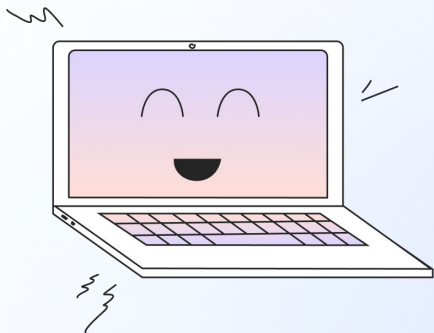


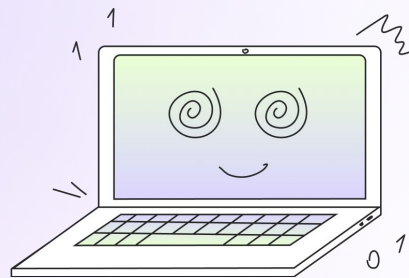
Программирование на Python

Урок 4
Циклы While и For





**Ставь + в чат,
если хорошо видно и слышно**



Давайте вспомним предыдущий урок =)

Переходим на сайт с викториной по **ссылке**,
которую отправит преподаватель





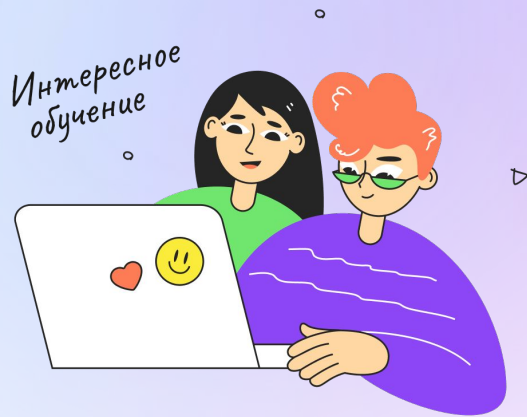
Что будет на уроке сегодня?

- Знакомство с новыми модулями - turtle и random
- Векторное рисование
- Циклы For и While
- Снова о важности отступов
- Вложенные конструкции
- Игровой цикл и игра «угадайка»





Знакомство с черепашкой





Подключаем черепашку

Черепашка — это дополнительный модуль python, с помощью которого можно создавать программируемую графику

Чтобы подключить черепашку к проекту, нужно в самом начале написать команду:

```
import turtle
```





Создаем холст и ставим паузу

Черепашке, как и любому художнику нужно пространство для рисования.

Создадим холст, с помощью команды:

```
t = turtle.Pen()
```

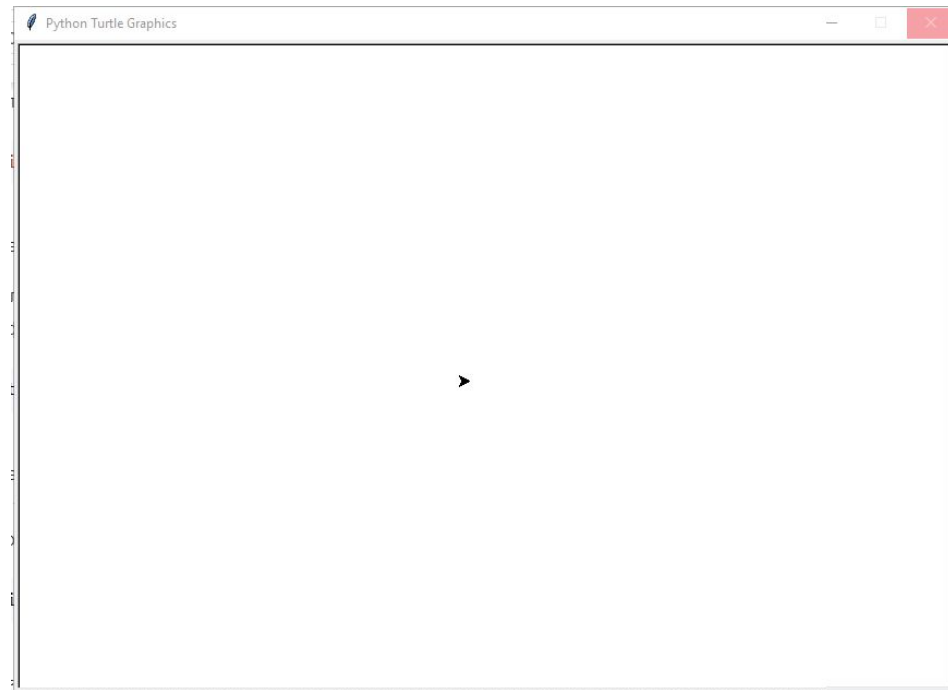
И в самом низу напишите команду, чтобы холст не закрывался быстрее, чем мы можем его увидеть:

```
turtle.exitonclick()
```



