

## ИГРОВЫЕ ЦИКЛЫ

На этом уроке вы узнаете о том, как создавать собственные циклы, сначала определяя повторяющиеся шаги программы, а затем перенося в цикл нужного типа.

С помощью цикла **for** вы сможете повторять выполнение участка кода заданное количество раз.

С помощью цикла **while** вы сможете повторять выполнение кода до тех пор, пока какое-либо условие не станет истинным, или до наступления определенного события.

К концу урока в вышей папке должны находиться работающие программы:

- 1) **names.py**
- 2) **while\_1.py**
- 3) **while\_2.py**
- 4) **numbers.py**
- 5) **guess\_nuymber.py**
- 6) **\*guess\_nuymber\_2.py**
- 7) **colors.py**
- 8) **name10.py**
- 9) **circle\_6.py**
- 10) **circle\_rand\_1.py**
- 11) **circle\_rand\_2.py**
- 12) **circle\_rand\_3.py**
- 13) **circle\_rand\_4.py**

## Игровые циклы и цикл `while`

На предыдущих уроках вы научились использовать в программах на языке Python цикл `for`. Это мощное средство для выполнения повторяющихся кодов программы. Однако, если мы захотим, чтобы выполнение цикла прекратилось, если произойдет какое-либо событие, вместо прохождения по всему длинному списку чисел? Или, что если мы не уверены в том, сколько раз следует запускать цикл?

Например, представьте себе *игровой цикл*, когда нам необходимо написать программу игры, где пользователь сам выбирает, нужно ли продолжить играть или выйти из программы.

Одним из способов решения проблемы игрового цикла является использование циклов другого рода, а именно цикла `while`. В отличие от цикла `for` ("Для" - в переводе с англ.), цикл `while` ("Пока" - в переводе с англ.) не производит циклическое повторение по заранее установленному списку значений. Цикл `while` может проверять *условие*, или ситуацию, и решать, выполнить ли еще один проход цикла, либо прекратить выполнение.

Формат оператора цикла `while` выглядит следующим образом:

```
while условие :  
    Инструкции  
    тела цикла
```

### Задание 1 (names.py)

Давайте испробуем программу с циклом `while`, чтобы увидеть его в действии.

1. Создай файл с именем `names.py` в своей папке. Введи следующий код:

```
# Спросить у пользователя его имя  
name = input("Как тебя зовут? ")  
# Печатать имена, пока мы не захотим выйти  
while name != "":  
    # Напечатать имя 50 раз  
    for k in range(50):  
        # Печатать имена через пробел, а не с новой строки  
        print(name, end = " ")  
        print() # После цикла for пропустить строку  
    # Запросить еще одно имя или выйти  
    name = input("Введите еще имя или нажмите Enter для выхода: ")  
print("Спасибо за игру!")
```

2. Запусти скрипт из окна редактора.

3. Введи несколько имен, а затем для выхода из программы нажми клавишу Enter.



## Счётчик цикла

В большинстве случаев использования цикла **while** мы должны сначала объявить переменную, которая будет выполнять роль счетчика циклов. Назовем эту переменную **counter** (счетчик).

Условие в операторе **while** может быть истиной, если значение счетчика будет меньше (или больше) некоторого определенного значения. Если это так, то цикл будет продолжаться.

Для этого случая цикл **while** имеет следующий формат:

```
counter = начальное значение
while условие :
    блок инструкций
    приращение counter
```

Последовательность работы цикла **while**:

- 1) переменной-счетчику присваивается начальное значение;
- 2) проверяется условие, если оно истинно, то выполняются инструкции внутри цикла, иначе выполнение цикла завершается;
- 3) переменная-счетчик изменяется на величину, указанную в параметре приращение;
- 4) переход к пункту 2).

### Задание 3 (while\_1.py)

1. Создай файл с именем **while\_1.py** в своей папке.
2. Рассмотрим программу, которая выводит на экран числа от 1 до 10.
3. Введи следующий код:

