

Wstęp do programowania w języku Python

Python Turtle – grafika żółwia

KLASA 6

ŚCIĄGAWKA

Na początku programu MUSI być:

```
import turtle
```

```
t = turtle.Turtle()
```

(t to nazwa “naszego” żółwia)

Podstawowe ruchy żółwia

Podstawowe operacje, to przemieszczanie się naszego żółwia. Oto gotowe przykłady najważniejszych funkcji:

<code>t.fd(100)</code>	żółw t przesuwa się do przodu o 100 „kroków” (pikseli)
<code>t.bk(50)</code>	żółw t cofa się w tył o 50 „kroków”
<code>t.rt(90)</code>	żółw t obraca się w prawo o 90 stopni
<code>t.lt(45)</code> <code>t.home()</code>	żółw t obraca się w lewo o 45 stopni żółw t przesuwa się na środek okienka (bez obrotu, ciągle rysując)
<code>t.pu()</code>	„podnieś pisak” – powoduje, że żółw przemieszcza się bez rysowania
<code>t.pd()</code>	„opuść pisak” - żółw znowu rysuje

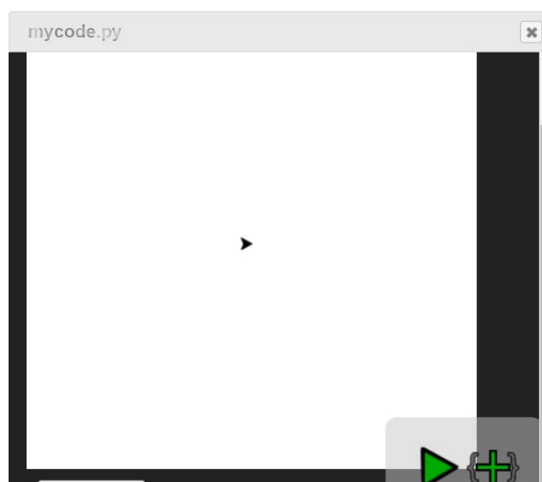
Kolory i rozmiar

<code>t.color('red')</code>	ustawienie czerwonego koloru dla linii, którą rysuje żółw
-----------------------------	---

	(jako x możemy podać takie kolory jak np. 'red','blue','green','yellow','black', 'white'
<code>t.pensize(8)</code>	ustawia grubość linii na 8 „pixeli”

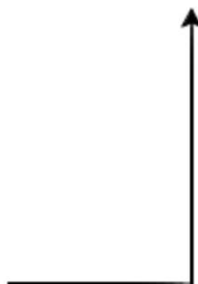
Lokalizacja żółwia

Przykład okienka dla żółwia - tutaj kwadrat 400x400 – żółw zawsze na początku jest w jego środku, „głową” skierowany w prawo.



Przykłady programów:

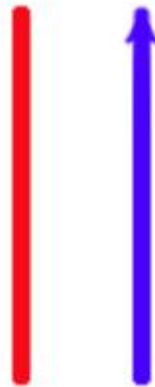
```
import turtle  
t=turtle.Turtle()  
t.fd(100)  
t.lt(90)  
t.fd(150)
```



```
import turtle
t=turtle.Turtle()
t.fd(150)
t.rt(120)
t.fd(50)
t.rt(60)
t.fd(100)
t.rt(60)
r.fd(50)
```



```
import turtle
t=turtle.Turtle()
t.lt(90)
t.color('red')
t.pensize(5)
t.fd(120)
t.bk(120)
t.rt(90)
t.pu()
t.fd(40)
t.pd()
t.lt(90)
t.color('blue')
t.fd(120)
```



Okręgi

Okrąg tworzymy takim rozkazem:

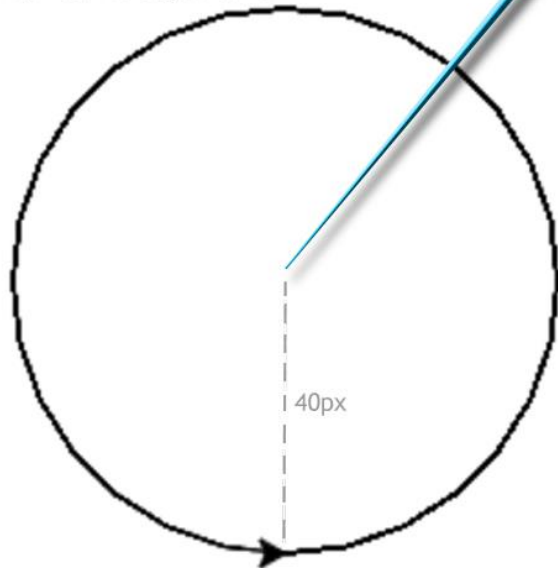
t.circle(40) w nawiasie podajemy promień okręgu
(w tym przykładzie to 40 pikseli)