Запускаем, проверяем

```
import turtle  
t = turtle.Pen()  
turtle.exitonclick()
```

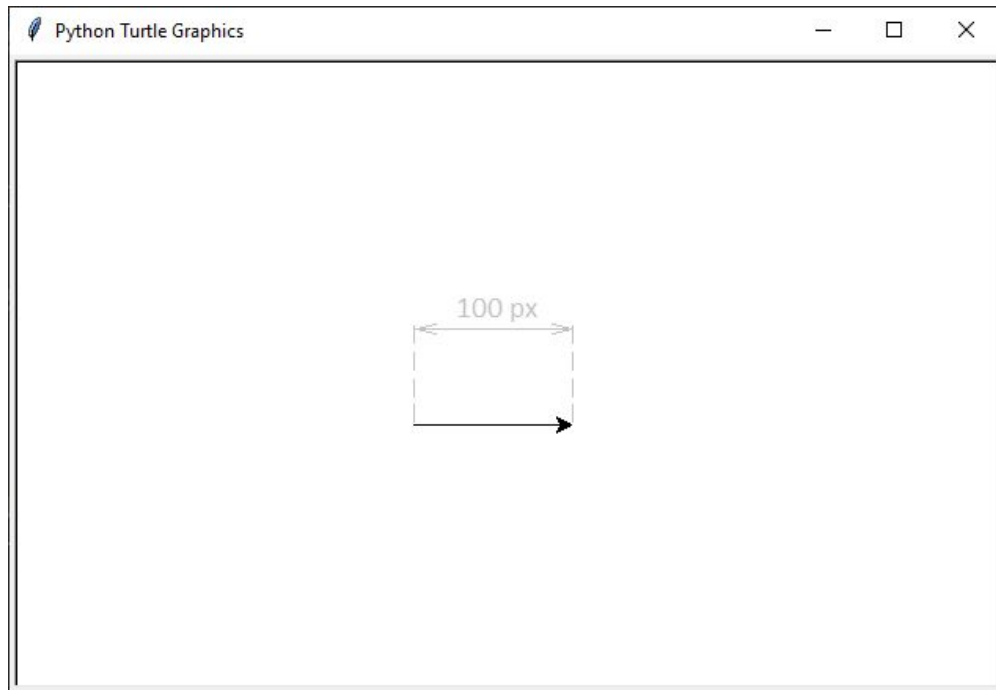




Рисуем линию

```
import turtle
t = turtle.Pen()
t.forward(100)      # Сдвигаем
                    # черепашку вперед на 100 пикселей
turtle.exitonclick()
```

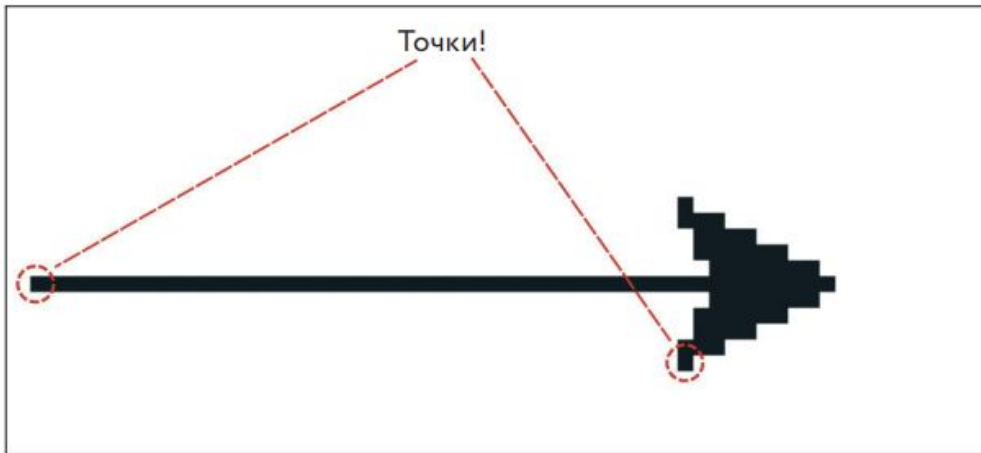
Важно! Черепашка сместилась вправо, потому что была направлена мордочкой в правую сторону. То есть она просто пошла вперед в том направлении, в которое смотрела.





