Python Turtle – grafika żółwia

Python Turtle, jest to bardzo popularny sposób nauki podstawowych operacji w Python. Pozwala ona w prosty sposób pisać aplikacje graficzne i mieć przy tej okazji, sporo radości. Włączając tworzenie ciekawych grafik, czy też bardzo prostych gier.

Instalacja oraz uruchomienie turtle

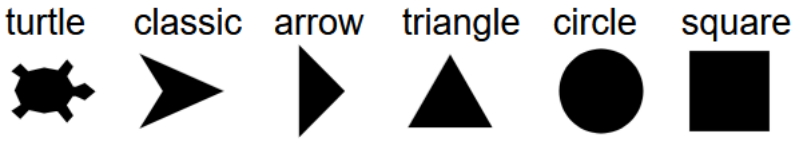
Standardowo biblioteka jest już dołączona do Python 3, tak więc wystarczy ją zaimportować:

```
import turtle
```

Pierwszy program w Python turtle

Stworzenie pierwszego programu, jest bardzo proste. Wystarczy utworzyć obiekt żółwia, a następnie uruchomi się okienko, natomiast na jego środku, będzie bohater biblioteki, czyli żółw. Domyślenie ma on kształt strzałki, jednak możemy w łatwy sposób nadać mu faktyczny kształt.

W celu zmiany postaci należy na żółwiu wykonać polecenie **shape()**, a jako argument podać jeden z kodów wbudowanych postaci. Jeden z nich *turtle* pozwala na wyświetlenie postaci żółwia zamiast standardowego znaczka. Oto pełna lista wbudowanych postaci oraz ich kodów:



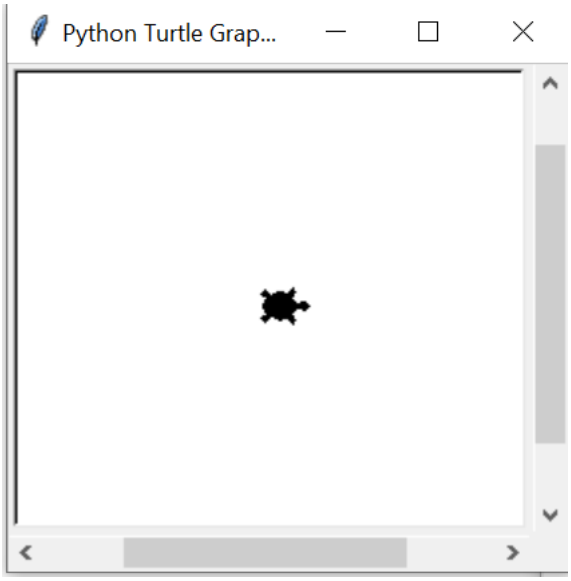
Przykładowo, aby zmienić strzałkę na żółwia należy wpisać:

żółw.shape("turtle")

Przykład 1

```
import turtle
żółw = turtle.Turtle()
żółw.shape('turtle')
turtle.exitonclick()
```

a następnie, po uruchomieniu programu, zobaczymy następujące okienko.



funkcja **exitonclick(),** umieszczona na końcu, powoduje że okienko się nie zamknie, dopóki nie klikniemy.

Podstawowe ruchy turtle

Podstawowe operacje w turtle, to przemieszczanie się naszego żółwia. Mamy do dyspozycji dość sporą liczbę funkcji. Najważniejsze to:

forward(x) lub fr(x)	do przodu, o podany dystans x
backward(x) lub bc(x)	do tyłu, o podany dystans x
right(x) lub rt(x), left(x) lub lf(x)	obróć się w prawo / lewo, o podany kąt x
goto(x,y)	pójdź do konkretnego punktu (x,y)
home()	wróć na środek ekranu
circle(x)	narysuj koło o podanym promieniu

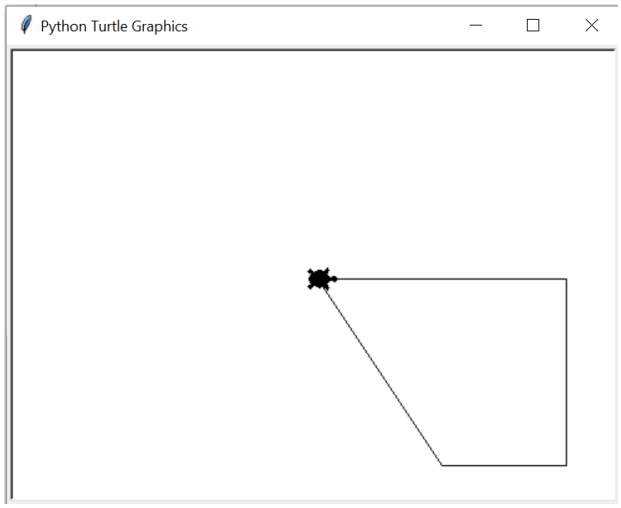
Przykład 2

W rezultacie, po dodaniu następujących linijek do naszego programu:

```
import turtle
```

```
żółw = turtle.Turtle()
żółw.shape('turtle')
żółw.forward(200)
żółw.right(90)
żółw.forward(150)
żółw.right(90)
żółw.forward(100)
żółw.home()
turtle.exitonclick()
```

Otrzymamy następującą figurę:

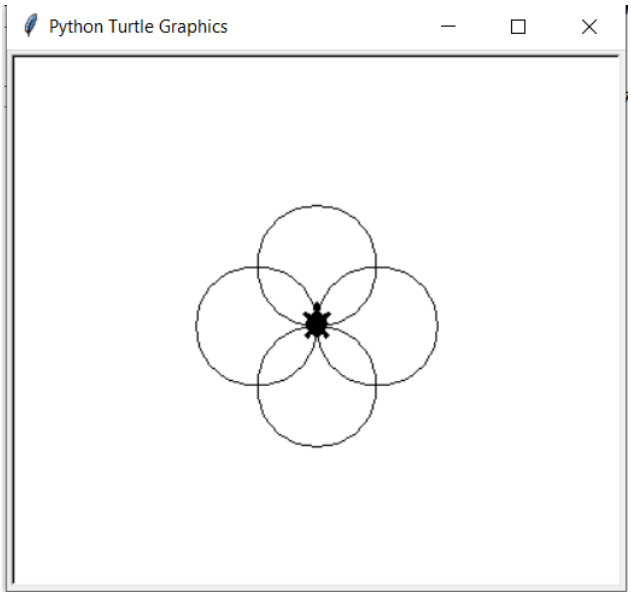


W analogiczny sposób, możemy rysować dowolnie złożone figury. Dodatkowo, warto zauważyć, że funkcje mają również swoje nazwy skrócone.

