1 Biblioteka Turtle – grafika żółwia

2 Podstawowe ruchy turtle

Zaczynamy od Importu biblioteki turtle:
from turtle import*

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

forward(x) lub fd(x) -	do przodu, o podany dystans
backward(x) lub bk(x) -	do tyłu, o podany dystans
right(x) lub rt(x), left(x) lub lf(x) -	obróć się w prawo / lewo, o podany kąt
goto(x,y) -	pójdź do konkretnego punktu
home() -	wróć na środek ekranu
circle(x) -	narysuj koło o podanym promieniu
bgcolor(x) -	zmiana koloru tła. Jako x możemy podać takie kolory jak 'red','blue','green','yellow','black', 'white' i wiele innych. Możemy również podać kolor w RGB (x,y,z)
pencolor(x) –	zmiana koloru pisaka
color(x)	zmienia kolor naszego żółwia i linii które rysuje
pensize(x)	zmiana grubości pisaka
speed(x) -	prędkość poruszania się pisaka
	0 - najszybciej
	1 - najwolniej
	3- wolno
	6 - normalnie
	10 - szybko
shape("turtle") -	zmiana kształtu pisaka
write("KONIEC", font=("Arial", 12))	tworzenie napisów
penup()	powoduje że żółw przemieszcza się bez rysowania
pendown()	żółw znowu rysuje
hideturtle() oraz showturtle()	żółw znika / pojawia się
fillcolor("kolor") -	ustawia kolor wypełnienia
begin_fill() -	początek wypełnienia figury
end_fill()	koniec wypełnienia figury

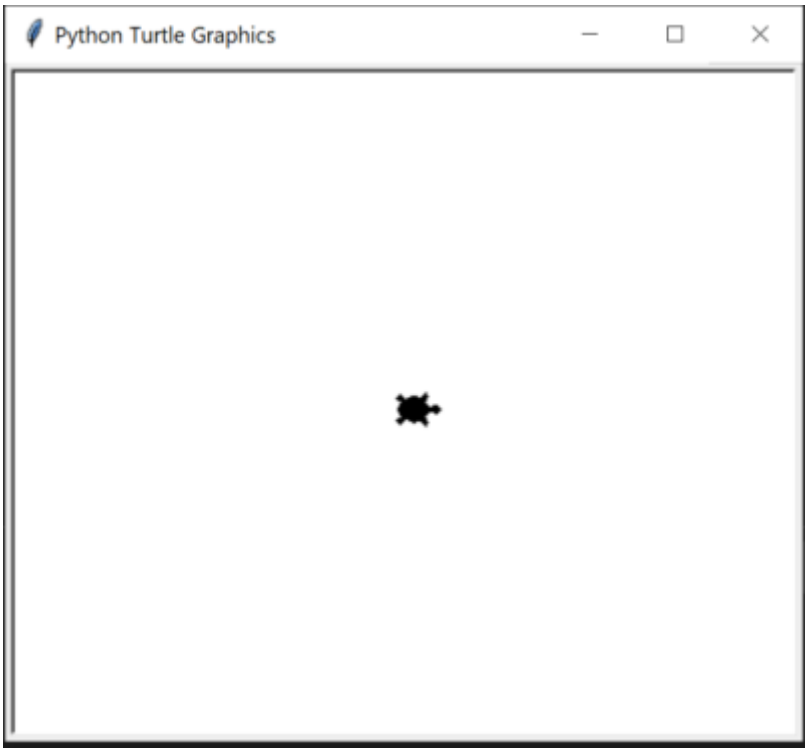
Losowe kolory:
from random import*
bgcolor(randrange(1,256),randrange(1,256),randrange(1,256))