Что такое пиксели

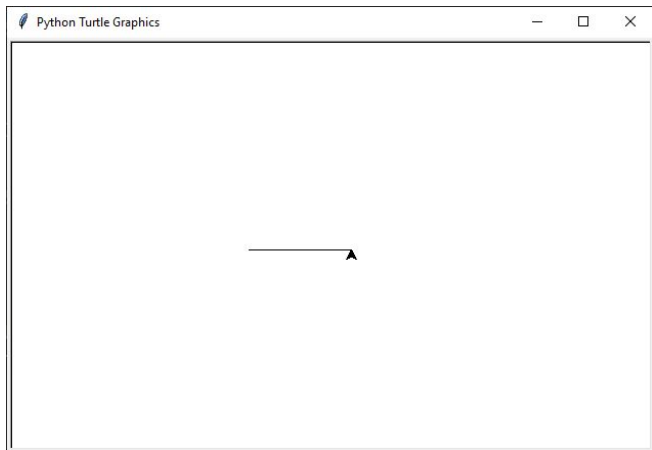
Пиксель — это одна экранная точка, самый маленький элемент изображения. Все, что вы видите на экране монитора, состоит из пикселей — крошечных квадратных точек. Если посмотреть в увеличении на холст и линию, которую нарисовала черепашка, обнаружится, что и след черепашки, и она сама — просто набор пикселей. Это и есть самая простая компьютерная графика.





Поворачиваем черепашку

```
import turtle
t = turtle.Pen()
t.forward(100)      # Смещаем черепашку вперед на 100 пикселей
t.left(90)         # Поворачиваем влево на 90°
turtle.exitonclick()
```





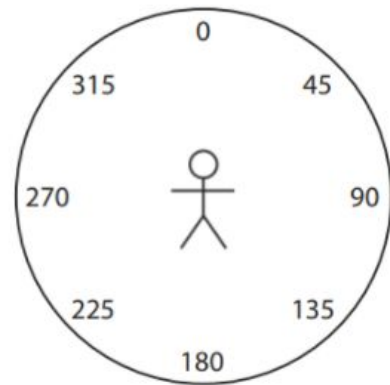
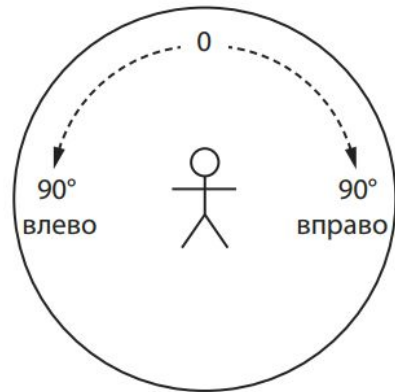
Что такое градусы

Вообразите, что вы стоите в центре круга.

- Направление, в котором вы смотрите, это 0 градусов.
- Если вы вытянете левую руку вбок, это будет 90 градусов влево.
- Если вы вытянете вбок правую руку, это будет 90 градусов вправо.

Если продолжать двигаться по часовой стрелке от вашей правой руки и дальше, 180 градусов — это прямо за вашей спиной. 270 градусов — там, куда указывает левая рука, а 360 градусов — направление вашего взгляда, то есть точка, откуда мы начали. Получается, что градусы проходят полный круг от 0 до 360. Вот круг, размеченный на градусы слева направо, с шагом в 45 градусов:

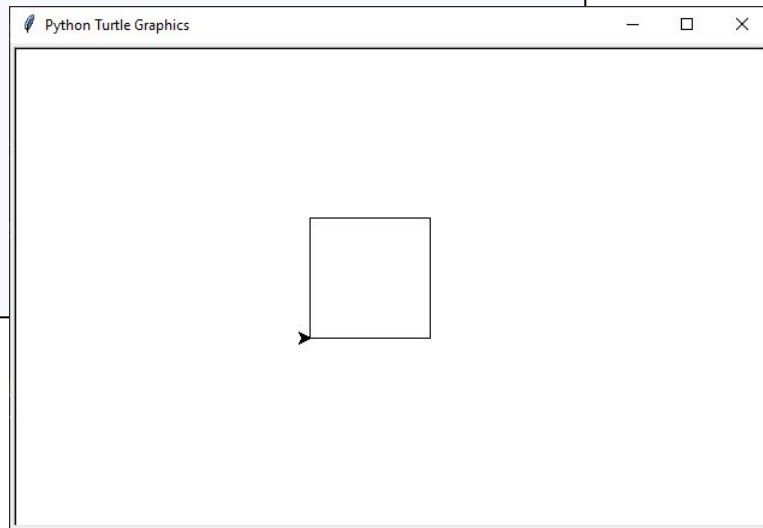
Поэтому команда `t.left(90)` разворачивает стрелочку острием вверх (так как вначале она указывала вправо):





