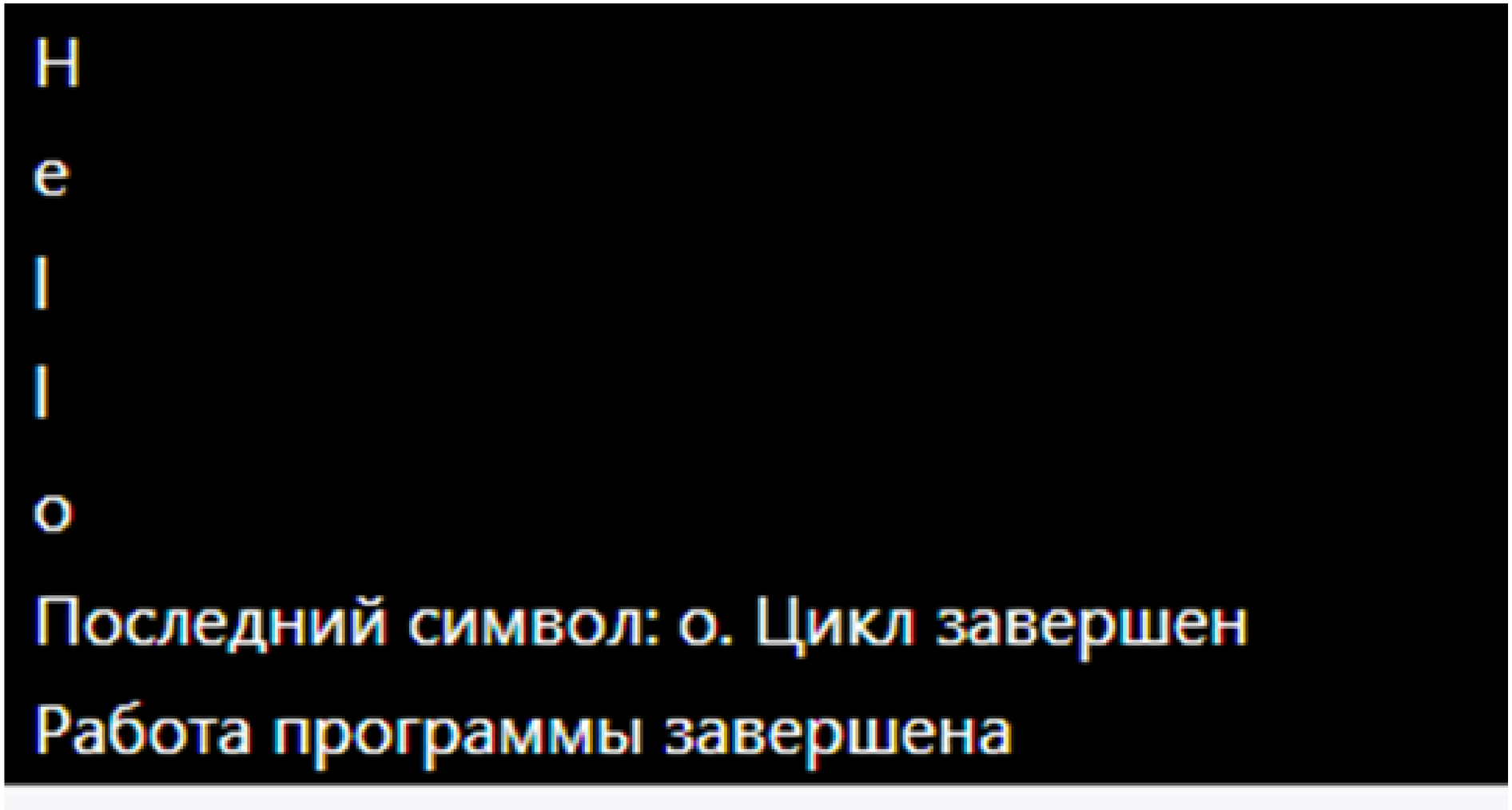
```
1  message = "Hello"
2
3  for c in message:
4      print(c)
```