I jeszcze kilka kółek, aby pokazać co nasz żółw potrafi:

Przykład 3

```
import turtle
żółw = turtle.Turtle()
żółw.shape('turtle')
żółw.circle(40)
żółw.right(90)
żółw.circle(40)
żółw.right(90)
żółw.circle(40)
żółw.right(90)
żółw.circle(40)
turtle.exitonclick()
```



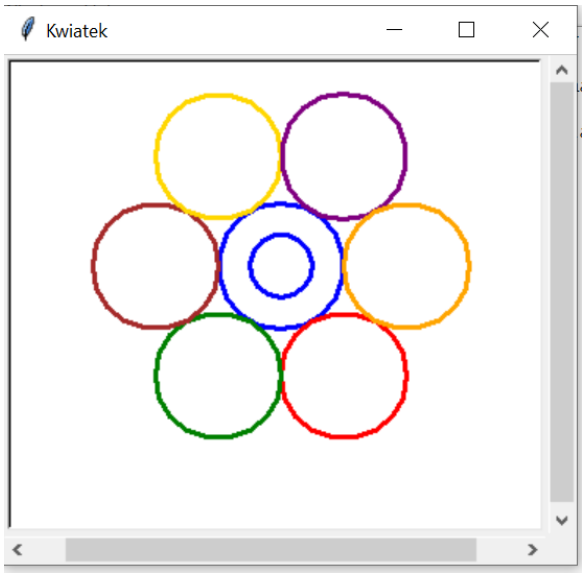
Kolory, rozmiar oraz prędkość

Jeżeli umiemy już poruszać się naszym żółwiem, możemy uatrakcyjnić jego wygląd, za pomocą kilku prostych funkcji:

turtle.bgcolor(x)	zmienia kolor tła. Jak x możemy podać takie kolory jak 'red','blue','green','yellow','black', 'white' i wiele innych. Możemy również podać kolor w RGB (x,y,z)
color(x)	zmienia kolor naszego żółwia i linii które rysuje
pensize(x)	ustawia grubość linii
penup()	powoduje że żółw przemieszcza się bez rysowania
pendown()	żółw znowu rysuje
hideturtle() oraz showturtle()	żółw znika / pojawia się
speed(x)	ustawia prędkość żółwia
write(tekst)	żółw pisze

Zadanie 1

Wykorzystując przykład 3, napisz program wykonujący poniższy rysunek:



Ukrywanie żółwia.

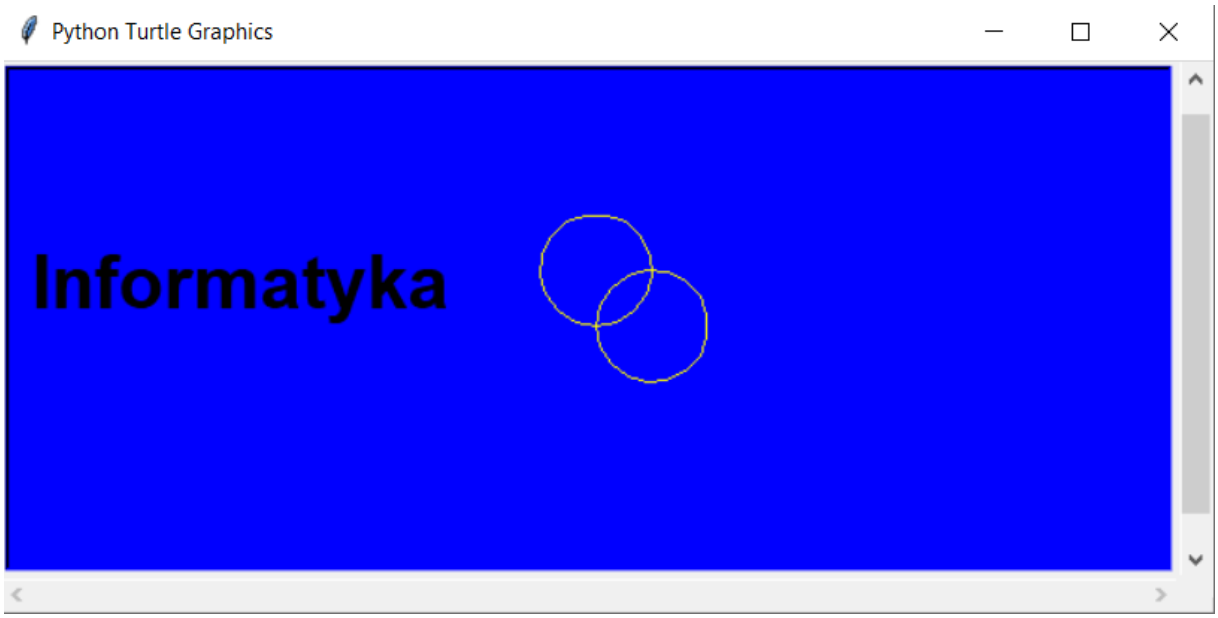
Można to zrobić przy pomocy polecenia **hideturtle()**, a żeby pokazać ponownie wystarczy **showturtle()**. Niezależnie od widoczności żółwia nadal jest możliwe rysowanie nim po ekranie czy zmienianie jego właściwości. Po ponownym jego pokazaniu wszystkie zmiany będą widoczne.