Tak rysujemy okrąg cyrklem:
- wybieramy środek
- potem rysujemy okrąg



Tak rysuje okrąg żółw w Pythonie:
po rozkazie t.circle(40) żółw *tak jak stoi*,
tak zaczyna rysować okrąg (w lewo, patrząc z góry)

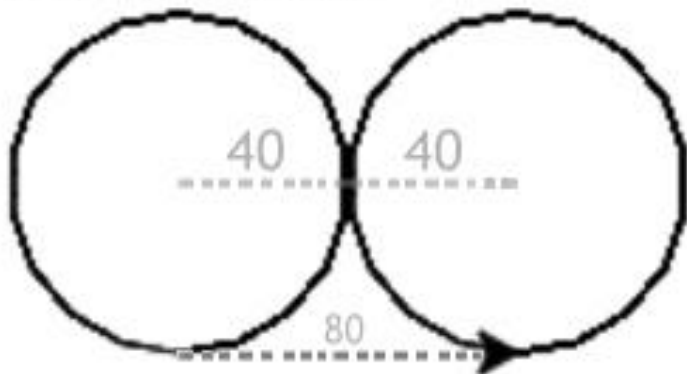
Musisz sobie więc sam (-a) wyobrazić, gdzie jest
środek okręgu (w tym przykładzie będzie gdzieś tutaj),
i że żółw przemieszcza się *pilnując* odległości
40 pikseli od tego środka.



Przykład programu:

```
import turtle  
t=turtle.Turtle()  
t.circle(40)  
t.pu()  
t.fd(80)  
t.pd()  
t.circle(40)
```

Dwa styczne okręgi o promieniu 40
(zauważ, że aby te okręgi stykały się,
to żółw po komendzie t.pu() musiał
przesunąć się w *powietrzu* o 80 pikseli
- czyli dwa promienie)




Losowanie liczby z danego przedziału:

najpierw po `import turtle` trzeba zaimportować nową bibliotekę:

import random

a w samym programie np. tak: `los=random.randrange(0,100)`
losuje liczbę naturalną z przedziału od 0 do 99 (100 jest tutaj *granicą*)

Przykład użycia:

Kolejne linie programu	Wyjaśnienie „co się dzieje”
<code>import turtle</code> <code>import random</code>	Wczytanie bibliotek: grafiki żółwia i losowania liczb
<code>t = turtle.Turtle()</code>	nasz żółw ma nazwę – to t
<code>los1 = random.randrange(0,100)</code>	losowanie liczby z przedziału od 0 do 99 (wylosowana liczba jest zapamiętywana pod nazwą los1)
<code>los2 = random.randrange(0,360)</code>	losowanie liczby z przedziału od 0 do 359 (wylosowana liczba jest zapamiętywana jako zmienna los2)
<code>t.fd(los)</code>	żółw rysuje odcinek o wylosowaną ilość pikseli
<code>t.rt(los2)</code>	żółw obraca się w prawo o wylosowany kąt Oto efekt – żółw przesunął się o wylosowane 82 piksele i obrócił o wylosowane 253 stopnie 

Podstawowe działania matematyczne

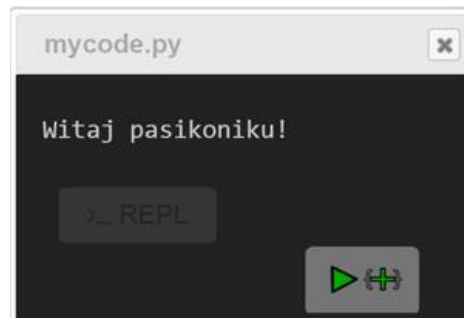
<code>a+b</code>	program dodaje liczby „ukryte” pod zmiennymi a i b
<code>a-b</code>	program odejmuje liczby „ukryte” pod zmiennymi a i b
<code>a*b</code>	program mnoży liczby „ukryte” pod zmiennymi a i b
<code>a/b</code>	program dzieli liczby „ukryte” pod zmiennymi a i b
<code>a**b</code>	a do potęgi b (np. 2 do potęgi 6 to w Pythonie <code>2**6</code>)
<code>int(a)</code>	część całkowita z liczby a (np. <code>int(3.47)</code> to 3)

Jak program „coś” napisze na ekranie?

```
print('Witaj pasikoniku!')
```

Tekst do wyświetlenia musi być w nawiasie, a przed nim i za nim muszą być apostrofy lub cudzysłowia.