2.1 Pierwszy program

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.

```
from turtle import *
t= Turtle()
t.shape('turtle')
exitonclick()
```

a następnie, po uruchomieniu programu, zobaczymy następujące okienko

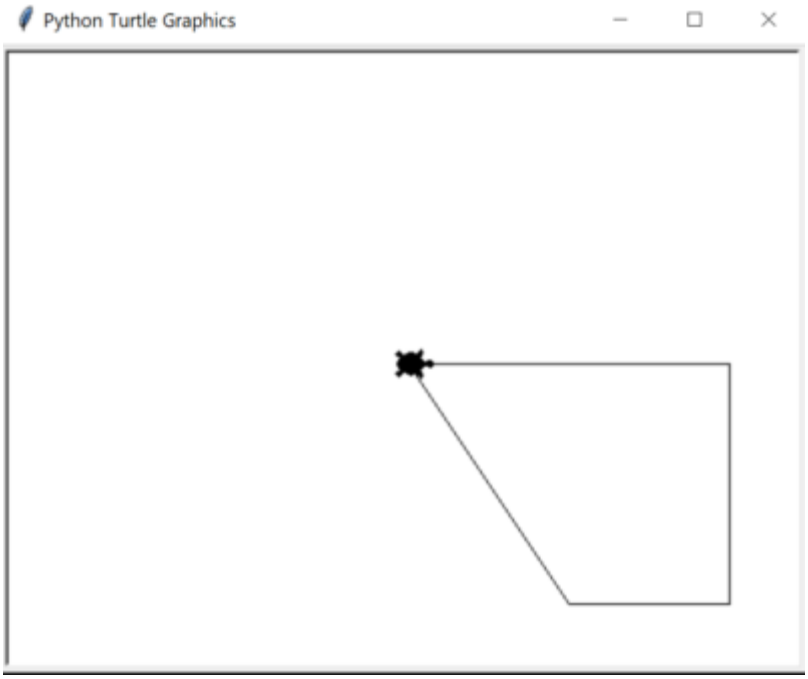


funkcja exitonclick(), umieszczona na końcu, powoduje że okienko się nie zamknie, dopóki nie klikniemy.

W rezultacie, po dodaniu następujących linijek do naszego programu:

```
from turtle import *
t = Turtle()
t.shape('turtle')
t.forward(200)
t.right(90)
t.forward(150)
t.right(90)
t.forward(100)
t.home()
texitonclick()
```

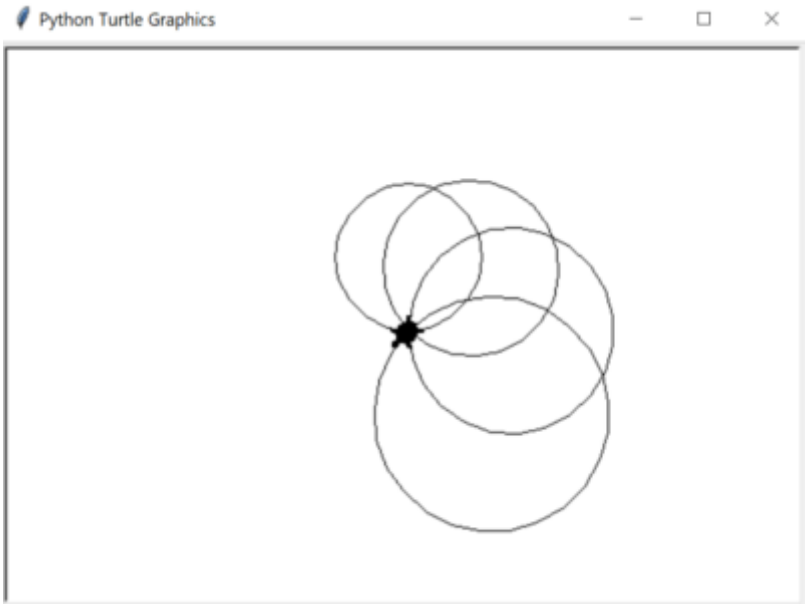
Otrzymamy następującą figurę:



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:

```
from turtle import *
t = Turtle()
t.shape('turtle')
t.circle(50)
t.right(45)
t.circle(60)
t.right(45)
t.circle(70)
t.right(45)
t.circle(80)
exitonclick()
```