I jeżeli połączymy kilka funkcji widocznych w powyższej tabeli w jeden program:

Przykład 4

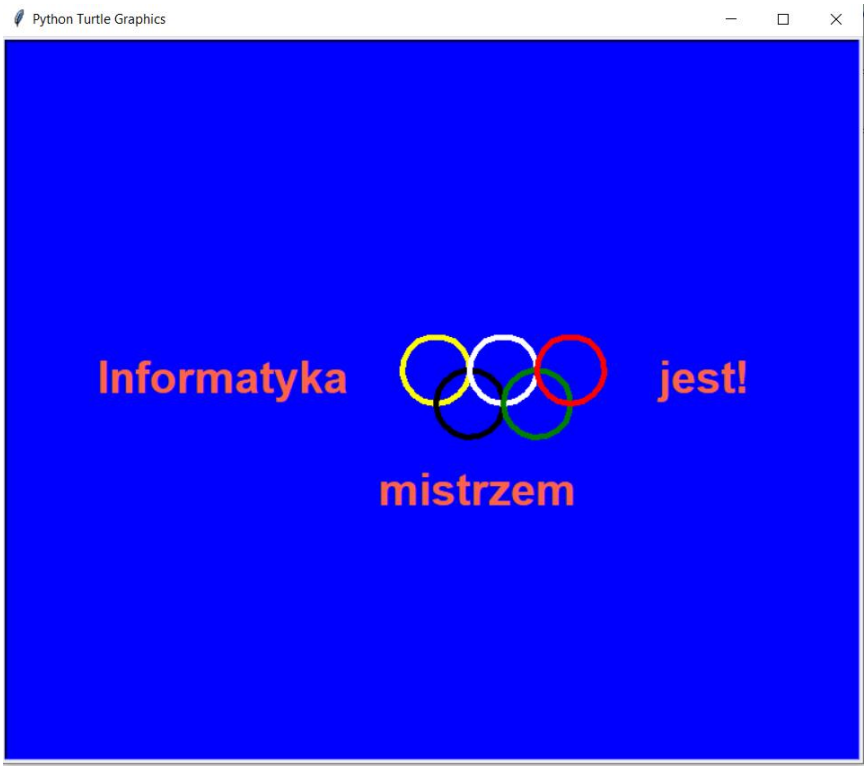
```
import turtle

turtle.bgcolor('blue')

zolw = turtle.Turtle()
zolw.shape('turtle')
zolw.color ('yellow')
zolw.circle(30)
zolw.right(90)
zolw.circle(30)

zolw.penup()
zolw.goto(-300,0)
zolw.pendown()

zolw.color('black')
zolw.write('Informatyka', font=("Comic Sanserif",30,"bold"))
zolw.hideturtle()
```

```
turtle.exitonclick()
```



Zadanie 2

Wykorzystując przykład 4, napisz program wykonujący poniższy rysunek:

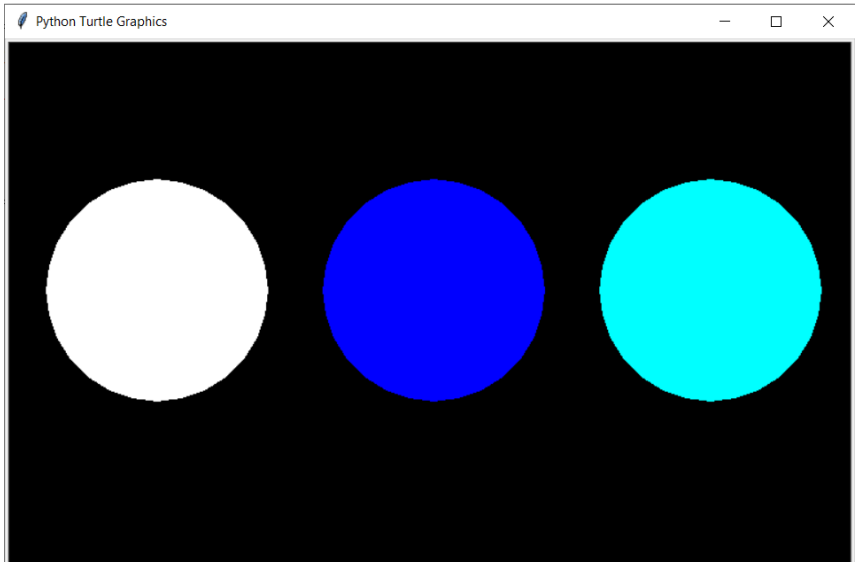


Wypełnianie kolorem w turtle

Biblioteke turtle, oferuje również prosty sposób wypełniania kształtów kolorem. Wystarczy, że przed rozpoczęciem rysowania figury, wywołamy funkcję **begin_fill()**, a po zakończeniu **end_fill()**, i narysowana przez nas figura się wypełni.

