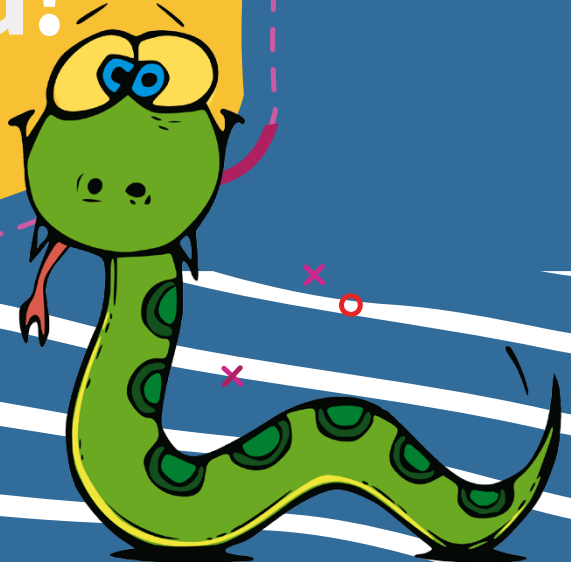


# PROGRAMMING WITH python

'Hello,  
world!'



# Lekcja # 3

## Grafika żółwia Pętla for

### Spis treści

<b>Zmienne .....</b>	<b>3</b>
<b>Pętle w programowaniu .....</b>	<b>5</b>
<b>Pętla for .....</b>	<b>5</b>
<b>Funkcja range().....</b>	<b>9</b>
<b>Rysowanie różnych kształtów.....</b>	<b>12</b>

# Zmienne

Python jest językiem **dynamicznym**, co oznacza, że **dane w programie mogą ulegać zmianie podczas jego wykonywania** i nie jest konieczne ich ręczne zmienianie dla każdej zmiennej.

Spójrzmy na przykład i dokończmy go:

```
a = 8
b = 0.8
c = "string"
d = True
print(type(a), type(b), type(c), type(d))
```

Stworzyliśmy różne zmienne i zdefiniowaliśmy ich typ. Aby sprawdzić typ, w Pythonie istnieje specjalna funkcja `type()`. Wynikiem tego kodu będzie:

```
<class 'int'> <class 'float'> <class 'str'>
<class 'bool'>
```

Są to 4 podstawowe typy danych, z którymi się już zapoznaliśmy. Odpowiednio: `int` (**integer**) jest używany dla liczb całkowitych (-765, 0, 20), `float` dla liczb zmiennoprzecinkowych (-15.0, 0.1, 225.5), `str` (**string**) dla tekstu ("Tomek", "Hello World") i `bool` dla zmiennych logicznych – np. czy warunek jest spełniony lub nie (`True/False`) (Rysunek 1).



Rysunek 1

W takiej formie lub innej zmienne będziemy używać w każdym programie, szczególnie w **pętlach** i **warunkach**.

# Pętle w programowaniu

Ludzie wciąż powtarzają pewne czynności i programiści nie są tu wyjątkiem. Niespecjalnie lubimy wykonywać dodatkową pracę, więc używamy **pętli** do powtarzania akcji.

**Pętla jest powtórzeniem konkretnego zbioru komend.** Na przykład jeśli człowiek robi 10 pompek, musi powtórzyć 10 razy jedną czynność.

**Wyróżniamy różne typy pętli:**

- ze znaną liczbą powtórzeń (z **parametrem**);
- bez znanej liczby powtórzeń (**warunek**).

Jeśli wrócimy do przykładu z pompkami, stwierdzenie “zrób 40 pompek” oznacza, że “pompka” powinna być w pętli, wykonanej 40 razy. Jeśli powiemy: “Rób pompki aż się zmęczysz”, będzie to pętla warunkowa, której ilości powtórzeń nie znamy.

## Pętla for

**Pętla for jest pętlą z parametrem (konkretna ilość powtórzeń).** Powinniśmy jej używać w następujących przypadkach:

- powtórzenie akcji określoną ilość razy;
- używaniu liczby, która zmienia się z każdym powtórzeniem.