Рисуем квадрат

```
import turtle
t = turtle.Pen()
t.forward(100)      # Смещаем черепашку вперед на 100 пикселей
t.left(90)         # Поворачиваем влево на 90°
t.forward(100)
t.left(90)
t.forward(100)
t.left(90)
t.forward(100)
t.left(90)
turtle.exitonclick()
```





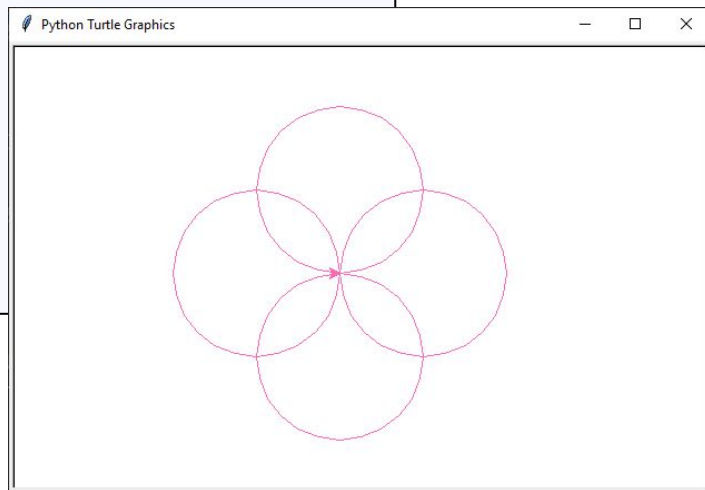
Ещё команды для рисования черепашкой:

```
t.right(90)           # Поворачиваем вправо на 90°
t.backward(100)      # Двигаемся назад
t.circle(100)        # Рисуем окружность диаметром 100 пикселей
t.color('#CD5C5C')   # Устанавливает цвет пера черепашки согласно коду цвета
t.penup()            # Поднимаем перо черепашки (черепашка не рисует)
t.pendown()          # Опускаем перо черепашки (черепашка рисует)
```



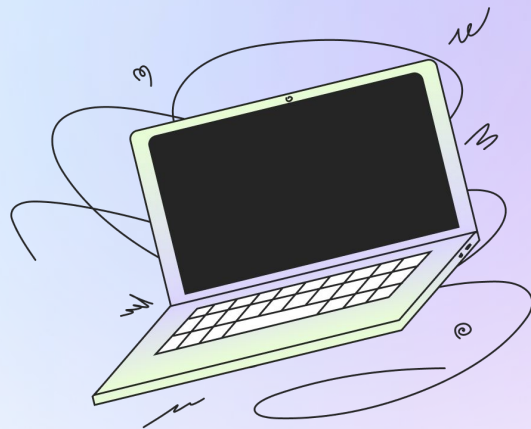
Поменяем линии на круги и сделаем их цветными!

```
import turtle
t = turtle.Pen()
t.color("#FF69B4") # Цвет пера розовый
t.circle(100)      # Рисуем круг диаметром 100 пикселей
t.left(90)        # Поворачиваем вправо на 90°
t.circle(100)
t.left(90)
t.circle(100)
t.left(90)
t.circle(100)
t.left(90)
turtle.exitonclick()
```





Цикл FOR





Повторение — мать учения, но не в коде программы :)

Программисты очень не любят, когда код в их программе повторяется.