Przykład 5

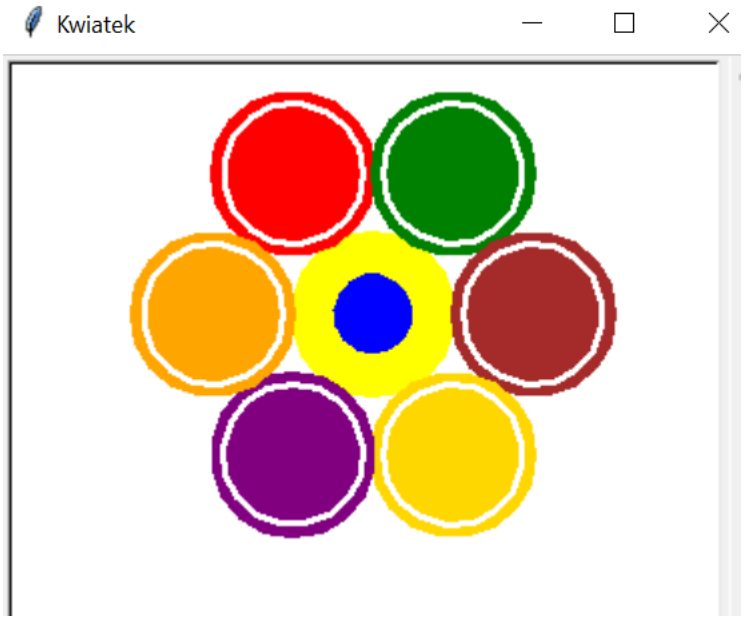
```
import turtle
turtle.bgcolor('black')
zolw = turtle.Turtle()
zolw.shape('turtle')

zolw.color(0,0,1)
zolw.begin_fill()
zolw.circle(100)
zolw.penup()
zolw.end_fill()

zolw.forward(250)
zolw.pendown()
zolw.color(0,1,1)
zolw.begin_fill()
zolw.circle(100)
zolw.penup()
zolw.end_fill()

zolw.back(500)
zolw.color(1,1,1)
zolw.pendown()
zolw.begin_fill()
zolw.circle(100)
zolw.penup()
zolw.end_fill()
zolw.hideturtle()

turtle.exitonclick()
```



Zadanie 3

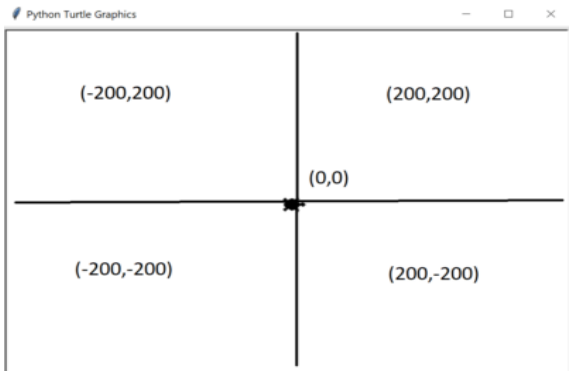
Na podstawie przykładów 3 i 4 i 5 popraw zadanie 1 aby otrzymać poniższy rysunek:

Możesz oszczędzić sobie pracy, definiując funkcje np. rysujące koła. (Tworzenie własnych funkcji opisałem poniżej)



Lokalizacja żółwia

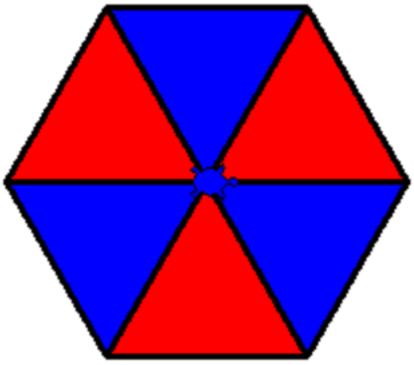
Nasz żółw startuje z punktu (0,0). Jest to środek planszy. Jeżeli chcemy go przemieszczać w różnych kierunkach, przy użyciu współrzędnych, to wykonujemy to w analogiczny sposób jak znany z geometrii:



Dodatkowo, w każdym momencie, możemy uzyskać położenie naszego żółwia, za pomocą funkcji ***position()*.**

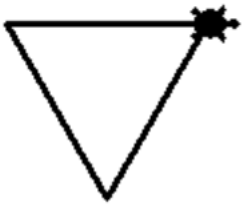
Zadanie 4

Wykonaj następująca figurę:



Dla realizacji tego programu, możemy stworzyć funkcje własne pracujące na rzecz funkcji głównej. Np. funkcję rysującą trójkąt

```
def trojkat():
    for i in range(3):
        turtle.right(120)
        turtle.forward(100)
```



def - definiujemy funkcję o nazwie „trojkat”, zwracając uwagę na dwukropek na zakończenie deklaracji.

Funkcja to fragment kodu, który ma wykonać określone zadanie i którego można wielokrotnie używać w programie. W programach możesz korzystać ze standardowych funkcji języka programowania (np. funkcja print()), z własnych funkcji, a także z funkcji napisanych przez innych programistów.

Tworzenie funkcji w Pythonie zaczyna się od słowa kluczowego **def**. Po nim określa się **nazwę funkcji**, następnie umieszcza parę nawiasów **( )** oraz dwukropek **:**

Zestaw instrukcji, który ma wykonywać funkcja umieszcza się w kolejnych liniach, z przesunięciem w prawo, czyli z **wcięciem**. Prosta funkcja może wyglądać np. tak

```
def wyswietl():
    print( "Hi there!" )
```

W ten sposób utworzyliśmy funkcję o nazwie **wyswietl()**, której zadaniem jest wyświetlenie komunikatu Hi there!

Wywoływanie funkcji

Efekt działania funkcji jest widoczny dopiero po jej użyciu, czyli wywołaniu. Wywołanie funkcji polega na podaniu jej nazwy i ewentualnie argumentów, np.:

```
def wyswietl():
    print( "Hi there!")

wyswietl()
```

```
Hi there!
```

Pętla for umożliwia iterowanie po elementach sekwencji, tzn. przechodzenie po kolejnych elementach sekwencji.

Działanie pętli for zobacz na przykładzie:

```
for x in "ABC":
   print( x )
```

```
A
B
C
```

W nagłówki pętli po słowie kluczowym for deklarujemy zmienną pomocniczą, do której, w każdej iteracji pętli przypisywany jest kolejny element sekwencji, którą umieszczamy po operatorze in. Zwróć uwagę także na dwukropek, po którym w nowej linii z wcięciem rozpoczyna się ciało pętli - czyli instrukcje wykonywane w każdej iteracji.

Jak widzisz w powyższym przykładzie, do zmiennej pomocniczej x przypisywane są kolejne znaki łańcucha "ABC", a w ciele pętli są one kolejno wyświetlane.