Консольный вывод при работе с циклом for

Пример программы с использованием цикла for и блока else

```
message = "Hello"
for c in message:
    print(c)
else:
    print(f"Последний символ: {c}. Цикл завершен");
print("Работа программы завершена")  # инструкция не имеет отступа, поэтому не относится к else
```

Консольный вывод при работе с циклом for и блоком else

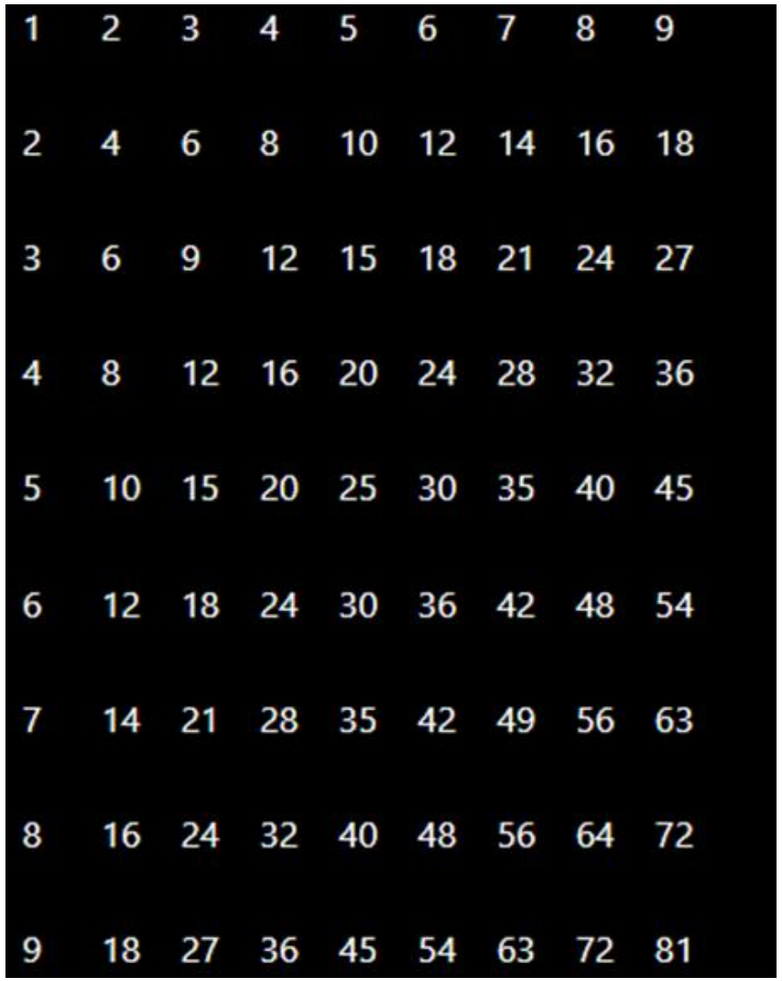


ВЛОЖЕННЫЕ ЦИКЛЫ

Пример программы
с вложенными циклами

```
1  i = 1
2  j = 1
3  while i < 10:
4      while j < 10:
5          print(i * j, end="\t")
6          j += 1
7      print("\n")
8      j = 1
9      i += 1
```

Консольный вывод при работе
с вложенными циклами



Пример программы
с вложенными циклами

```
1 for c1 in  "ab":
2     for c2 in "ba":
3         print(f"{c1}{c2}")
```