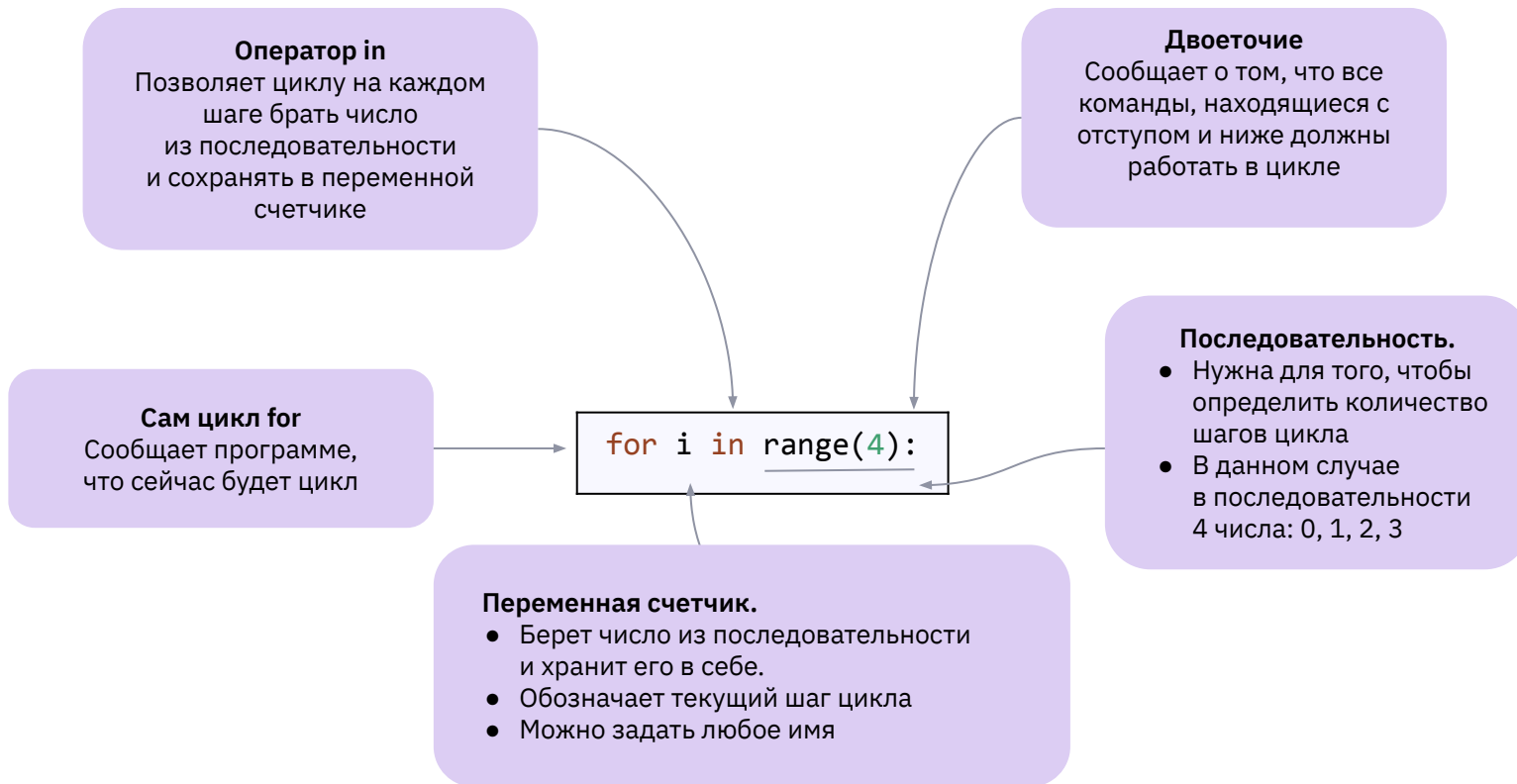
Наверное поэтому и придумали циклы :) Один из самых

распространенных циклов — FOR. **Его структура показана ниже:**

```
for <счетчик> in <последовательность> :  
    Команда 1  
    Команда 2  
    ...  
    Команда N
```



Структура цикла FOR





Оптимизируем розетку

```
import turtle
t = turtle.Pen()
t.color("#FF69B4")      # Цвет пера розовый

for i in range(4):      # Цикл повторяет 4 раза две команды с отступом ниже
    t.circle(100)
    t.left(90)

turtle.exitonclick()   # Эту команду не повторяем, поэтому без отступа
```

Запустите и убедитесь, что результат тот же, а кода стало меньше!
Да и программа смотрится теперь приятнее. Вот она сила циклов!



Немного поэкспериментируем

Давайте поменяем некоторые параметры (выделены желтым) и посмотрим на результат

```
import turtle
t = turtle.Pen()
t.color("#0000CD")      # Цвет пера синий

for i in range(6):      # Цикл повторяет 6 раз две команды с отступом ниже
    t.circle(100)
    t.left(60)

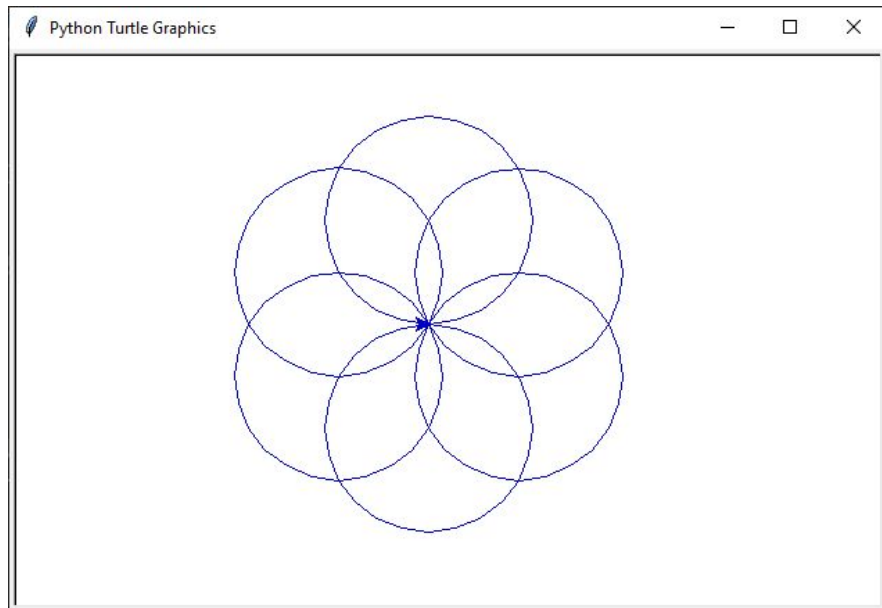
turtle.exitonclick()   # Эту команду не повторяем, поэтому без отступа
```