Oto efekt działania tego programu:



Jak wprowadzić dane liczbowe?

Wprowadzając dane liczbowe musimy wiedzieć, czy będzie to liczba całkowita (bez przecinka) czy wymierna (z przecinkiem).

UWAGA! Python używa w liczbach kropek! (np. dwa i pół to 2.5)



Przykład Program, który zapyta jaki kąt ma narysować (w stopniach), po czym żółw narysuje ten kąt.

[Oto ten program:](#)

```
import turtle
t=turtle.Turtle()
kat=int(input('Podaj, ile stopni ma mieć narysowany kąt?'))
t.fd(180)
t.bk(180)
t.lt(kat)
t.fd(180)
```

Oto efekt działania tego programu:



Przykład Program, który narysuje prostokąt, ale najpierw zapyta:

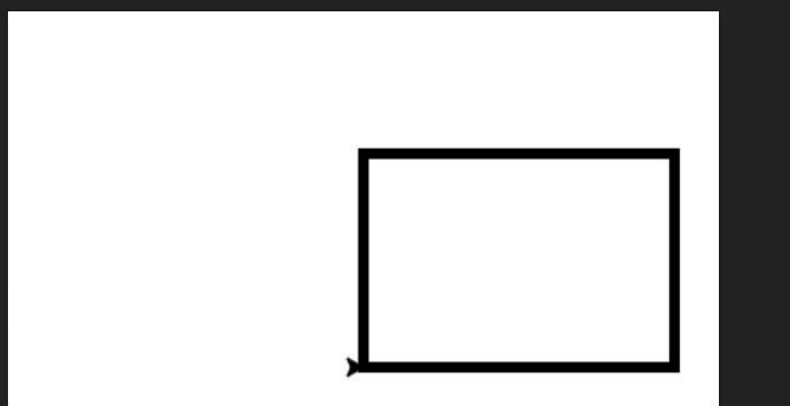
- jaka ma być grubość pisaka,
- jakiej długości ma być „poziomy” bok prostokąta,

- jakiej długości ma być „pionowy” bok prostokąta,
po czym żółw narysuje ten prostokąt.

[Oto ten program i przykład działania:](#)

```
1 import turtle
2 t=turtle.Turtle()
3 gruby=int(input('Jaką grubość ma mieć pisak?'))
4 a=int(input('Jaką długość ma mieć poziomy bok prostokąta?'))
5 b=int(input('Jaką długość ma mieć pionowy bok prostokąta?'))
6 t.pensize(gruby)
7 t.fd(a)
8 t.lt(90)
9 t.fd(b)
10 t.lt(90)
11 t.fd(a)
12 t.lt(90)
13 t.fd(b)
14 t.lt(90)
15
16
17
18
19
20
21
```

```
Jaką grubość ma mieć pisak? 6
Jaką długość ma mieć poziomy bok prostokąta? 175
Jaką długość ma mieć pionowy bok prostokąta? 120
```

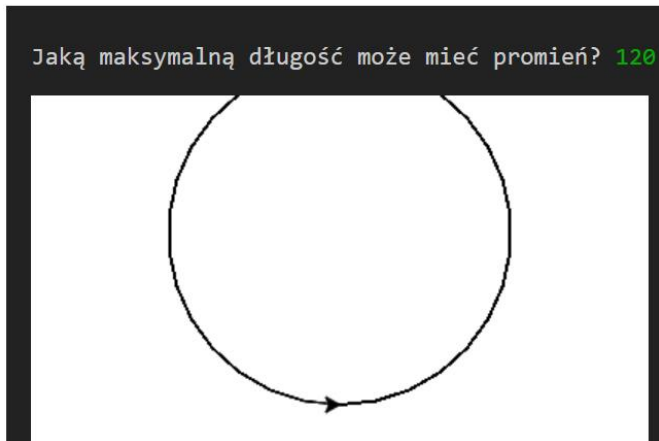


Przykład Program, który narysuje okrąg o wylosowanym przez niego promieniu, ale najpierw zapyta jaką maksymalną długość ma mieć losowany promień.