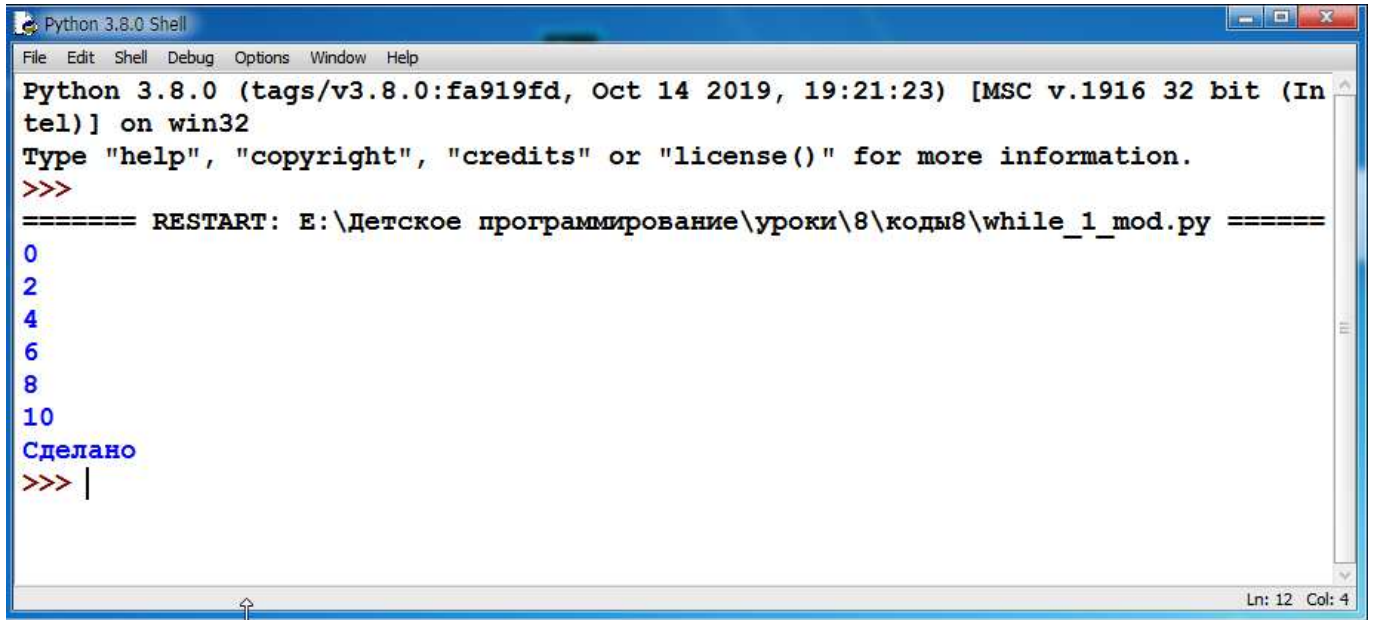
```
counter = 1                # Начальное значение
while counter <= 10 :      # Условие
    print(counter)         # Инструкции
    counter = counter + 1  # Приращение
print("Сделано")
```

Запусти скрипт из окна редактора. Убедись в правильности его работы.

#### Задание 4 (самостоятельно)

Открой скрипт в файле с именем **while\_1.py** в своей папке.  
Измени программу:

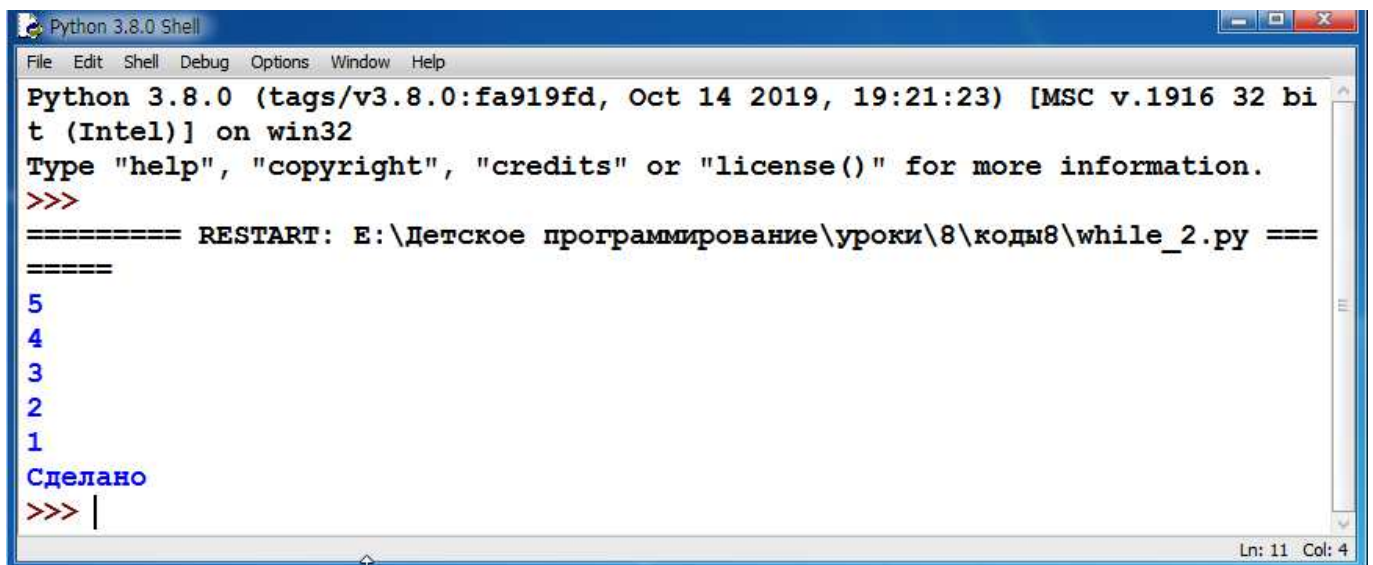
- пусть на экран выводятся все чётные числа от 0 до 10 включительно.