Что получили?

Если нам нужно нарисовать розетку из шести окружностей, то мы должны будем разделить ее на 6 поворотов влево, а не на 4. Вокруг центра рисунка можно описать 360 градусов: четыре поворота по 90 градусов провели нас на $4 \times 90 = 360$ градусов вокруг центра.

Если же мы разделим 360 на 6, а не на 4, то получим по $360 \div 60 = 60$ градусов для каждого поворота. Таким образом, при выполнении команды `t.left()` нам нужно поворачивать влево на 60 градусов при каждом прохождении цикла, то есть `t.left(60)`.





И внесем немного модернизации

Теперь, зная закономерность, можно вводить желаемое количество окружностей с клавиатуры! Только не забудьте преобразовать введенное число в цифру :)

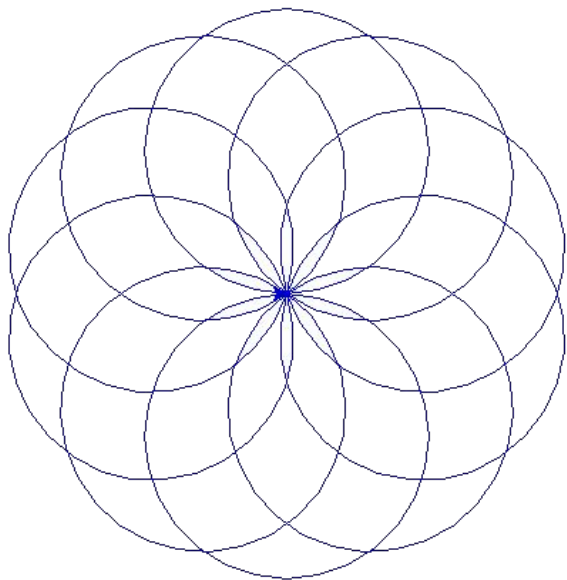
```
import turtle
t = turtle.Pen()
t.color("#0000CD") # Цвет пера синий
number = input('Введите количество окружностей') # Сохраняем ввод в переменную
number = int(number) # Преобразуем в число
for i in range(number):
    t.circle(100)
    t.left(360 / number) # Рассчитываем угол

turtle.exitonclick()
```



И внесем немного модернизации

Вот такая розетка получится, если задать количество окружностей 10!