Funkcja range tworzy sekwencję liczb w podanym zakresie. Domyślnie pierwszą liczbą jest 0, a kolejne liczby zwiększane są o 1.

Np. W pętli **for** iteracja wykonana jest w zakresie **range** (3) – czyli wykonuje się trzykrotnie, zwróci nam 0,1,2. **range(50, 90):** - zwróci nam liczby 50,51,52…89.

Funkacja range(2,9,2) – zwróci liczby 2,4,6,8.

Poniżej przedstawiam propozycję fragmentu kodu, do zadania 4:

Przykład 6

```
import turtle
turtle.shape("turtle")
turtle.pensize(3)

def trojkat():
    for i in range(3):
        turtle.right(120)
        turtle.forward(100)

trojkat()

for i in range (3):
    turtle.right(60)
    turtle.begin_fill()
    turtle.fillcolor("red")
    trojkat()
    turtle.end_fill()

turtle.mainloop()
```

Wiele żółwi

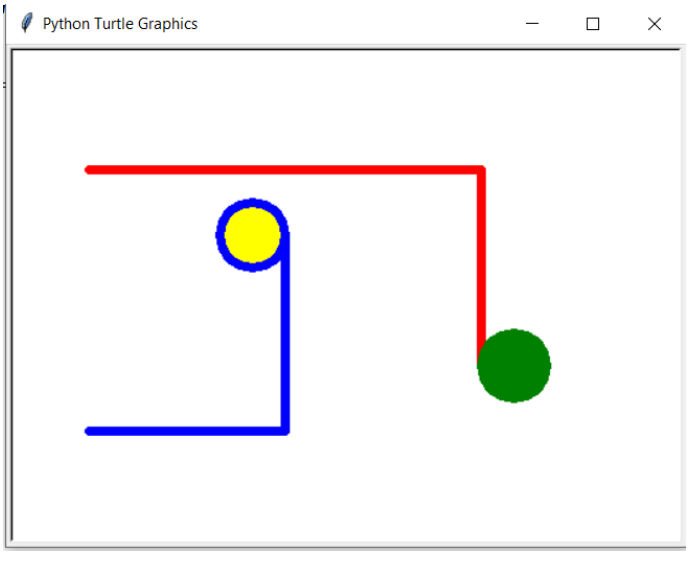
Naszego żółwia, możemy skopiować, za pomocą funkcji clone(), jak i również możemy utworzyć kolejne żółwie za pomocą turtle.Turtle().

W ten sposób możemy tworzyć grafikę, w kilku miejscach planszy, niezależnie.

Przykład 7

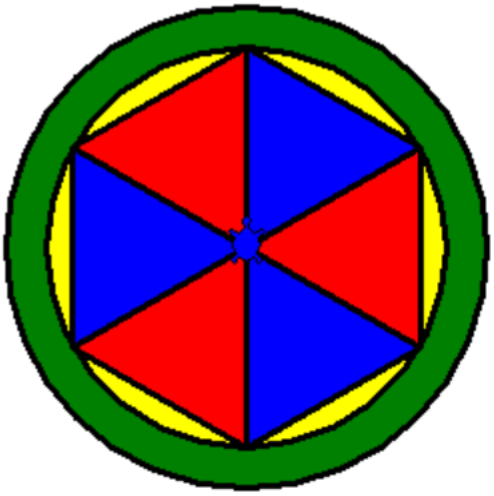
```
import turtle

zolw = turtle.Turtle()
zolw.shape('turtle')
zolw.color('red')
zolw.pensize(7)
```

```
zolw.penup()
zolw.goto(-200,100)
zolw.pendown()
zolw.forward(300)
zolw.right(90)
zolw.forward(150)
zolw.color('green')
zolw.begin_fill()
zolw.circle(25)
zolw.end_fill()
zolw.hideturtle()

zolw2 = turtle.Turtle()
zolw2.shape('turtle')
zolw2.color('blue')
zolw2.pensize(7)
zolw2.penup()
zolw2.goto(-200,-100)
zolw2.pendown()
zolw2.forward(150)
zolw2.left(90)
zolw2.forward(150)
zolw2.begin_fill()
zolw2.color('yellow')
zolw2.circle(25)
zolw2.end_fill()
zolw2.color('blue')
zolw2.circle(25)
zolw2.hideturtle()

turtle.exitonclick()
```



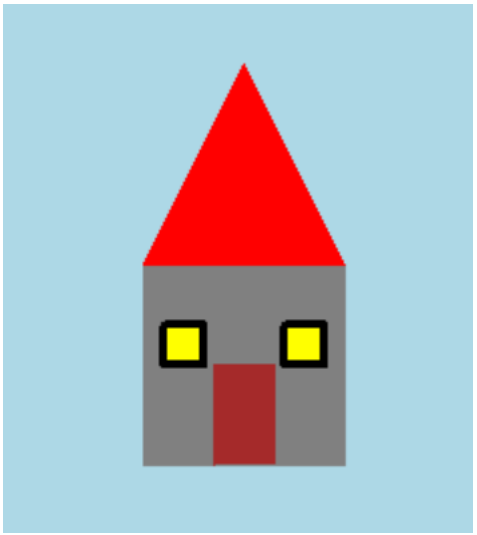
Zadanie 5

Na podstawie zadania 4 wykonaj następująca figurę:



Zadanie 6

Wykonaj rysunek domku:



Figury złożone w Python turtle

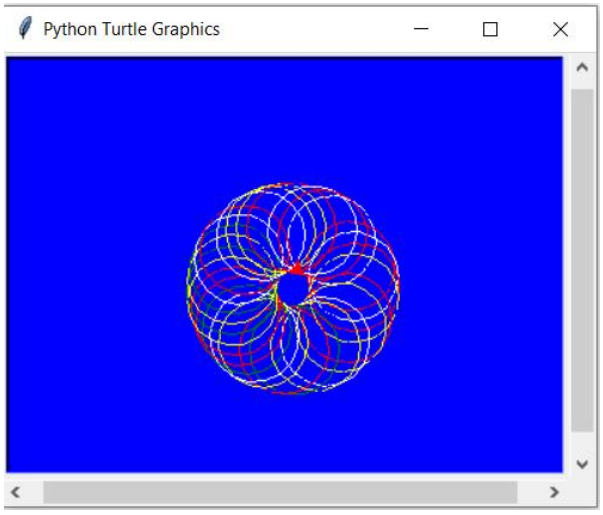
Używając powyższej wiedzy, oraz odrobinę pętli, możemy rysować bardzo abstrakcyjne kształty. W poniższym kodzie, malujemy tło na czarno, następnie, ustawiamy prędkość żółwia na maksymalną i w pętli, rysujemy:

- losujemy kolor linii;
- rysujemy żółwiem koło;
- obracamy żółwia o 30 stopni;
- przesuwamy i obracamy żółwia;
- powtarzamy pętlę.

Przykład 8

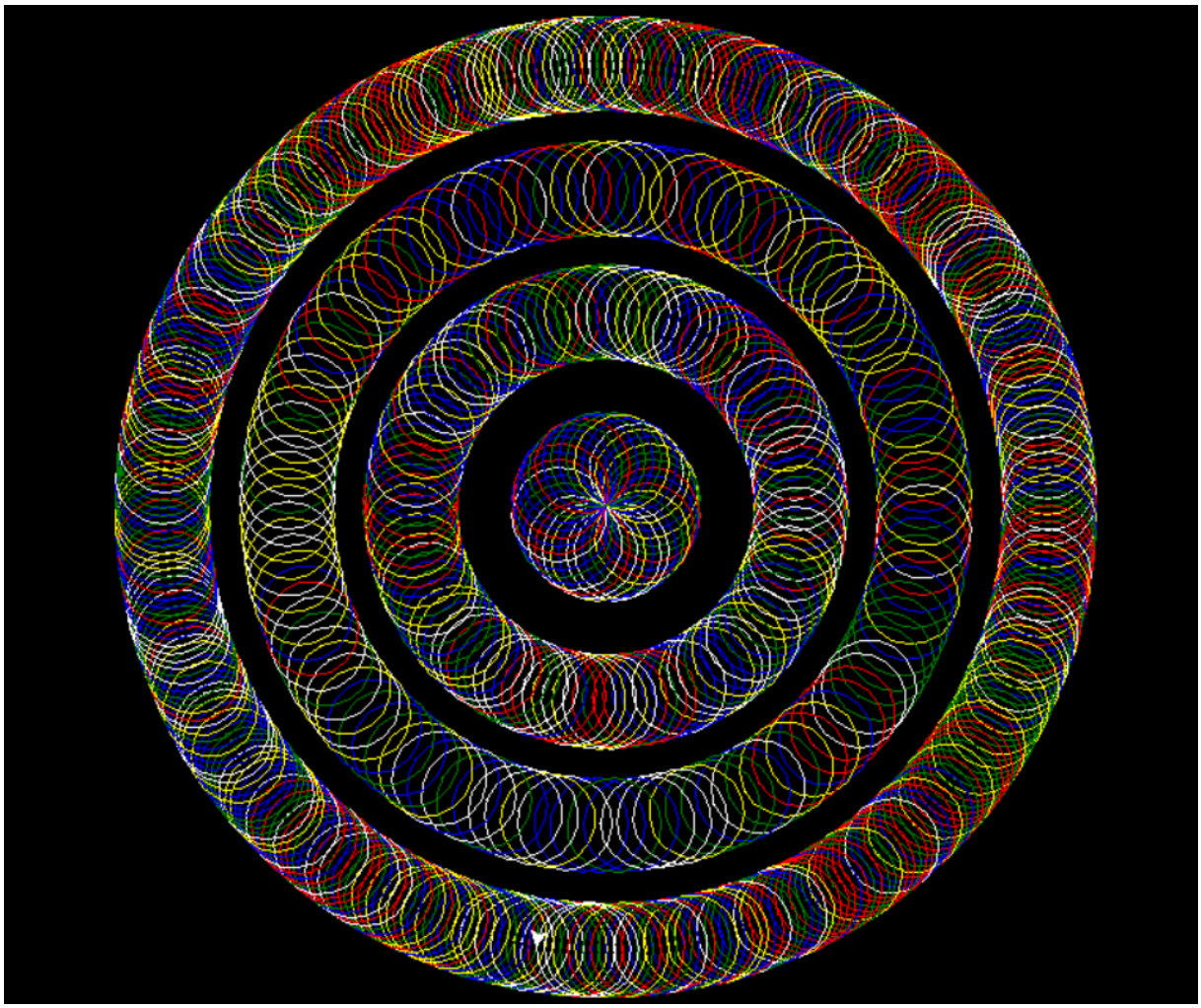
```
import turtle
import random
turtle.bgcolor(0,0,1)
zolw = turtle.Turtle()
zolw.speed(0)

for x in range(50, 90):
    zolw.color(random.choice(['white','red','blue','green','yellow']))
    zolw.circle(30)
    zolw.forward(2)
    zolw.right(10)

turtle.exitonclick()
```

W przykładzie użyliśmy funkcji losowej random.choice. Dzięki dołączonej bibliotece import random losujemy z zakresu 5 wybranych kolorów. Metoda choice() zwraca listę z losowo wybranym elementem z określonej sekwencji.



Zadanie 7

Wykorzystując przykład 8, napisz program wykonujący poniższy rysunek:



Obsługa zdarzeń

Do tej pory pokazywaliśmy podstawowe operacje w Turtle, które umożliwiały nam rysowanie różnych kształtów. Tym razem zaczniemy obsługiwać zdarzenia. Na przykład poprzez naciśniecie klawisza lub kliknięcie myszki. Tym samym nasze aplikacji będą mogły wchodzi w interakcję z użytkownikiem. Czyli wykonanie odpowiedniej akcji. Pozwala to na pisanie interaktywnych programów.

