

ЦИКЛЫ

На этом уроке вы:

- Научитесь использовать для повторяющихся задач циклы, чтобы избежать утомительных вводов вручную.
- Группировать команды, которые нужно дублировать, в блоки кода и помещать их в циклы.
- Научитесь работать со списками.

Оператор цикла *for*

Предположим, что нам нужно вывести на дисплей все числа от 1 до 100 по одному в строке. Обычным способом пришлось бы писать 100 строк программы:

```
print(1)
print(2)
print(3)
print(4)

# и так 100 раз
print(100)
```

При помощи цикла **for** те же действия можно выполнить гораздо короче:

```
for k in range(1, 101) :
    print(k)
```

Углубимся в код.

Отступы играют важную роль при программировании на любом языке, так как позволяют обозначить структуру программы и выделить группы инструкций, принадлежащих другим инструкциям.

Но в языке Python отступы играют еще более важную роль: они помогают интерпретатору Python понять смысл вашей программы.

Отступы важны в циклах, так как показывают, какие инструкции принадлежат циклу и должны повторяться.

Отступ — это пространство слева в каждой строке программы. Отступы используются, чтобы обозначить структуру программы и сгруппировать программные инструкции в циклах и других инструкциях. В языке Python отступы играют еще более важную роль, так как определяют смысл программы.

В языке Python используются два типа цикла: **for** и **while**.

Цикл **for** применяется для перебора элементов последовательности. Обычно цикл **for** имеет следующий формат:

```
for переменная_цикла in range( ) :
```

Инструкции внутри цикла

Инструкция внутри цикла называется в программировании блоком. **Блок** – это набор сгруппированных инструкций. Все инструкции в блоке должны иметь одинаковый отступ. В Python принято соглашение, что отступы для инструкции в блоке должны иметь 4 пробела.

Когда вы набираете текст программы в оболочке или редакторе Python, то после набора инструкции

```
for переменная_цикла in range( ) :
```

вы нажимаете клавишу Enter, программа автоматически выполнит отступ в 4 пробела и курсор окажется на 4 пробела правее.

Все инструкции блока будут иметь одинаковый отступ в 4 пробела. Когда нужно отменить ввод инструкций блока, следует ликвидировать отступ, щелкнув мышкой по началу новой строки.

Функция **range()** служит для создания списка чисел, который начинается с первого числа в скобках и заканчивается числом на единицу меньше второго числа. Она создает список элементов. Эти элементы поступают в переменную цикла при каждом выполнении цикла.

Слово **range** переводится как *диапазон*. Таким образом, функция **range()** задает диапазон изменения переменной цикла.

Функция **range()** имеет следующий формат:

```
range( Начало, Конец, Шаг)
```

- Первый параметр – Начало, задает начальное значение диапазона. Если этот параметр не указан, то по умолчанию используется значение 0.
- Во втором параметре – Конец, указывается конечное значение. Следует отметить, что это значение не входит в возвращаемое значение.
- Третий параметр – Шаг, определяет закон изменения переменной цикла. Если параметр Шаг не задан, то по умолчанию используется значение 1.

Рассмотрим примеры реализации цикла **for**.