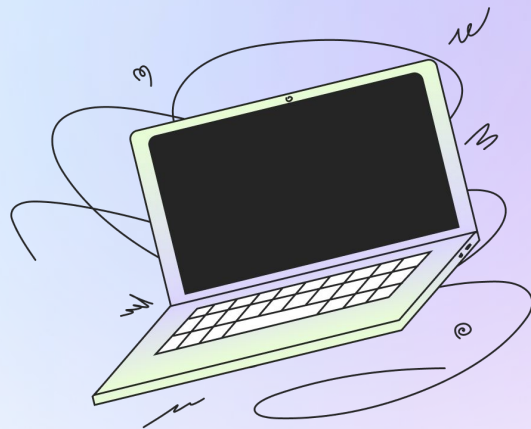
Перерыв

10 мин





Цикл While





Повторять можно по-разному

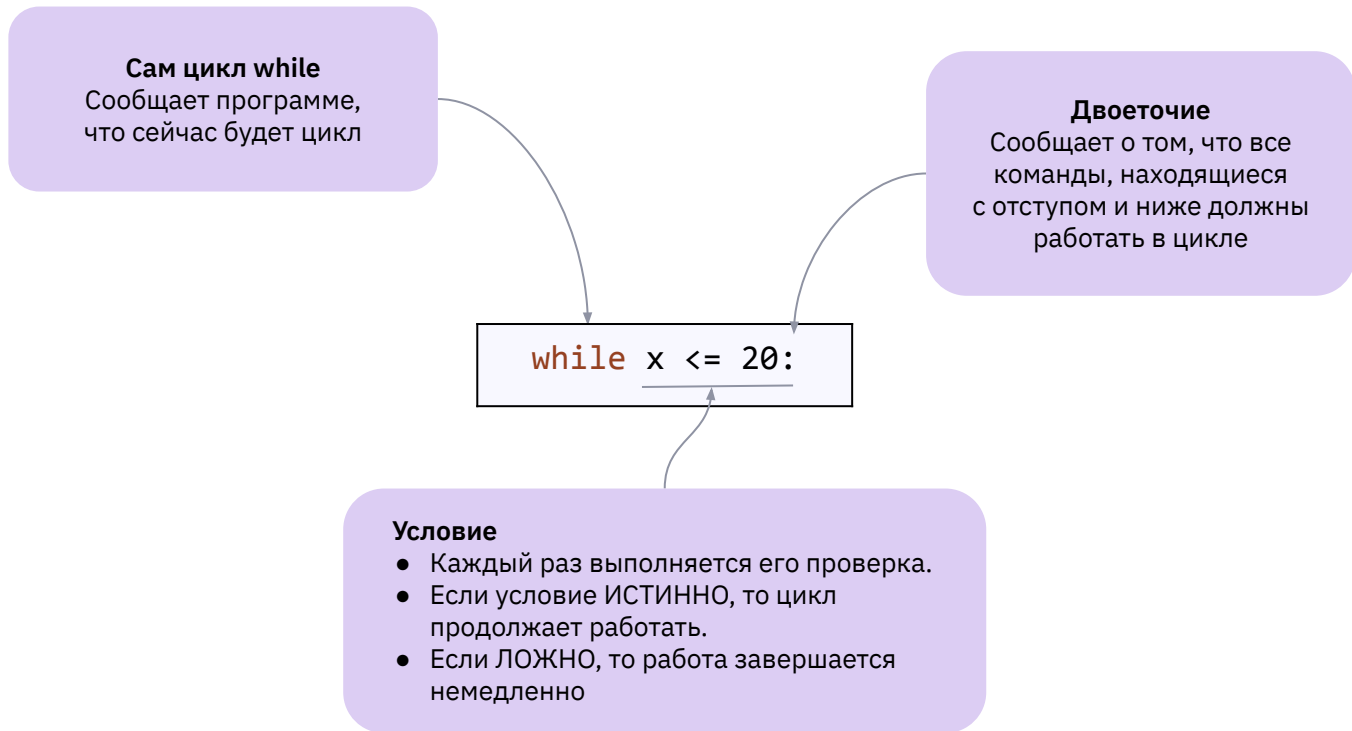
Цикл `for` — не единственный вид циклов в языке Python. Есть также цикл `while`, который используется, **если количество повторов заранее неизвестно**.

Структура цикла `while` выглядит так:

```
while <условие>:  
    Команда 1  
    Команда 2  
    ...  
    Команда N
```



Структура цикла WHILE





Вежливая программа

```
name = input('Как тебя зовут? ')
while name != 'выход':
    print(f'Привет, {name}!')
    name = input('Введите еще имя или "выход", чтобы выйти ')
```

Внутри цикла мы каждый раз будем просить ввести имя. Если это имя не будет равно слову `выход` (то есть наше условие цикла `while` будет в этом случае **истинно**) то цикл будет продолжать работать: здороваться и просить снова ввести имя.

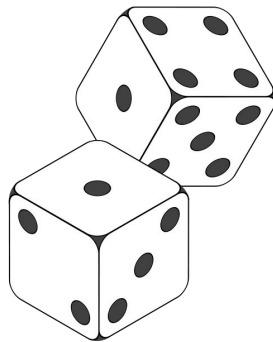
И так до тех пор, пока мы не введем слово `выход`. В этом случае условие `name != 'выход'` станет ложным и цикл завершит свою работу.



Подключаем случайность

Для следующего примера нам потребуется модуль, который создаёт случайные числа. Подключить мы его можем, так же как и черепашку, только имя у него другое — `random`.

```
import random
```





Генерируем случайные числа

Теперь, что создать случайное целое число, нам нужно воспользоваться функцией внутри `random` — `randint()` и задать диапазон в рамках которого хотим создать это число.