Podstawową funkcją, którą trzeba użyć jest **turtle.onkey** (funkcja, zdarzenie). Pierwszy parametr, to funkcja która ma zostać wykonana, w momencie, kiedy nastąpi zdarzenie opisane przez drugi parametr.

Następnie na końcu naszego programu, należy poinformować Python turtle, że ma nasłuchiwać na zdarzenia (listen), oraz całość ma być powtarzana (mainloop- funkcja główna w pętli).

```
turtle.listen()

turtle.mainloop()
```

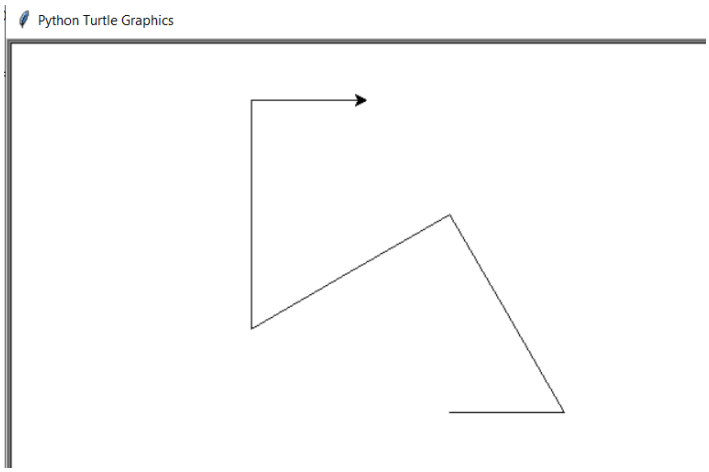
Jego zadaniem, jest nasłuchiwanie na 4 klawisze – strzałka do góry, w lewo, w prawo oraz klawisz 'q':

- strzałka do góry, przemieszcza żółwia o 100
- strzałka w prawo, obraca go w prawo o 30 stopni
- strzałka w lewo, obraca go w lewo o 30 stopni
- 'q' kończy działanie naszego programu

W tym celu, zdefiniowaliśmy 4 funkcje, które wykonują odpowiednie akcje.

Przykład 9

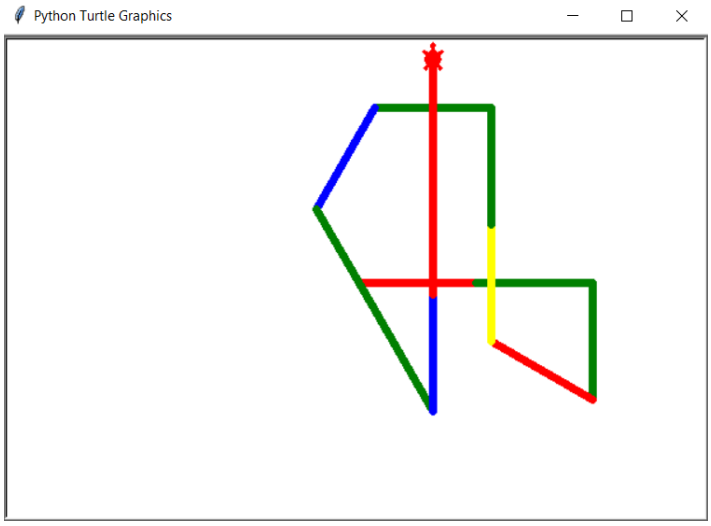
```
import turtle
t = turtle.Turtle()
def up():
    t.forward(100)
def left():
    t.left(30)
def right():
    t.right(30)
def bye():
    turtle.bye()
turtle.onkey(up,'Up')
turtle.onkey(left,'Left')
turtle.onkey(right,'Right')
turtle.onkey(bye,'q')
turtle.listen()
turtle.mainloop()
```



Aby program był atrakcyjniejszy, zróbmy tak aby przy każdym ruchu, żółw losował swój kolor.

Przykład 10

```
import turtle
import random
window = turtle.Screen()
t = turtle.Turtle()
t.pensize(7)
t.shape('turtle')
def up():
    t.color(random.choice(['red','blue','yellow','green']))
    t.forward(100)
def left():
    t.left(30)
def right():
    t.right(30)
def bye():
    turtle.bye()
def click(x,y):
    t.color(random.choice(['red','blue','yellow','green']))
    t.goto(x,y)

window.onkey(up,'Up')     # strzałka w górę
window.onkey(left,'Left')   # strzałka w lewo
window.onkey(right,'Right')
window.onkey(bye,'q')
window.onclick(click)
window.listen()
window.mainloop()
```



Przykład 11

Przypisanie zdarzenia do klawisza

Najprostszy kod, który obsługuje naciśnięcie przez użytkownika klawisza, wygląda tak:

```
import turtle

window = turtle.Screen()
t = turtle.Turtle()

def circle():
t.circle(90)

window.onkey(circle,'c')

window.listen()
window.mainloop()
```

Najważniejsza jest funkcja window.onkey(). Jako drugi parametr przyjmuje zdarzenie które chcemy obsłużyć, czyli jaki klawisz nas interesuje. W tym przypadku jest to klawisz c.

W momencie kiedy użytkownik naciśnie klawisz c, wykona się funkcja circle().