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\Детское программирование\уроки\8\коды8\while_1_mod.py =====
0
2
4
6
8
10
Сделано
>>> |
```

#### Задание 5 (while\_2.py ) (самостоятельно)

1. Создай файл с именем **while\_2.py** в своей папке.
2. Напиши программу, которая выводит на экран числа от 5 до 1 включительно.
3. Запусти скрипт из окна редактора.
4. Убедись в правильности его работы.



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\Детское программирование\уроки\8\коды8\while_2.py =====
5
4
3
2
1
Сделано
>>> |
```

## Случайное веселье и игры на удачу!

До сих пор мы программировали компьютер на принятие решений на основании условий. Сейчас же мы будем программировать компьютер на выбор случайного числа в диапазоне от 1 до 100, бросать игральные кубики, играть в "Камень, ножницы, бумага".

Все эти игры объединяет одна общая идея – *случайность*. Мы хотим, чтобы компьютер выбирал случайное число в диапазоне от 1 до 100, а мы будем угадывать это число. Мы хотим, чтобы компьютер случайным образом выбирал камень, бумагу или ножницы, а затем мы будем выбирать, чем играть, и смотреть, кто победил. Все эти примеры плюс игра с кубиками называются *азартными играми*. Когда мы бросаем 5 раз кубик, мы каждый раз получаем отличный от предыдущего результат. Именно этот элемент случайности делает такие игры интересными.

Можно запрограммировать компьютер на случайное поведение. В языке Python есть особый модуль под названием **random**, позволяющий нам сделать случайный выбор. Мы будем использовать этот модуль для программирования азартных игр и рисования на экране случайных фигур.

### Получение случайных чисел

Сначала нам необходимо импортировать модуль **random** с помощью команды **import random**. В модуле реализовано несколько разных функций для генерации случайных данных.

Мы будем использовать функцию **randint()** для генерации *случайного целого числа*. Функция **randint()** ожидает от нас получения двух аргументов, которые мы будем указывать между скобок после имени функции: наименьшее и наибольшее числа диапазона. Указание в скобках наименьшего и наибольшего числа сообщит функции **randint()** границы диапазона, из которого мы хотим выбирать случайные числа.

Рассмотрим примеры получения случайных чисел:

```
import random
# Получение случайного целого числа в диапазоне от 1 до 10 включительно
number = random.randint(1, 10)
# Получение случайного целого числа в диапазоне от 1 до 100 включительно
number = random.randint(1, 100)
```

## Задание 6 (numbers.py)

Рассмотрим программу **numbers.py**, которая выводит на экран 10 случайных чисел. Для вывода используется цикл **while**.