Консольный вывод при работе
с вложенными циклами

ВЫХОД ИЗ ЦИКЛА.
BREAK И CONTINUE

Пример программы с использованием оператора break

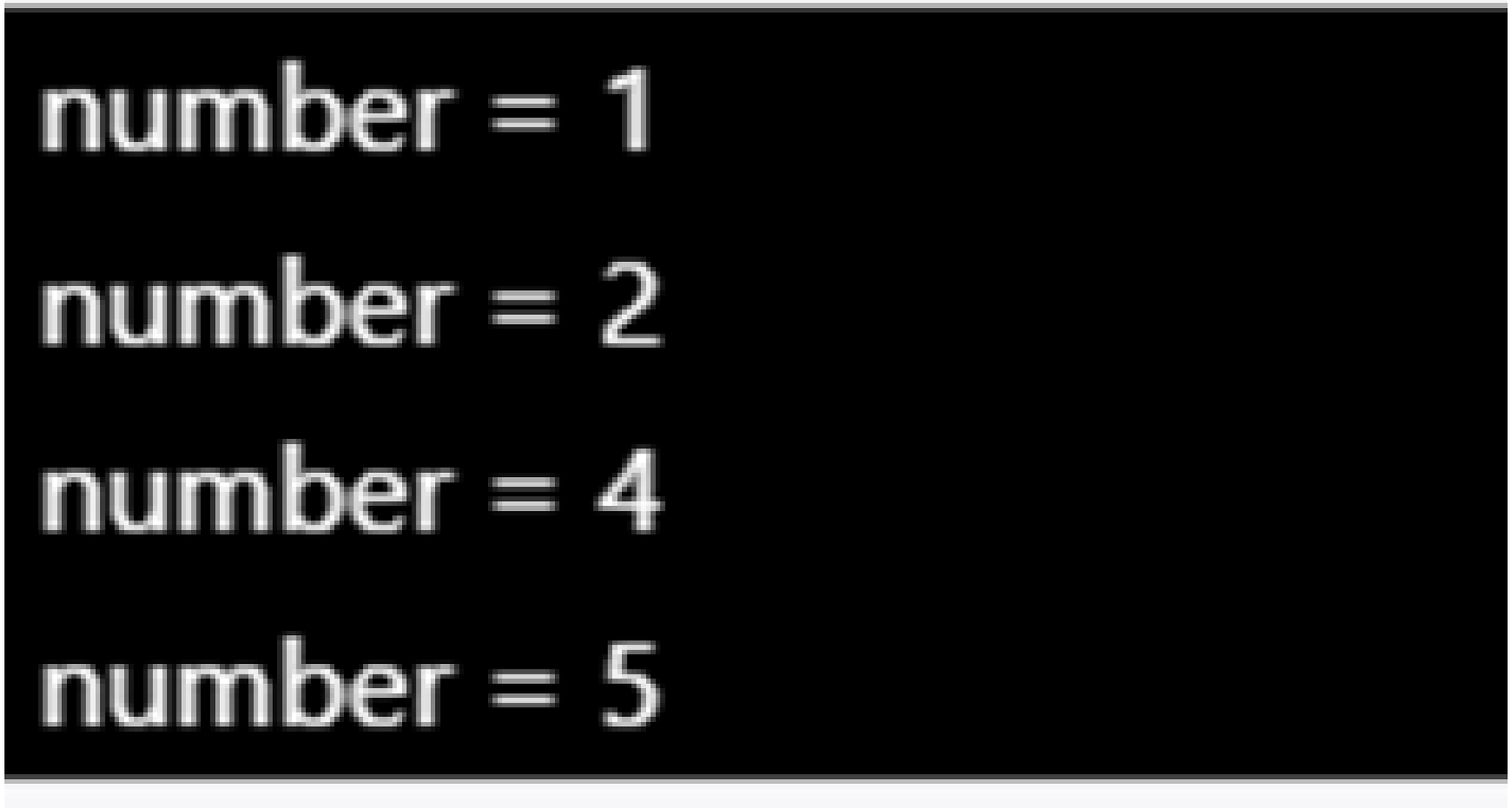
```
number = 0
while number < 5:
    number += 1
    if number == 3 :    # если number = 3, выходим из цикла
        break
    print(f"number = {number}")
```