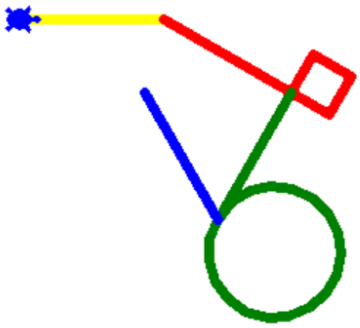
Na początku kodu, po imporcie biblioteki turtle, tworzymy obiekt **window = turtle.Screen()**

Na końcu kodu**,** wykonujemy jeszcze 2 ważne funkcje window.listen() oraz window.mainloop()

Zadanie 8

Wykorzystując przykłady 10 i 11, napisz program w którym zdefiniujesz funkcje:

- rysującą koło;
- rysującą kwadrat;
- wykonującą powrót kolejno o jeden ruch;
- wykonującą powrót do początku rysowania;



Przykładowe gry:

Turtle to jedna z pierwszych bibliotek w Python. Pozwala tworzyć proste programy graficzne, w tym bardzo, bardzo proste gry. Wyścig żółwi jest klasycznym przykładem i grą polecaną jako jeden z pierwszych projektów, które warto zrobić. Pozwala ona przećwiczyć podstawowe operacje w Python, takie jak pętle, funkcje czy też wyrażenia warunkowe.

Może już spotkaliście grę wyścig żółwia. Na planszy mamy 2 lub więcej zawodników, które w przemieszczają się w kierunku mety. Wykonują ruchy na zmianę, natomiast długość ruchu zależy od wylosowanej liczby.

Import bibliotek

Zaczynamy od zaimportowania bibliotek, które będą nam potrzebne, ostawienia koloru tła na czarne, oraz nadaniu nazwy okienku programu

```
import turtle
import time
import random

turtle.bgcolor('white')
turtle.title("Wyścig żółwi")
```

Meta

Na początku możemy zająć się, narysowaniem mety. Tworzymy do tego celu nowy obiekt Turtle(), ustawiamy go w odpowiednim miejscu, piszemy – Finish Line. Następnie przesuwamy go na początek mety i rysujemy.

```
meta = turtle.Turtle()
meta.color('white')
meta.penup()
meta.goto(240,220)
meta.write("Finish Line", font=("Arial", 15, "bold"))
```

```
meta.penup()
meta.goto(300,200)
meta.pendown()
meta.goto(300,-200)
meta.hideturtle()
```

Zawodnicy

Kolejnym krokiem jest utworzenie 2 żółwi. Odpowiednio – czerwonego oraz niebieskiego. Podnosimy długopis (penup), oraz ustawiamy je na swoich miejscach startowych.

```
red = turtle.Turtle()
red.shape('turtle')
red.color('red')
red.pensize(7)
red.penup()
red.goto(-200,100)
red.pendown()

blue = turtle.Turtle()
blue.shape('turtle')
blue.color('blue')
blue.pensize(7)
blue.penup()
blue.goto(-200,-100)
blue.pendown()
```

Odliczanie

Jak żółwie zajmą już pozycję, możemy rozpocząć odliczanie. W naszym przypadku 3, 2, 1, Start !!!

```
odliczanie = turtle.Turtle()
odliczanie.color('white')
odliczanie.penup()
odliczanie.goto(-200,200)
odliczanie.hideturtle()

for x in range(3):
    odliczanie.write(3-x, font=("Arial", 40, 'bold'))
    time.sleep(1)
    odliczanie.clear()

odliczanie.write("Start !!!", font=("Arial", 30, 'bold'))
```

Przemieszczanie się żółwi

Żółwie przemieszczają się na zmianę. raz jeden raz drugi. Aby było uczciwie, musimy zacząć od wylosowania, który z żółwi zaczyna. Następnie losujemy liczbę o jaką ma się przesunąć do przodu, z przedziału od 0 do 70. Pętlę powtarzamy co 0.3 sekundy.

Wartości 0-70 oraz 0.3 sekundy, są dobrane tylko po to, aby żółwie nie przesuwały się ani za szybko ani za wolno. Można umieścić dowolne inne wartości.

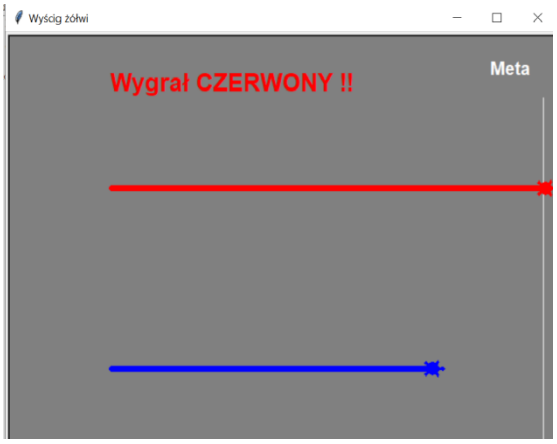
Po każdym ruchy, wywołujemy funkcję sprawdź, która sprawdza czy dany żółw, przekroczył linię mety. Jeżeli tak, to pętla zostaje przerwana i żółwie przestają się ruszać.

```
first = random.choice(['red','blue'])

while True:

    if first == 'red':
        red.forward(random.randint(0,70))
        if sprawdz(): break
        blue.forward(random.randint(0,70))
        if sprawdz(): break
    else:
        blue.forward(random.randint(0,70))
        if sprawdz(): break
        red.forward(random.randint(0,70))
        if sprawdz(): break

    time.sleep(0.3)
```

Sprawdzanie czy któryś z żółwi wygrał

Jednocześnie nasza funkcja, drukuje komunikat o tym, który z zawodników wygrał.

```
def sprawdz():

    if red.position()[0] >= 300:
        odliczanie.clear()
        odliczanie.color('red')
        odliczanie.write('Wygrał CZERWONY !!', font=("Arial", 20, 'bold'))
        return True

    if blue.position()[0] >= 300:
        odliczanie.clear()
        odliczanie.color('blue')
        odliczanie.write('Wygrał NIEBIESKI !!', font=("Arial", 20, 'bold'))
        return True

    return False
```



Zadanie 9

Wykorzystując kod gry w żółwie, uzupełnij program o następujące funkcje:

- wybór żółwia który wygra – komunikat;
- informacje którego żółwia wybrał gracz;
- informacje o trafności typowania;
- sumę typowań poprawnych i niepoprawnych.