Składnia pętli **for** wygląda następująco:

```
for i in sekwencja_powtorzen:
```

```
    instrukcje
```

Użyjmy żółwia do narysowania 3 figur (zobacz Rysunek 2 na stronie 7):

```
from turtle import*

window = Screen()
reset()

shape("turtle")
bgcolor("dark slate gray")
color("alice blue")

speed(2)
pensize(3)

left(20)

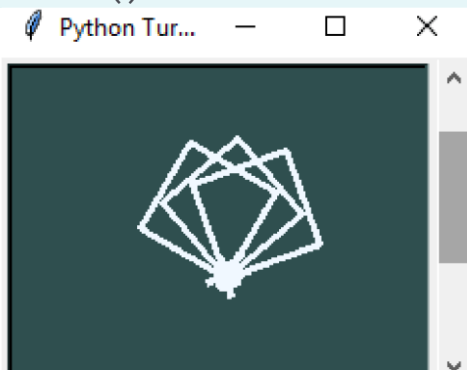
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
left(20)
forward(50)
```

```
left(90)
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
```

```
left(20)
```

```
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
forward(50)
left(90)
```

```
window.exitonclick()
```



Rysunek 2

Możesz zauważyć ile napisaliśmy kodu. A co jeśli potrzebowalbyś 100 kwadratów?

Nie jest najlepszą decyzją przepisywać wszystko kilka razy, prawda? Co więcej nie jest to nawet wygodne dla programistów. Dlatego z pomocą przychodzi pętla `for`.

Zobacz o ile bardziej jest czytelny ten kod:

```
from turtle import*

window = Screen()
reset()

shape("turtle")
bgcolor("dark slate gray") color("alice blue")
speed(4)
pensize(3)
for i in 1,2,3,4,5,6:
    left(30)
    forward(40)
    left(90)
    forward(40)
    left(90)
    forward(40)
    left(90)
    forward(40)
    left(90)

window.exitonclick()
```



W naszym kodzie użyliśmy konstrukcji:

```
for i in 1,2,3,4,5,6
```

Ta pętla powtórzy się 6 razy.

Lecz jeśli będziesz chciał wykonać 100 lub 200 powtórzeń, ta składnia nie będzie wygodna. Wyobraź sobie ile czasu byś na to poświęcił! Tu przychodzi z pomocą funkcja `range()`.

## Funkcja `range()`

Pętla `for` działa zgodnie z zasadą: przechodzi przez wszystkie wartości parametru i wykonuje kod znajdujący się za dwukropkiem.

**Funkcja `range()` zwraca serię numerów w obrębie podanego parametru.** Parametr podany w nawiasie określa liczbę wykonanych powtórzeń np. `(10)` spowoduje narysowanie 10 figur. Wartość w nawiasie sprawi, że komendy powtórzą się 10 razy! Funkcja `range()` może przyjąć więcej niż jeden parametr np. `(0, 10)` lub `(10, 20)`. W drugim przypadku funkcja przejdzie przez wartości od 10 do 19 i powtórzy się 10 razy, jak we wszystkich przykładach.

***Zapamiętaj:** możesz określić wielkość kroku. Na przykład `range(0, 12, 2)` zwróci poniższą sekwencję:*

*0,2,4,6,8,10, a krok może być nawet ujemny:*

```
range(10, 0, -1):
```

```
for i in range(10, 0, -1):  
    print(i)
```

Napiszmy kod w którym użyjemy pętli `for` i funkcji `range()`. Nauczmy żółwia rysować obraz składający się z kwadratów (zobacz Rysunek 3):

```
from turtle import*

window = Screen()
reset()

shape("turtle")
bgcolor("dark slate gray")
color("alice blue")

speed(4)
pensize(3)
for i in range(12):
    left(30)
    forward(40)
    left(90)
    forward(40)
    left(90)
    forward(40)
    left(90)
    forward(40)
    left(90)

window.exitonclick()
```