Задание 1. for1.py

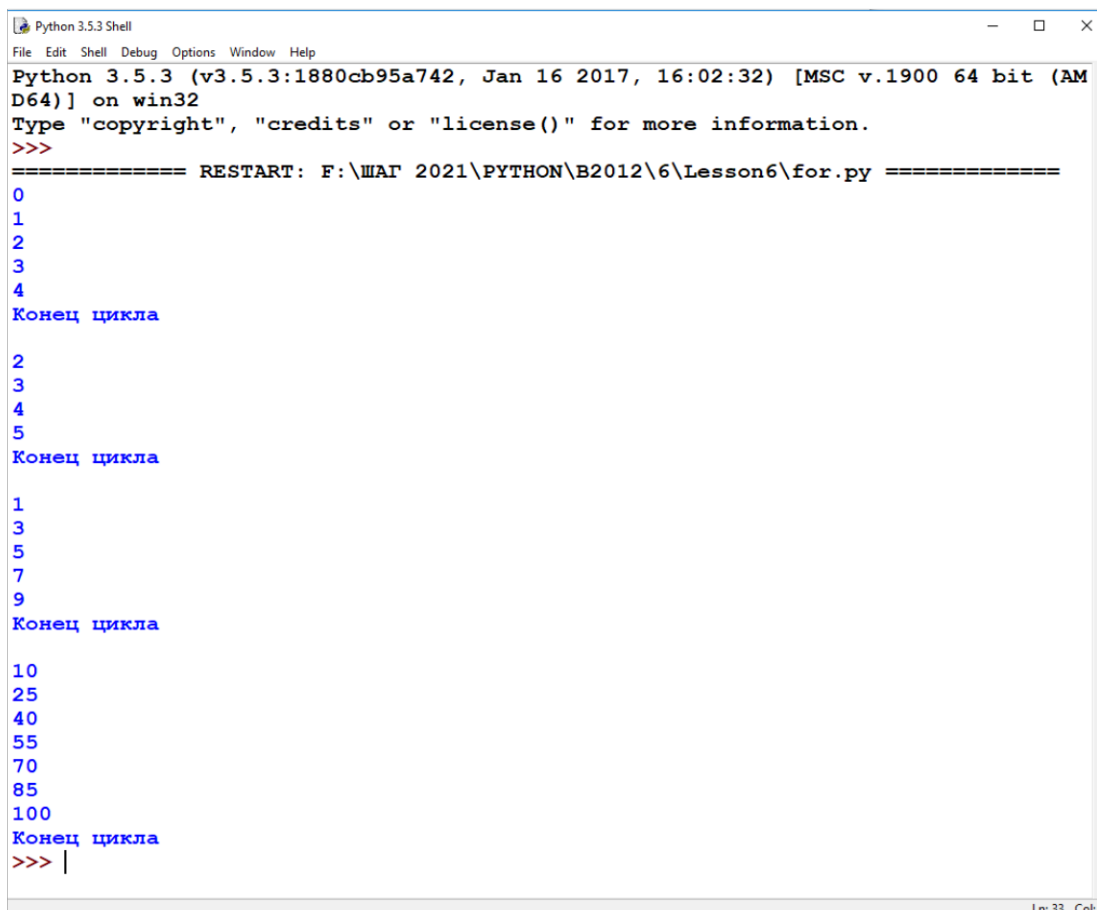
1. Создай файл с именем **for1.py** в папке **Lesson6**.
2. Введи код. Обязательно соблюдай все отступы:

```
for k in range(5):      # Диапазон k: от 0 до 4, шаг равен 1
    print(k)
print("Конец цикла")

input() # Для нажатия Enter
for k in range(2, 6):   # Диапазон k: от 2 до 5, шаг равен 1
    print(k)
print("Конец цикла")

input() # Для нажатия Enter
for k in range(1, 10, 2): # Диапазон k: от 1 до 9, шаг равен 2
    print(k)
print("Конец цикла")
```

3. Запусти скрипт на выполнение.
4. Убедись в правильности работы программы. Не забывай нажимать клавишу **Enter** для продолжения вывода результатов.
5. Добавь в программу код, который будет выводить числа от **10** до **100** с шагом **15**.



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\ШАГ 2021\PYTHON\B2012\6\Lesson6\for.py =====
0
1
2
3
4
Конец цикла

2
3
4
5
Конец цикла

1
3
5
7
9
Конец цикла

10
25
40
55
70
85
100
Конец цикла
>>> |
```

Задание 2 for2.py

1. Создай файл с именем **for2.py** в папке **Lesson6**.
2. Введи код:

```
x = "Молодец"
for j in range(20):
    print(x, end = " ")
print("Конец цикла")
```

3. Запусти скрипт на выполнение:

```
===== RESTART: F:\ШАГ 2021\PYTHON\B2012\6\Lesson6\for2.py =====
Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец
Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец Молодец
Конец цикла
>>>
```

В выражении **print()** в последней строке программы появилось кое-что новое: теперь эта функция содержит ключевое слово-аргумент.

В данном случае ключевым словом является слово **end**, с помощью которого мы говорим программе заканчивать каждое выражение **print()** пробелом (между кавычками находится пробел: " ") вместо обычного символа переноса строки.

Как правило, на языке Python выражения **print()** заканчиваются символом переноса строки, это аналогично нажатию клавиши Enter на клавиатуре. Но с помощью данного ключевого слова-аргумента мы сообщаем Python о том, что не хотим распечатывать текст каждый раз на новой строке.

4. Измени код в программе таким образом, чтобы после каждого слова молодец выводился восклицательный знак:

```
===== RESTART: F:\ШАГ 2021\PYTHON\B2012\6\Lesson6\for2.py =====
Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец!
Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец! Молодец!
Молодец! Молодец! Конец цикла
>>> |
```

Задание 3* for3.py

1. Создай и сохрани скрипт под именем **for3.py** в папке **Lesson6**.
2. Напиши программу, которая будет выполнять следующие действия:
 - 1) Запрашивать у пользователя его имя
 - 2) Выводить полученное имя в строчку 50 раз.

Подсказка. Сначала придется создать переменную с именем **name** которая будет запрашивать имя пользователя с помощью функции **input()** :

```
name = input("Как тебя зовут? ")
```

- ### 3. Запусти скрипт на выполнение:

[illegible]

Задание 4*. for4.py

1. Сохрани программу **for3.py** под новым именем **for4.py**
2. Модифицируй программный код таким образом, чтобы, компьютер печатал фразу "**name** – молодец!" 50 раз. Вместо **name** – введение имя.

Подсказка. Ключевое слово аргумент **end** = "– Молодец! "

- ### 3. Запусти скрипт на выполнение:

```
===== RESTART: F:\ШАГ 2021\PYTHON\B2012\6\Lesson6_mod\for4.py =====  
Как тебя зовут? Вероника  
Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Веро  
ника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника  
- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Мо  
лодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молоде  
ц! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! В  
ероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Верон  
ика- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника-  
Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Мол  
одец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец  
! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Ве  
роника- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Верони  
ка- Молодец! Вероника- Молодец! Вероника- Молодец! Вероника- Молодец! Конец цикл  
а  
>>>
```

Задание 5 circle.py

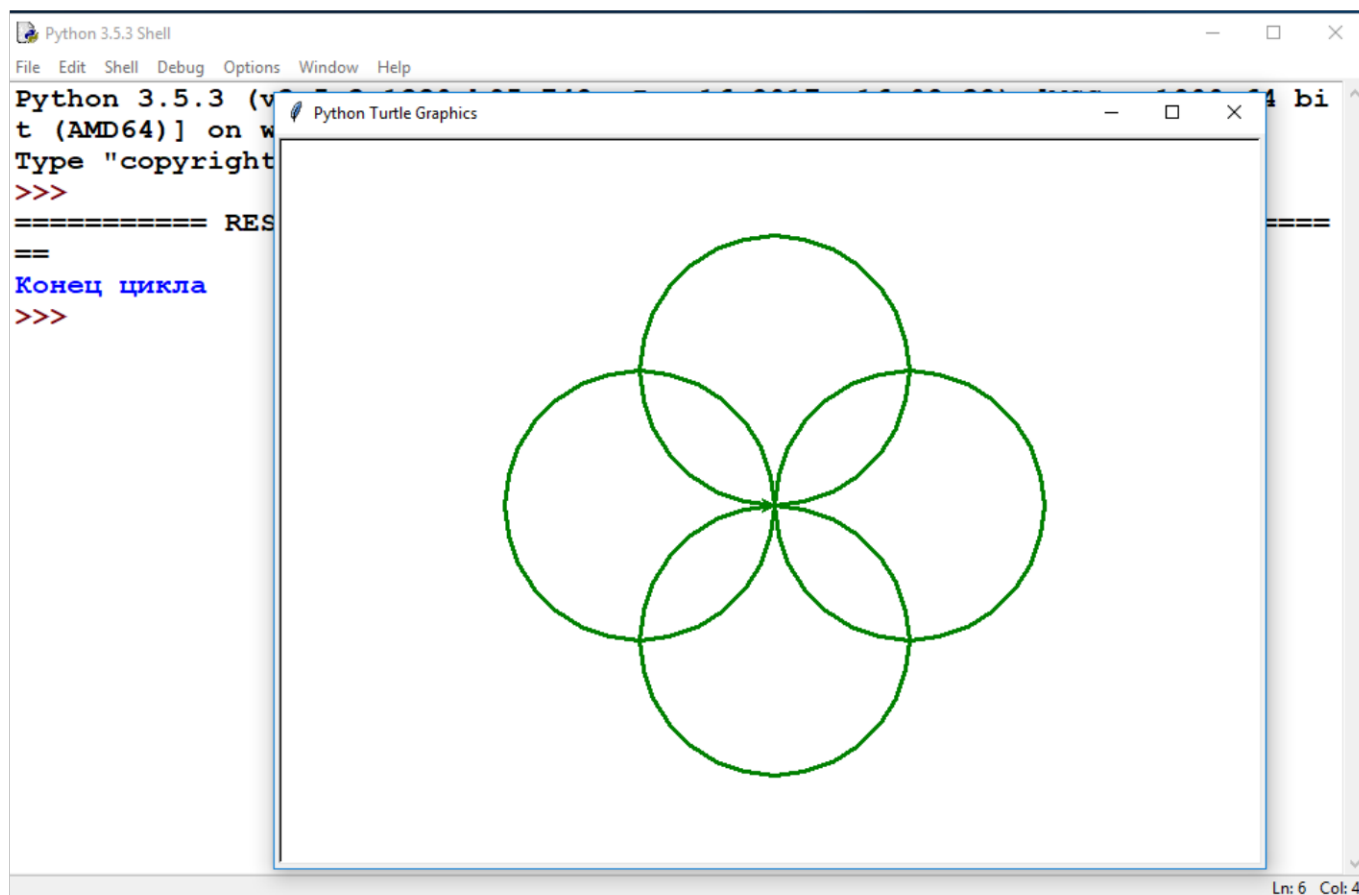
Напиши скрипт демонстрации цикла *for* для рисования четырех окружностей.

1. Создай файл с именем **circle.py** в папке **Lesson6**.
2. Введи код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()

t.width(3)           # толщина линии черепашки 3 пикселя
t.color('green')      # задаем зеленый цвет пера
for k in range(4):    # рисуем 4 окружности
    t.circle(100)      # радиус 100 пикселей
    t.left(90)         # поворот черепашки на 90 градусов.
print("Конец цикла")
```

3. Запусти скрипт на выполнение и проверь его работоспособность:



Задание 6 circle6.py (самостоятельно)