Консольный вывод при работе с оператором break

Пример программы с использованием оператора continue

```
number = 0
while number < 5:
    number += 1
    if number == 3 :     # если number = 3, переходим к новой итерации цикла
        continue
    print(f"number = {number}")
```

Консольный вывод с использованием оператора continue



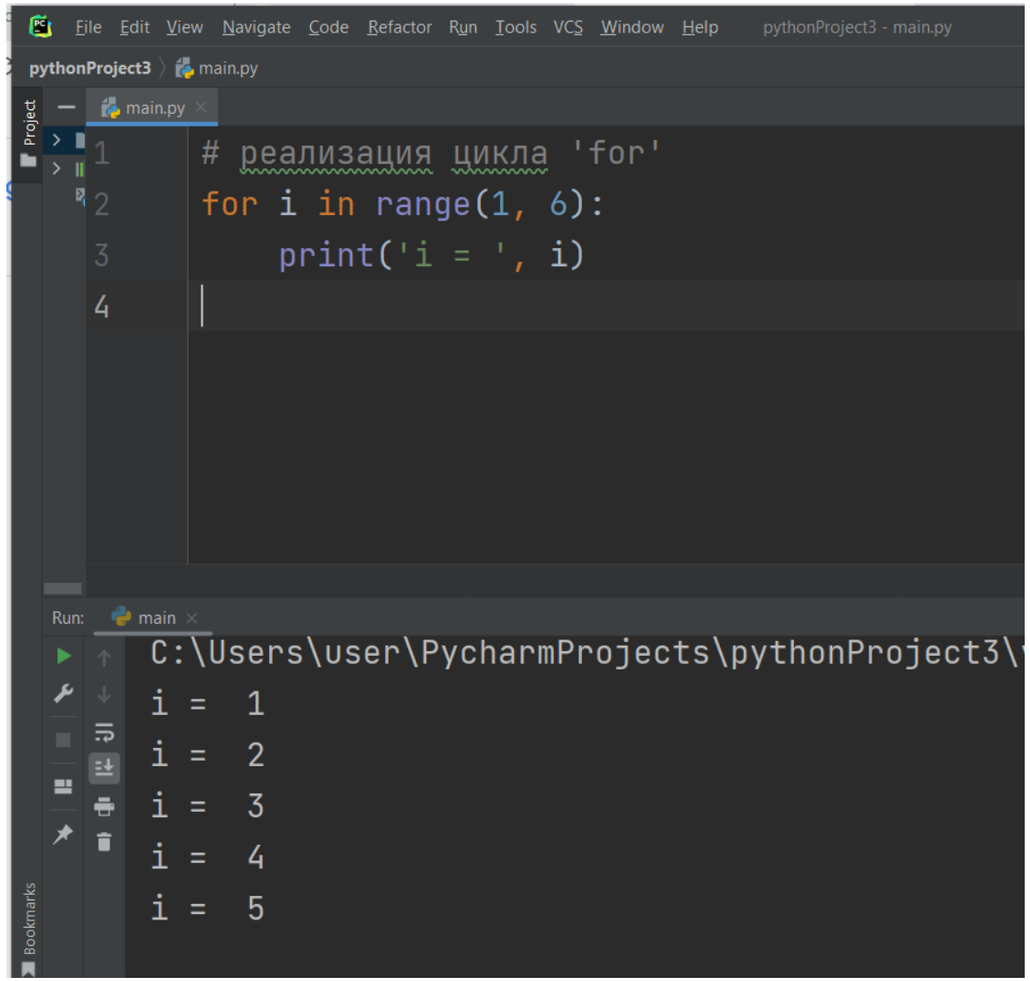
ПРАКТИЧЕСКИЕ ЗАДАЧИ

Задача 1

Создать программу с использованием цикла **for**. Данный цикл перебирает значения переменной **‘i’** от 1 до 5 и выводит на консоль.