Например для чисел в диапазоне от 1 до 10 включительно напишем:

```
num1 = random.randint(1, 10)
```

Важно! Результат необходимо сохранить в переменную, иначе мы его потеряем. В этом примере результат сохраняется в переменную `num1`



Таблица умножения для первоклассников

У вас есть младшие брат или сестра? А может у ваших друзей есть? Давайте сделаем полезную программу, которая будет проверять знания таблицы умножения у первоклассников, а у вас знания циклов :)

X	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

```
import random
num1 = random.randint(1, 10)
num2 = random.randint(1, 10)
answer = int(input(f'Сколько будет {num1} * {num2} ?'))
while answer != num1 * num2:
    print('Неправильно! попробуйте еще раз!')
    answer = int(input(f'Сколько будет {num1} * {num2} ?'))
```

Генерируем первое случайное число
Генерируем второе случайное число
Пока ответ не будет правильным



Как работает таблица

Вся суть в цикле while, условие в котором проверяет правильность ответа:

```
answer != num1*num2
```

Если ответ, который ввел пользователь неправильный (цифра не равна произведению двух чисел) тогда заходим в цикл и выполняем команды, которые внутри цикла:

```
print('Неправильно! попробуйте еще раз!')  
answer = int(input(f'Сколько будет {num1} * {num2} ?'))
```

Именно внутри цикла мы снова просим пользователя ввести ответ еще раз. И сразу после введенный ответ попадает снова в переменную answer, а цикл снова выполняет ту же самую проверку.

И так до тех пор, пока не будет дан правильный ответ. Ведь только в этом случае условие будет ложным (**answer** будет равен произведению **num1** и **num2**)



Игра угадайка

Не останавливаемся на достигнутом. Усложним наш пример и создадим игру угадайка!

```
import random
num = random.randint(1, 100)      # Генерируем случайное число от 1 до 100
while True:                       # Цикл бесконечный
    answer = int(input('Какое число я загадал? '))
    if answer > num:
        print('Моё число меньше!')
    elif answer < num:
        print('Моё число больше!')
    else:
        print('Угадал!')
        break                     # Выходим из бесконечного цикла
```



Секрет игры

Здесь используется игровой бесконечный цикл, потому что условием его является True, то есть правда, которая никогда не станет ложью, а значит и наш цикл будет работать бесконечно

`while True:`

Как же все-таки мы выйдем из цикла? А все очень просто. Нам поможет в этом оператор break. Именно он осуществит выход из бесконечного цикла. Но сработает он только тогда, когда пользователь угадает число. То есть выполнится наша проверка:

`else:`

Подразумевающая, что пользователь ввел число не меньше и не больше загаданного, а именно равное загаданному.





Итоги

- Научились добавлять новые модули к нашей программе
- Узнали про циклы FOR и WHILE и в чем их отличие
- Создали игровой бесконечный цикл
- Изучили некоторые возможности рисования с помощью кода
- Написали программу - таблицу умножения
- Написали игру - «угадайка»





На следующем занятии:

- Узнаем про функции, и их назначение в коде
- Напишем код, который можно будет выполнять бесконечное количество раз всего одной строчкой
- Узнаем больше про встроенные функции в Python
- Напишем игру - «камень-ножницы-бумага»





**Немного
повторим**





С помощью какой команды можно подключить дополнительный модуль в программе?



**Можно ли задачу, которая решена циклом FOR
решить циклом WHILE?**



А наоборот? :)
**Можно ли задачу, которая решена циклом WHILE
решить циклом FOR?**



**Сколько повторений можно
задать в цикле FOR?**



Ваши вопросы





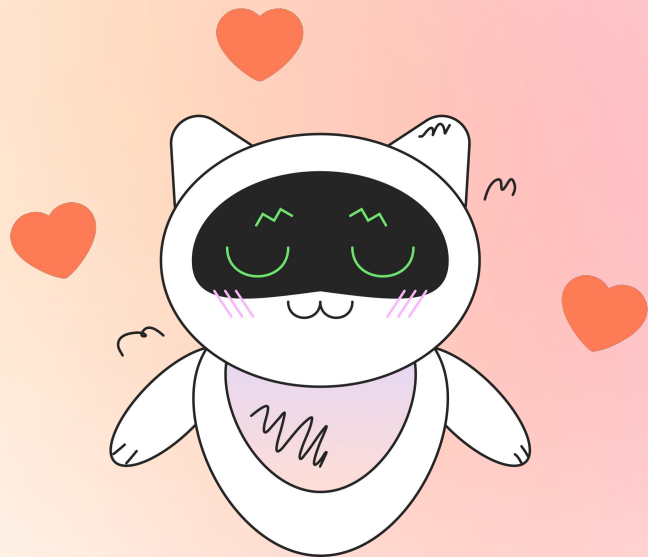
**Спасибо
за внимание**





Домашнее задание





Заполни, пожалуйста,
[форму обратной связи](#) по уроку



Напоминание для преподавателя

- Проверить заполнение Журнала
- Заполнить форму T22