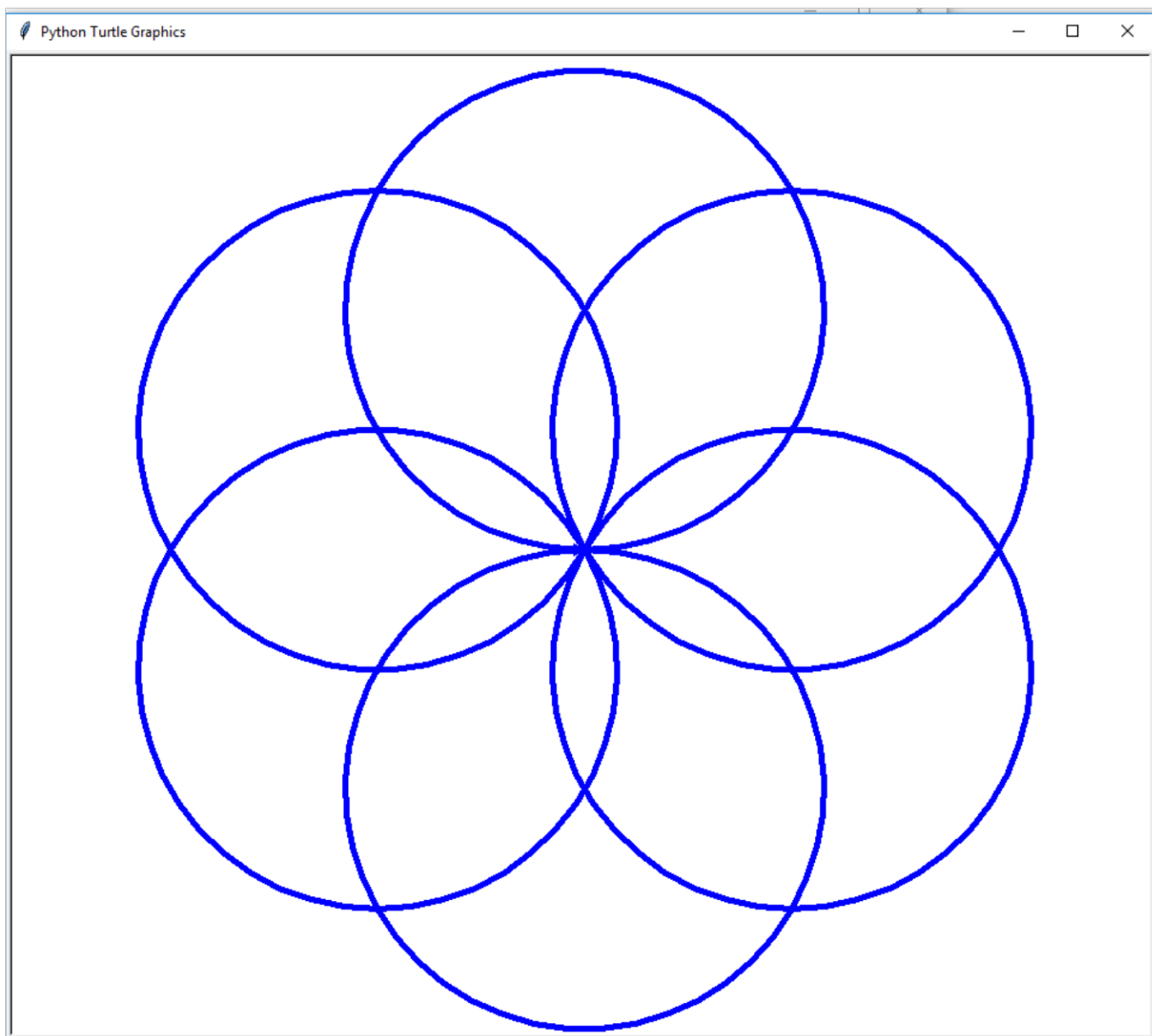
1. Сохрани программу **circle.py** под новым именем **circle6.py**
2. Измени скрипт таким образом:

программа должна рисовать **6 синих** окружностей, диаметром **200 пикселей**, установи толщину линии черепашки **5 пикселей**.

Указание. Для формирования шести окружностей надо использовать поворот черепашки на 60° .

Пояснение. Чтобы рассчитать угол поворота нужно 360° разделить на количество окружностей. Так как мы будем рисовать 6 окружностей, то угол поворота рассчитывается по формуле $360^\circ/6=60^\circ$.

3. Запусти скрипт на выполнение и проверь его работоспособность:



Списки

Во всех предыдущих программах рисования с помощью черепашки, все окружности были одного цвета.

При этом цвет задавался **перед** выполнением цикла.

Чтобы программа могла выбирать желаемый цвет во время выполнения цикла, надо поместить названия цветов в **список**.

Список — это последовательность элементов. Списки могут хранить числа, строки, по отдельности, или в комбинации с числами, а также хранить другие списки. Элементы списка отделяются друг от друга запятыми, а весь список заключается в квадратные скобки [].