Решение

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

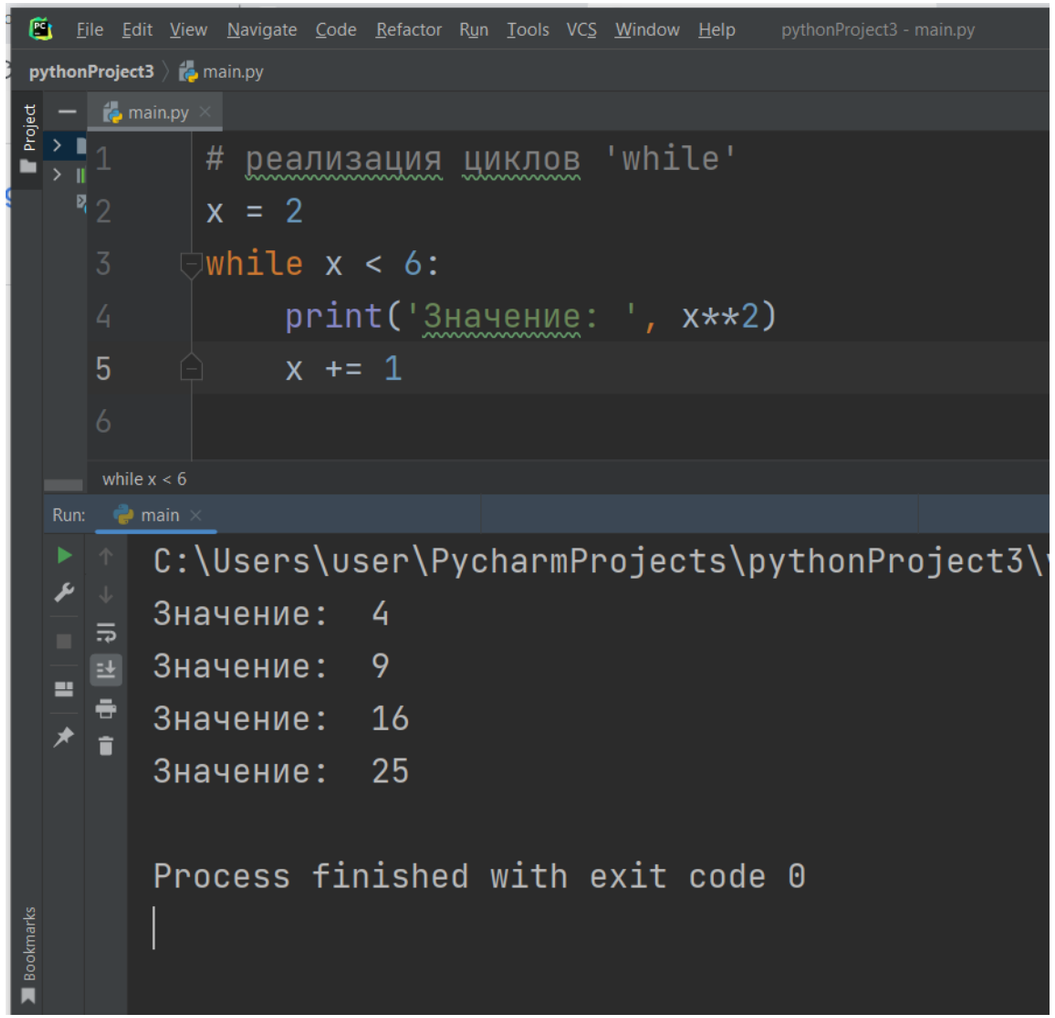


Задача 2

Создать программу с использованием цикла **while** и переменной **‘x’**, значение которой, изначально, равно 2. Данный цикл выводит значения переменной **‘x’** во второй степени до тех пор, пока значение переменной **‘x’** не станет равным 6. При каждой итерации значение переменной **‘x’** увеличивается на единицу.