[Oto ten program i przykład jego działania:](#)

```
1 import turtle
2 import random
3 t=turtle.Turtle()
4 maxr=int(input('Jaką maksymalną długość może mieć promień?'))
5 r=random.randrange(0,maxr)
6 t.circle(r)
7
8
9
10
11
12
13
14
15
```

```
Jaką maksymalną długość może mieć promień? 120
```



Obliczenia w programie

Przypomnienie, jak w Pythonie zapisuje się działania matematyczne:

$a+b$	program dodaje liczby „ukryte” pod zmiennymi a i b
$a-b$	program odejmuje liczby „ukryte” pod zmiennymi a i b
$a*b$	program mnoży liczby „ukryte” pod zmiennymi a i b
a/b	program dzieli liczby „ukryte” pod zmiennymi a i b
$a**b$	a do potęgi b (np. 2 do potęgi 6 to w Pythonie $2**6$)
$int(a)$	część całkowita z liczby a (np. $int(3.47)$ to 3)

Programy komputerowe mogą wykonywać najróżniejsze obliczenia, ale **pamiętaj o nadawaniu własnych nazw** otrzymywanych z obliczeń wyników:

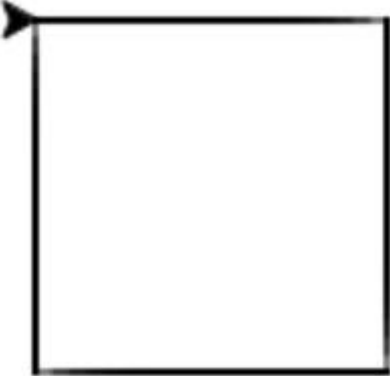
Przykład Program, który zapyta o jakąś liczbę, a potem narysuje odcinek o długości połowa z podanej liczby:

```
import turtle
t=turtle.Turtle()
a=int(input('Podaj jakąś liczbę'))
b=a/2
t.fd(b)
```

Przykład Program, który zapyta o jakąś liczbę, a potem narysuje odcinek o długości 3 razy dłuższej od podanej liczby:

```
import turtle
t=turtle.Turtle()
a=int(input('Podaj jakąś liczbę'))
b=3*a
t.fd(b)
```

Pętla for: (najlepiej zobacz przykład)

Program	Wyjaśnienie „co się dzieje”
<pre>import turtle t=turtle.Turtle() for i in range (0,4): t.fd(100) t.rt(90)</pre>	<p>Program będzie 4 razy powtarzał pewną pętlę.</p> <p>Zmienna i (licznik) może nazywać się inaczej.</p> <p>To co ma być wykonane wielokrotnie (w pętli) jest wcięte (odsunięte od lewego marginesu).</p> <p>Co będzie powtarzane 4 razy? <i>Naprzód 100</i> czyli <code>t.fd(100)</code> i <i>Obróć się w prawo o 90 stopni</i>, czyli <code>t.rt(90)</code></p> <p>Efekt działania programu:</p> 

Czyli pętla for ma taką „budowę”:

Program wykona 4 razy pętlę: zółw do przodu o 100 i obrót w prawo o 90 stopni, a na końcu napisze wyraz Kwadrat.

```
import turtle
t = turtle.Turtle()
for i in range (0,4):
    t.forward(100)
    t.right(90)
t.write('Kwadrat')
```

Po polsku "Od "

Licznik (można wymyśleć inną nazwę)

Po polsku "w zakresie"

Ilość powtórzeń pętli (od 0 do 3 czyli 4 razy)

Kwadrat

Rozkazy w pętli muszą mieć wcięcie (od lewego marginesu)

Pierwszy rozkaz przy lewym marginesie oznacza koniec pętli

Efekt działania programu

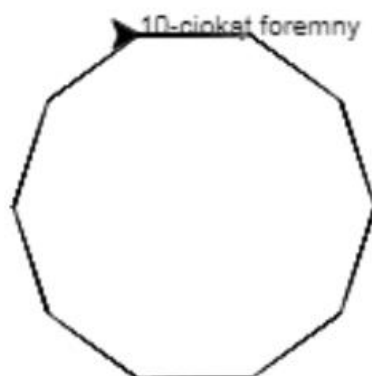
Jeszcze dwa przykłady:

Przykład 1 - rysowanie 10-ciokąta foremnego o boku 40

Pętla powtórzy się 10 razy: forward 40 i right 36 (dlaczego 36? Bo $360:10 = 36$ stopni)

```
import turtle
t = turtle.Turtle()

for i in range (0,10):
    t.forward(40)
    t.right(36)
t.write('10-ciokąt foremny')
```



Przykład 2 – program będzie w pętli 500 razy losował ilość pikseli „do przodu” (z przedziału od 0 do 20) i kąt, o jaki obróci się żółw (z przedziału od 0 do 360), a następnie o tyle co wylosuje, będzie przesuwiał i obracał żółwia.

```
import turtle
import random
t = turtle.Turtle()
t.speed(0)
for i in range (0,500):
    los=random.randrange(0,20)
    los2=random.randrange(0,360)
    t.forward(los)
    t.right(los2)
t.write('Koniec')
```