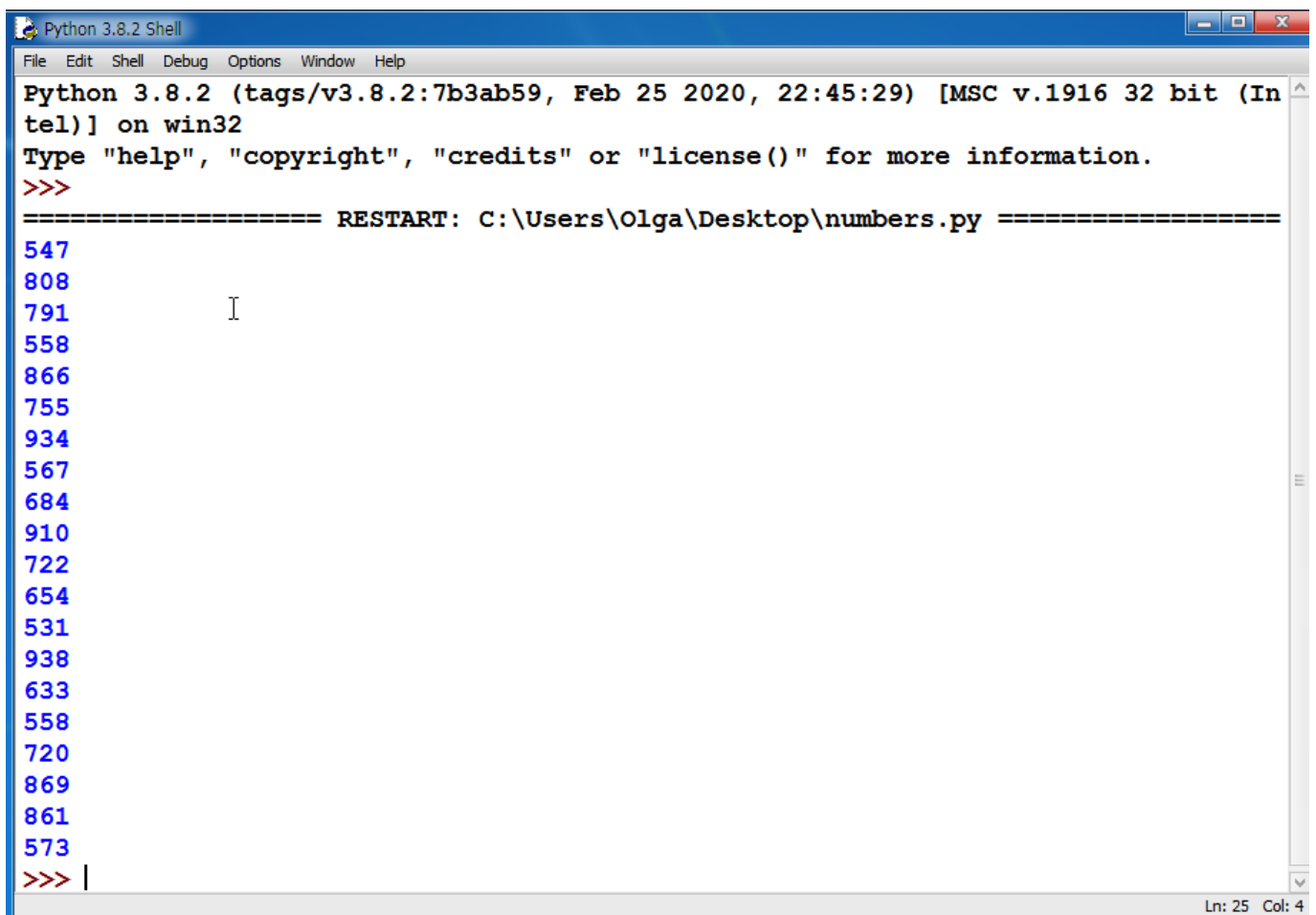
1. Создай скрипт **numbers.py** в своей папке, введи следующий код:

```
import random
k = 1
while k <= 10 :
    number = random.randint(1, 100)
    print(number)
    k = k + 1
```

2. Запустите скрипт из окна редактора. Убедись, что программа вывела случайные целые числа из диапазона от 1 до 100.

## Задание 7 (самостоятельно)

1. Открой скрипт **numbers.py**
2. Измени программу следующим образом: пусть на экран выводятся 20 случайных чисел в диапазоне от 500 до 1000 включительно.



```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Olga\Desktop\numbers.py =====
547
808
791
558
866
755
934
567
684
910
722
654
531
938
633
558
720
869
861
573
>>> |
```



## Угадай случайное число

Давайте опробуем работу модуля **random** в программе **guess\_number.py** (угадай число). В этой программе:

- Компьютер выбирает случайное число в диапазоне от 1 до 100.
- Игрок пытается отгадать это число.
- Компьютер сравнивает предложенное число с загаданным и сообщает игроку результат сравнения: больше, меньше или равенство.

В программе используется игровой цикл с оператором **while**.

В качестве условия используется оператор сравнения "**!=**" – не равно.

Цикл будет продолжаться до тех пор, пока игрок не отгадает задуманное компьютером число.

### Задание 8 (guess\_number.py)

1. Создай скрипт **guess\_number.py** в своей папке, введи следующий код:

```
import random
number = random.randint(1, 100)
print("Компьютер загадал число в диапазоне от 1 до 100")
print("Желаєте угадати это число?")
guess = int(input("Введите целое число от 1 до 100: "))
while guess != number :
    if guess > number :
        print(guess, "слишком велико. Попробуйте снова")
    if guess < number :
        print(guess, "слишком мало. Попробуйте снова")
    guess = int(input("Введите целое число от 1 до 100: "))
print(guess, " – правильное число. Вы угадали!")
```

2. Запусти скрипт из окна редактора.
3. Попробуй угадать число.
4. Выбери стратегию для ввода числа на основании сообщений программы.