Примеры объявления списков:

из чисел:	<code>num_list = [2, 4, 6, 8]</code>
из строк:	<code>family = ["Папа", "Мама", "Сестра", "Брат", "Я", "Кошка"]</code>
из строк и чисел:	<code>mix_list = ["abc", 4, "words", 7]</code>

Каждый элемент в списке характеризуется своей позицией (индексом), которая нумеруется цифрами: 0, 1, 2, ... Так, если список имеет 4 элемента, то каждый из них имеет индексы: 0, 1, 2, 3.

Рассмотрим скрипт для демонстрации работы со спискам.

Задание 7 (list.py)

1. Создай файл с именем **list.py** в папке **Lesson6**.
2. Введите код:

```
# Задание списка из четырех элементов – цветов для отображения

colors = ["red", "brown", "blue", "green"]
print(colors)

input()
print(colors[0]) # Доступ к 0-му элементу списка

input()
print(colors[3]) # Доступ к 3-му элементу списка
```


3. Запусти скрипт на выполнение и проверь его работоспособность. Для продолжения вывода результата на печать используй клавишу Enter:
4. Допиши в программе **list.py** код, чтобы он выводил в оболочку слово **"brown"**
5. Запусти скрипт на выполнение:

```
===== RESTART: F:\ШАГ 2021\PYTHON
['red', 'brown', 'blue', 'green']

red

green

green
>>> |
```

Напомню названия основных цветов:

Название цвета	Название цвета в Python
Белый	"white"
Черный	"black"
Коричневый	"brown"
Красный	"red"
Желтый	"yellow"
Синий	"blue"
Зеленый	"green"
Золотистый	"gold"
Оранжевый	"orange"
Розовый	"pink"
Серый	"gray"
Светло-серый	"light gray"

Рассмотрим скрипт для демонстрации рисования четырех разноцветных окружностей.

Задание 8 circle_color.py

Скрипт **circle_color.py** похож на скрипт **circle.py**, поэтому для удобства поступи следующим образом:

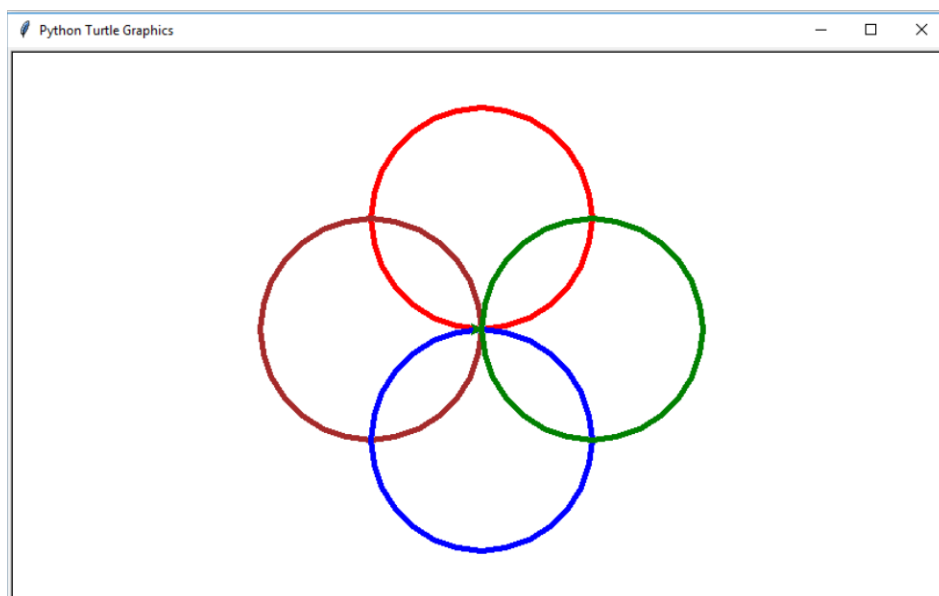
1. Открой файл с именем **circle.py**.
2. Сохрани его под новым именем **circle_color.py**
3. Внеси изменения (я выделила жирным):