Решение

Для решения данной задачи понадобится следующий код:

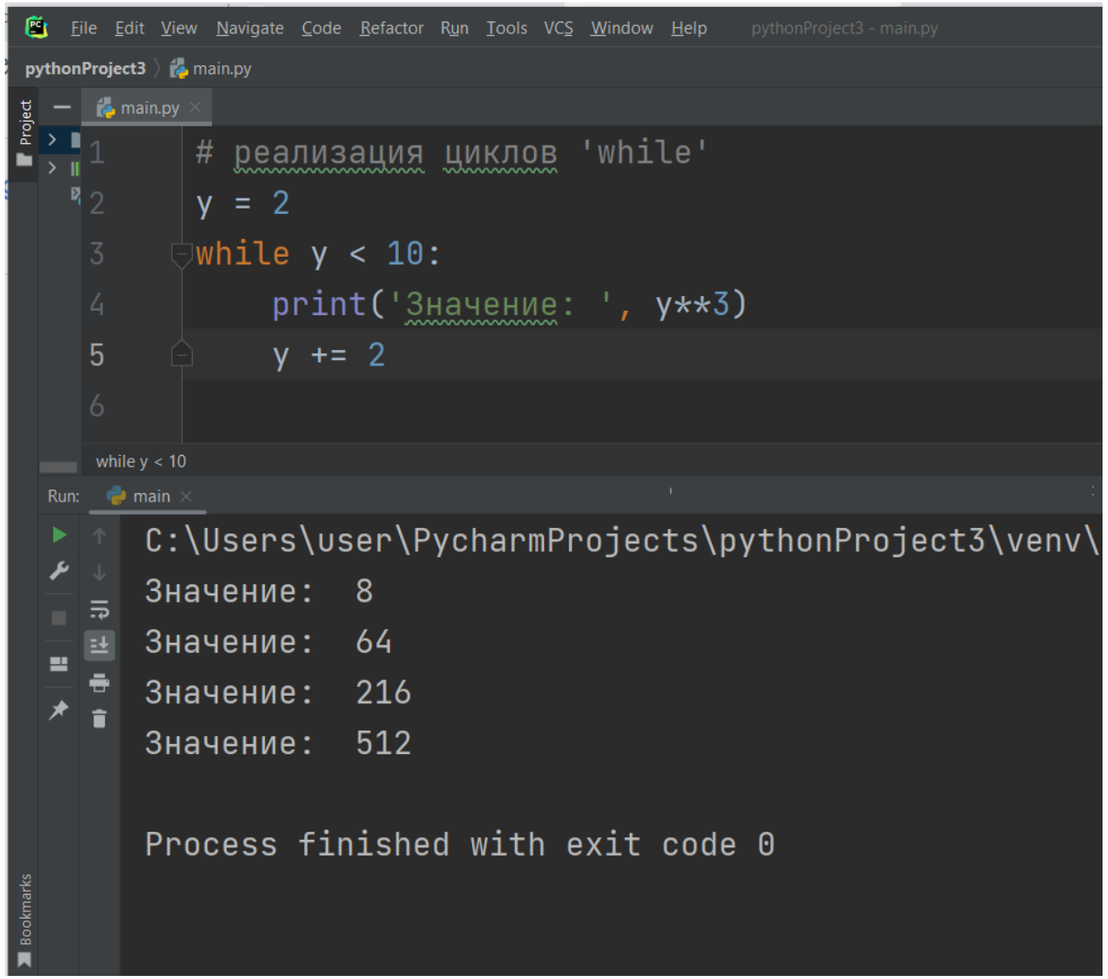


Задача 3

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

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

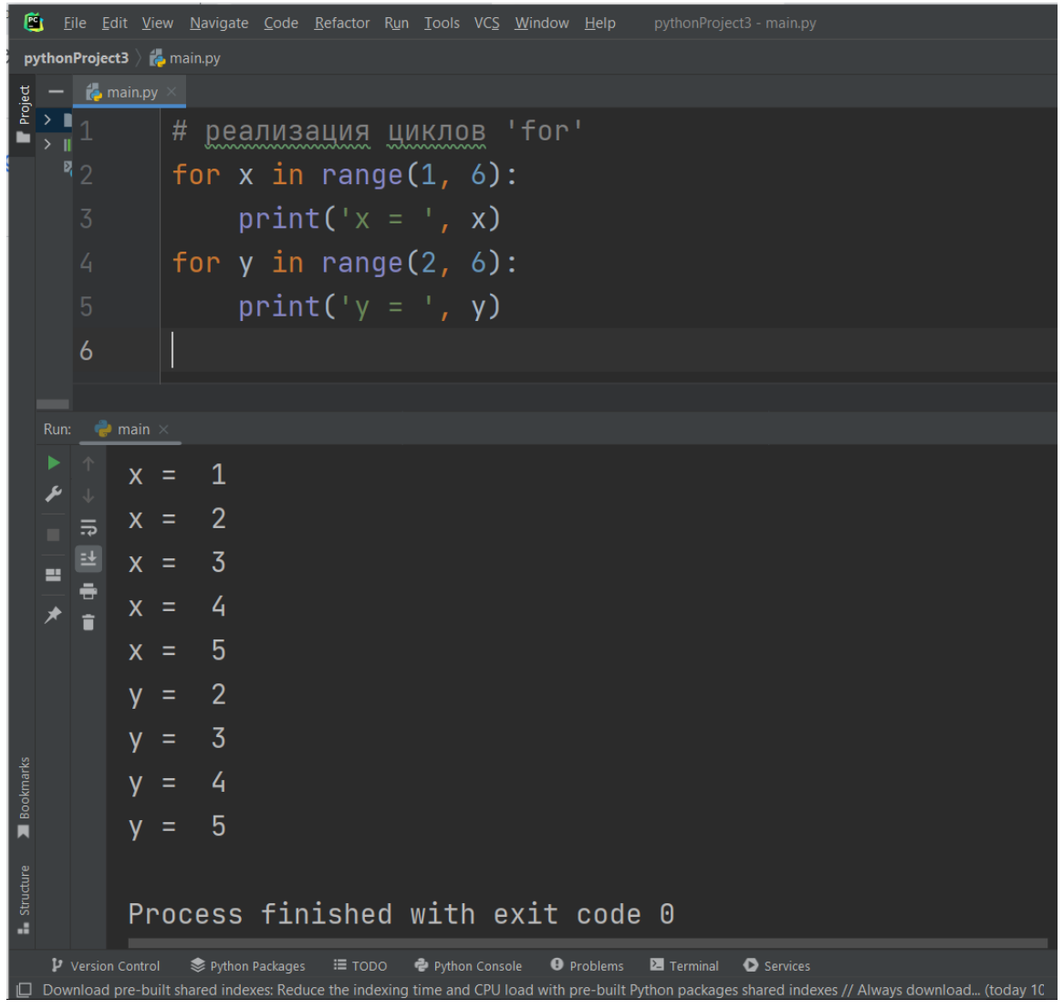


Задача 4

Создать программу с использованием цикла **for**. Данный цикл перебирает значения переменной **‘x’** от 1 до 5 и переменной **‘y’** от 2 до 5 и выводит соответствующие значения на консоль.

Решение

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



Вопросы

1. Какую роль выполняет цикл **for** в языке *Python*?
2. Что такое **цикл**?
3. Как можно передавать аргументы циклам?
4. Какую роль выполняет функция **len(x)** в языке *Python*?
5. Какое максимальное количество аргументов можно передать циклу на языке *Python* за один раз?