Rysunek 3

# Rysowanie różnych kształtów

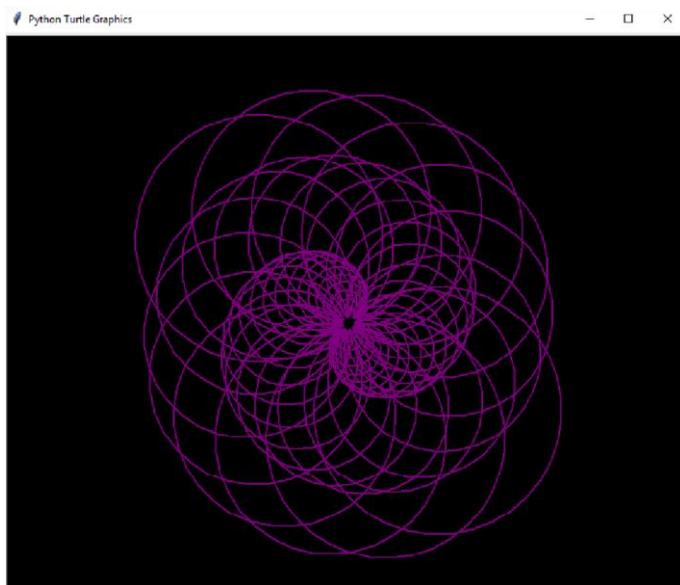
Jeśli dobrze zrozumiałeś poprzedni materiał, możesz teraz dokonać zmian w kodzie na własną rękę.

Użyjmy teraz pętli i żółwia aby narysować kształty składające się z okręgów (zobacz Rysunek 4 na stronie 14):

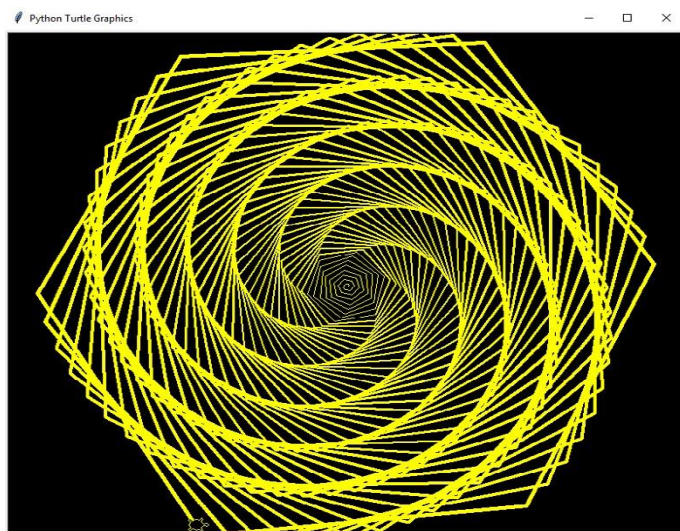
```
from turtle import*
window = Screen()
reset()
shape("turtle")
bgcolor("black")
pencolor("purple")
speed(10)
pensize(2)
for i in range(30):
    circle(5 * i)
    circle(-5 * i)
    left(i)
window.exitonclick()
```

Lub spiralę (zobacz Rysunek 5 na stronie 14):

```
from turtle import*
window = Screen()
reset()
shape("turtle")
bgcolor("black")
pencolor("yellow")
speed(10)
pensize(2)
for i in range(360):
    pensize(i/100 + 1)
    forward(i)
    left(59)
window.exitonclick()
```



Rysunek 4



Rysunek 5

Uzyskana wiedza pozwoli Ci tworzyć różnorodne figury i kształty oraz wykorzystywać parametry funkcji



## Lekcja # 3

# Grafika żółwia

### Pętla for

© STEP IT Academy

[www.itstep.org](http://www.itstep.org)

Wszelkie prawa do chronionych zdjęć, audio i wideo należą do ich autorów lub prawnych właścicieli. Fragmenty prac są wykorzystywane wyłącznie w celach ilustracyjnych w zakresie uzasadnionym celem w ramach procesu edukacyjnego oraz w celach edukacyjnych zgodnie z art. 1273 ust. 4 Kodeksu cywilnego Federacji Rosyjskiej oraz art. 21 i 23 Ustawy Ukrainy "O prawie autorskim i prawach pokrewnych". Zakres i metoda cytowanych prac są zgodne z normami, nie kolidują z normalnym wykorzystaniem utworu i nie naruszają uzasadnionych interesów autorów i podmiotów praw autorskich. Cytowane fragmenty utworów można zastąpić alternatywnymi, niechronionymi analogami i jako takie odpowiadają kryteriom dozwolonego użytku. Wszelkie prawa zastrzeżone. Wszelkie powielanie, w całości lub w części, jest zabronione. Zgoda na wykorzystanie utworów i ich fragmentów jest dokonywana z autorami i innymi właścicielami praw. Materiały z tego dokumentu mogą być używane tylko z linkiem do zasobów. Odpowiedzialność za nieuprawnione kopiowanie i komercyjne wykorzystanie materiałów określa się zgodnie z obowiązującym ustawodawstwem Ukrainy.