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.width(5) # Толщина линии
colors = ["red", "brown", "blue", "green"] # Список цветов
for k in range(4): # рисуем 4 окружности
    t.color(colors[k]) # выбираем цвета из списка по индексу k
    t.circle(100) # радиус 100 пикселей
    t.left(90) # угол поворота 90 градусов
print("Конец цикла")
```

Пояснение к скрипту.

Инструкция **t.color(colors[k])** выбирает из списка **colors** тип цвета по значению индекса **k**. При первом проходе цикла значение **k**, выбираемое из функции **range** равно 0. Поэтому из списка **colors** выбирается цвет **"red"** и рисуется красная окружность. Во втором проходе цикла **k = 1** и выбирается цвет **"brown"** – коричневый. И т.д.

4. Запусти скрипт на выполнение:



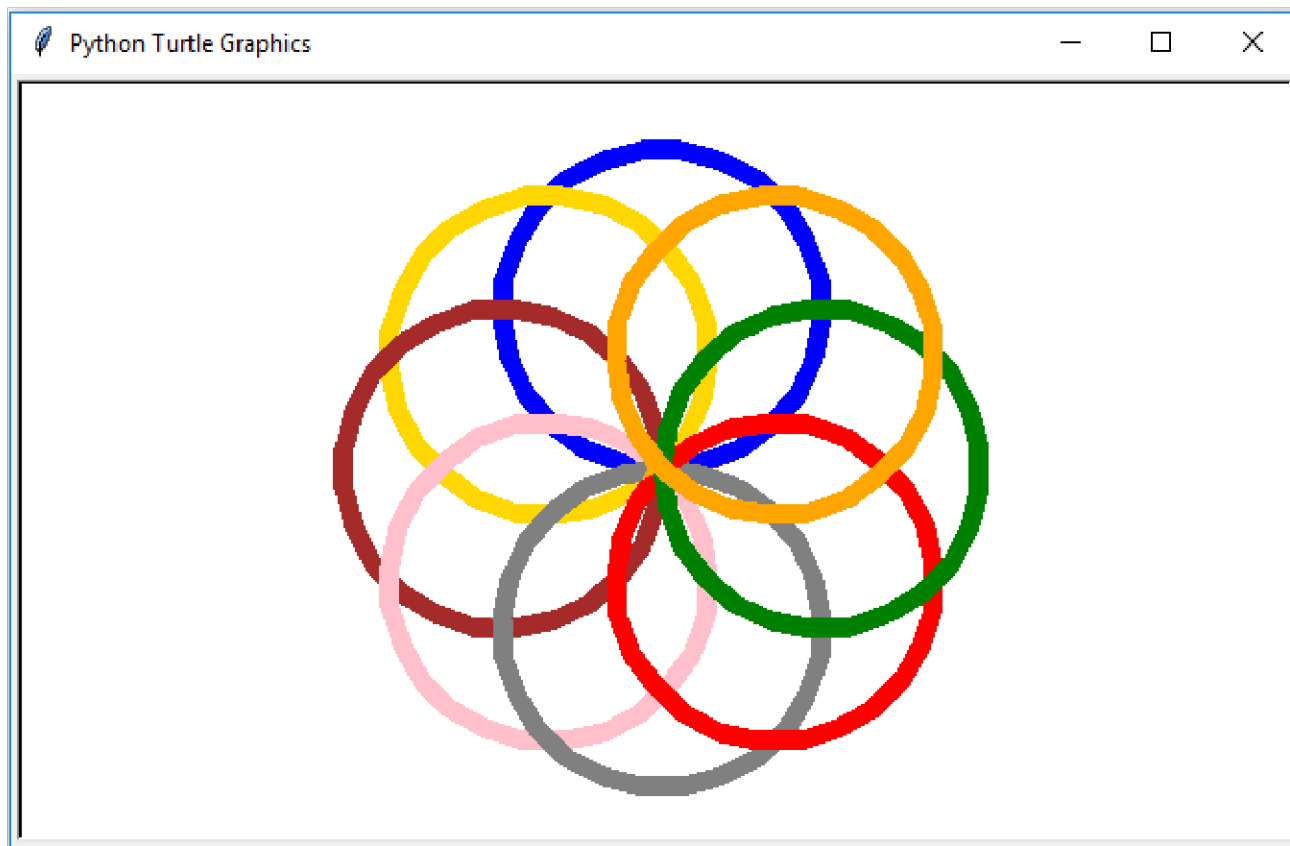
Задание 9 circle_color_8.py (самостоятельно)

1. Открой файл с именем **circle.py**, если ты его закрыл.
2. Сохрани его под новым именем **circle_color_8.py**
3. Модифицируй скрипт:
 - Напиши код для рисования **8 окружностей** диаметром **80 пикселей** с разными цветами. Толщина пера **10 пикселей**.
 - Для задания цветов создай список с восемью цветами:
 - синий,
 - золотистый,
 - коричневый,
 - розовый,
 - серый,
 - красный,
 - зеленый,
 - оранжевый.

Угол поворота должен быть равным 45° .

Пояснение. Чтобы рассчитать угол поворота нужно 360° разделить на количество окружностей. Так как мы будем рисовать 8 окружностей, то угол поворота рассчитывается $360^\circ/8=45^\circ$.

4. Запусти скрипт на выполнение:



Задание 10 circle_line.py

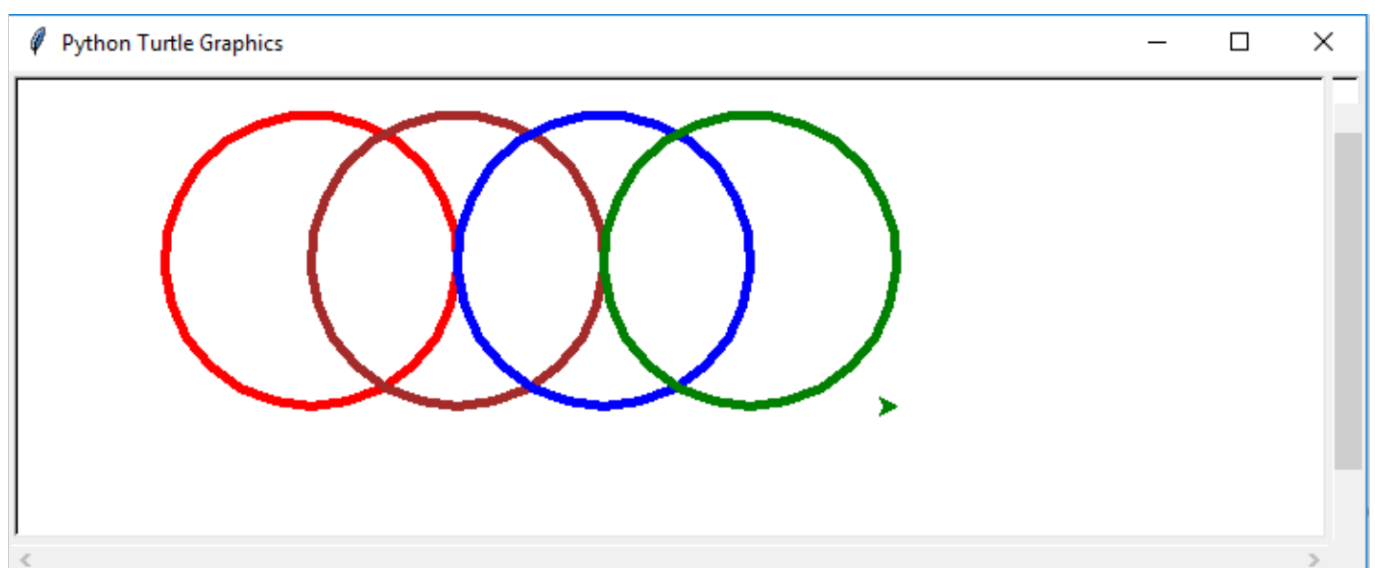
1. Создай файл с именем **circle_line.py** в папке **Lesson6**.
2. Введи код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.width(5)
# список из цветов для окружности:
colors = ["red", "brown", "blue", "green"]
t.penup() # поднять перо
t.goto(-200, 0) # Перемещаем черепашку в координаты: x=-200, y=0
for k in range(4): # рисуем 4 окружности
    t.pendown() # опустить перо
    t.color(colors[k]) # выбираем цвета из списка поочередно
    t.circle(80) # радиус окружности 80
    t.penup() # поднять перо
    dist = -200 + (k+1)*80 # Расчет смещения по оси x на 80 пикселей
    t.goto(dist, 0) # смещение
```