### Задание 9\* (самостоятельно)

Чтобы сделать программу более интересной, можно ввести в программу подсчет попыток пользователя при угадывании числа.

1. Сохрани скрипт **guess\_number.py** под новым именем **guess\_nuumber\_2.py** в своей папке.
2. Создай новую переменную, например, с именем **attempt** (попытка), к значению которой прибавлялась бы 1 при каждой новой попытке.
3. В конце работы программы нужно сообщить пользователю о количестве попыток, чтобы он знал, как хорошо он справился с заданием.
4. Запусти скрипт из окна редактора.



## 5. Проверь правильность работы программы.

### Получение случайных цветов из списка

В модуле **random** есть функция **choice()**. Эта функция может принимать список данных в качестве аргумента (указывается в скобках) и возвращать случайным образом выбранный элемент из этого списка. В нашем случае мы создадим список цветов и затем передадим этот список функции **choice()**, чтобы получить случайный цвет для отображения на экране.

Рассмотрим программу **colors.py**, которая выводит на экран 10 названий цветов, выбранных случайным образом из списка.

### Задание 10 (colors.py)

1. Создай скрипт **colors.py** в своей папке.
2. Введи следующий код:

```
import random
colors = ["red", "yellow", "blue", "green", \
          "orange", "grey", "brown", "white"]
k = 1
while k <= 10 :
    color = random.choice(colors)
    print(color)
    k = k + 1
```

3. Запусти скрипт из окна редактора.
4. Убедись, что программа вывела случайные цвета из заданного списка.

### Задание 11 (name10.py) (самостоятельно)

1. Создай файл с именем **name10.py** в своей папке.
2. Объяви список с именем **names**, в котором будет десять любых имен людей.
3. Выведи с помощью цикла пять имен, выбранных случайным образом из списка. В качестве примера используй скрипт **colors.py**

## Рисование окружностей со случайными параметрами

Рассмотрим для начала программу рисования с помощью черепашки нескольких окружностей в виде розетки.

### Задание 12 (circle\_6.py)

1. Создай файл **circle\_6.py**
2. Введи следующий код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
number = 6 # Количество окружностей 6
# Заголовок графического окна:
window.title("Количество окружностей = " + str(number))
for k in range(number) :
    radius = 100 # Задание радиуса окружностей
    color = "red" # задание цвета окружностей
    width = 5 # Задание толщины пера
    t.width(width)
    t.color(color)
    t.circle(radius)
    t.left(360/number)
```

3. Запусти программу на выполнение.
4. Убедись, что она работает верно

### Задание 13 (самостоятельно)

На основе скрипта **circle\_6.py** создай новый файл и назови его **circle\_rand\_1.py**, в котором количество окружностей выбирается случайным образом из диапазона от 4 до 10.

#### *Подсказка*

Переменная **number** выбирается с помощью функции **random.randint(4, 10)**

### Задание 14 (самостоятельно)

На основе скрипта `circle_rand_1.py` создай новый файл и назови его `circle_rand_2.py`.

Программа должна выбирать цвет окружностей случайным образом из списка цветов.

*Подсказка*

Перед циклом в программу необходимо поместить список цветов:

```
colors = ["red", "blue", "green", "orange", "brown", "purple"]
```

В цикле нужно записать инструкцию для выбора цвета случайным образом из списка:

```
color = random.choice(colors)
```

### Задание 15 (самостоятельно)

На основе скрипта `circle_rand_2.py` создайте новый файл и назовите его `circle_rand_3.py`.

Программа должна выбирать радиус окружностей случайным образом из диапазона от 10 до 150.

*Подсказка*

Для этого нужно записать в тело цикла ввести инструкцию

```
radius = random.randint(10, 150)
```

### Задание 16 (самостоятельно)

На основе скрипта `circle_rand_3.py` создайте новый файл и назовите его `circle_rand_4.py`.

Программа должна выбирать толщину пера случайным образом из диапазона от 1 до 6.

*Подсказка*

Для этого нужно записать в тело цикла вести инструкцию

```
width = random.randint(1, 6)
```

На следующем уроке вы закрепите свои знания по модулю `random` и узнаете о том, как работает программа «Камень, Ножницы, Бумага».