По умолчанию черепашка появляется в центре, с координатами $X = 0$, $Y = 0$. В строке 7 мы переместим черепашку из центра влево `t.goto(-200, 0)`. Это нужно для того, чтобы при рисовании окружностей общий рисунок будет смещаться вправо.

В строке 13 мы ввели переменную **dist**. С помощью нее мы будем смещать перо на длину радиуса нашей окружности – 50 пикселей. Таким образом каждая последующая окружность будет сдвигаться вправо, перекрывая половину предыдущей.

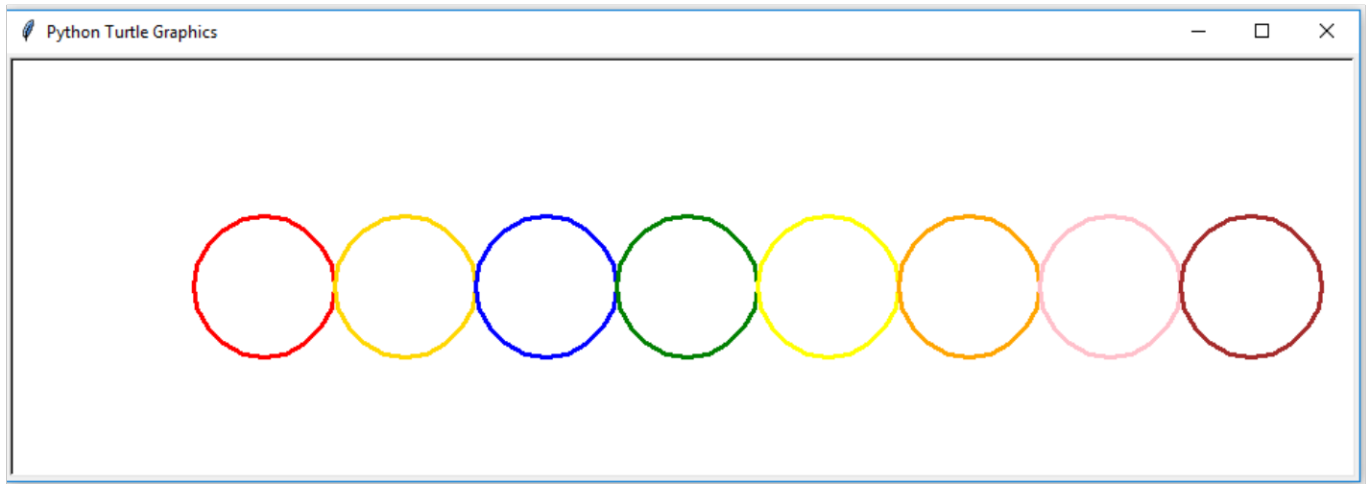
3. Запусти скрипт на выполнение и проверь его работоспособность.



4. (самостоятельно)

Модифицируй скрипт **circle_line.py** таким образом, чтобы он рисовал **8 окружностей** разных цветов. Толщина линии окружности **3 пикселя**, диаметр **40**.

Окружности должны смещаться по оси **X** таким образом, чтобы не накладывались друг на друга, а рисовались рядом, одна за одной.



Рассмотрим способы построения различных фигур с помощью черепашки.

Рисование квадратной спирали

Задание 11 square.py

1. Создайте файл с именем **square.py** в папке **Lesson6**.
2. Введите код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.speed(0) # Устанавливаем максимальную скорость черепашки
for k in range(100) :
    t.forward(k)
    t.left(90) # угол поворота 90
```

Пояснение к программе.

Стрелка черепашки на экране рисует прямую линию длиной **k** пикселей, затем поворачивает налево на 90° и снова рисует. И так повторяется снова и снова. При этом длина линии после каждого поворота увеличивается на 1 пиксель. Конечная длина линии будет равна 99 пикселей и цикл заканчивается.

3. Запусти скрипт на выполнение и проверь его работоспособность.

Задание 11 square200.py (самостоятельно)

1. Сохрани файл с именем **square.py** под новым именем **square200.py**
2. Внеси изменения:
3. Задай красный цвет квадратов увеличь число повторений цикла до 200.

Подсказка. Нужно добавить в код инструкцию **t.color('red')** до начала выполнения цикла.

Рисование спиралевидных квадратов

Задание 12 square_spiral.py

1. Создай файл с именем **square_spiral.py** в папке **Lesson6**.
2. Введите код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.speed(0)    # Устанавливаем максимальную скорость черепашки
for k in range(100) :
    t.forward(k)
    t.left(91) # угол поворота 91
```

Пояснение к скрипту.

При повороте черепашки 4 раза в цикле на 90° создается идеальный квадрат. Поворот же на угол чуть больше 90° , в данном случае на 91° , немного искажает квадрат. И поскольку к следующему повороту квадрат немного искажен, новая форма выглядит все менее и менее похожей на квадрат по мере выполнения программы.

В результате такого искажения программа создает спиралевидную фигуру, которую ты видишь на экране.

3. Запустите скрипт на выполнение и проверьте его работоспособность.
4. *(самостоятельно)*

Задай синий цвет квадратов в программе и измени угол поворота на 94° , увеличь число повторений цикла до 200.

Задание 13 circle100.py

1. Создайте файл с именем **circle100.py** в папке **Lesson6**.
2. Введите код:
- 3.

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.speed(0)
for k in range(100) :# количество повторений 100
    t.circle(k)
    t.left(91)# угол поворота 91 градус
```

4. Запусти скрипт на выполнение и проверь его работоспособность.
5. *(самостоятельно)*
Установи зеленый цвет окружностей в скрипте и увеличь число повторений цикла до 200, а угол поворота сделай 100°.

Четырехцветная спираль

Задание 14 spiral_color.py

1. Создайтфайл с именем **spiral_color.py** в папке **Lesson6**.
2. Введи код:

```
import turtle
window = turtle.Screen()
t = turtle.Pen()
t.speed(0)
colors = ["red", "yellow", "blue", "green"]
for k in range(100) : # количество повторений 100
    t.color(colors[k % 4])
    t.circle(k)
    t.left(91)          # угол поворота 91 градус
```

3. Запусти скрипт на выполнение и проверь его работоспособность.

Пояснение к программе.

Символ `%` в выражении `[k % 4]` представляет операцию определения остатка от деления.

Например, `5 % 4` равняется `1` с остатком `1`, так как `5` можно нацело разделить на `4` только с остатком `1`, остаток от деления `6 % 4` равняется `2`, и так далее.

Оператор определения остатка от деления может быть полезен при необходимости циклично пройти по нескольким элементам списка, аналогично тому, как мы поступаем со списком из четырех цветов.

За 100 проходов цикла выражение `colors[k % 4]` пройдет по всем четырем цветам (0, 1, 2 и 3 для красного, желтого, синего и зеленого) 25 раз.

При первом проходе по циклу используется первый цвет в списке `"red"` – красный, во втором проходе используется `"yellow"` - желтый и т.д.

Затем на пятом проходе цикла Python вновь вернется красному и цикл повторится. После каждого четвертого прохода цикл будет вновь возвращаться к красному цвету.

4. (самостоятельно)

Измени цвет холста (цвет фона) на черный. Для этого нужно использовать инструкцию `window.bgcolor("black")`, которую нужно поместить перед циклом.

Измени количество повторений на 150, а угол поворота на 100° .