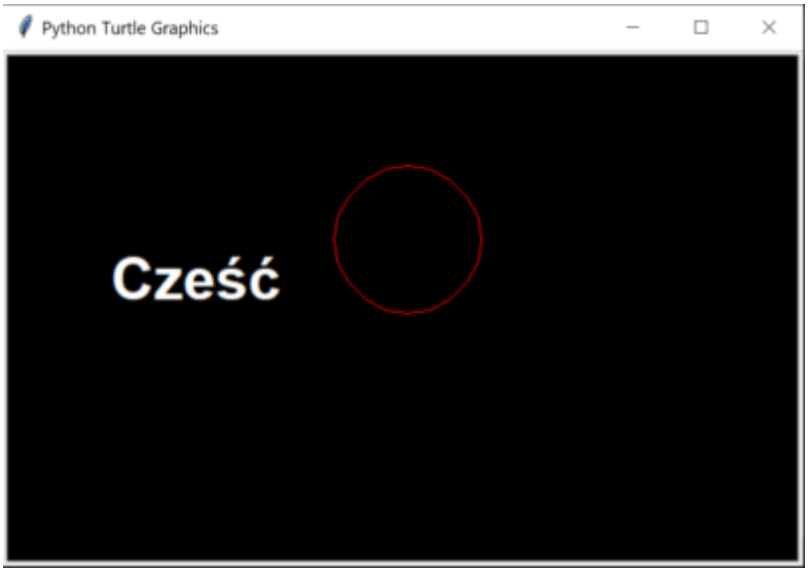
3 Kolory, rozmiar oraz prędkość

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

I jeżeli połączymy kilka tych funkcji w jeden program:

```
from turtle import*
bgcolor('black')
t = Turtle()
t.shape('turtle')
t.color('red')
t.circle(50)
t.penup()
t.goto(-200,0)
t.pendown()
t.color('white')
t.write('Cześć', font=("Arial",30,"bold"))
t.hideturtle()
exitonclick()
```

Otrzymamy następująca grafikę:

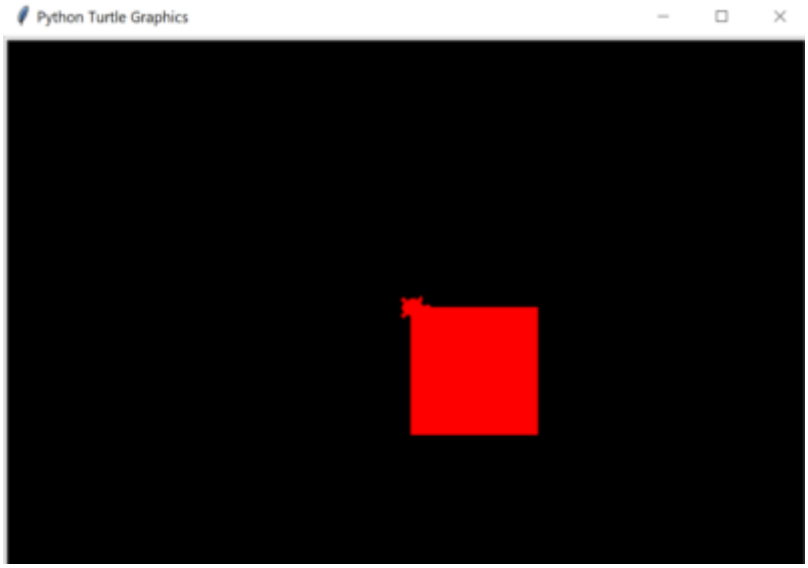


4 Wypełnianie kolorem w turtle

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.

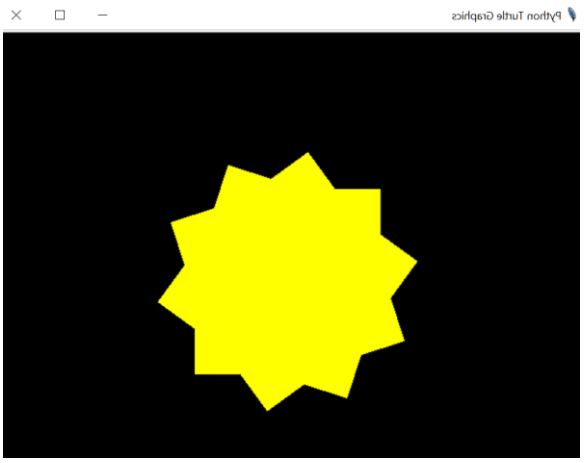
```
from turtle import*
bgcolor('black')
t = Turtle()
t.shape('turtle')
t.color('red')
t.begin_fill()
for i in range(4):
        t.forward(100)
        t.right(90)
t.end_fill()
exitonclick()
```

A w rezultacie otrzymamy:



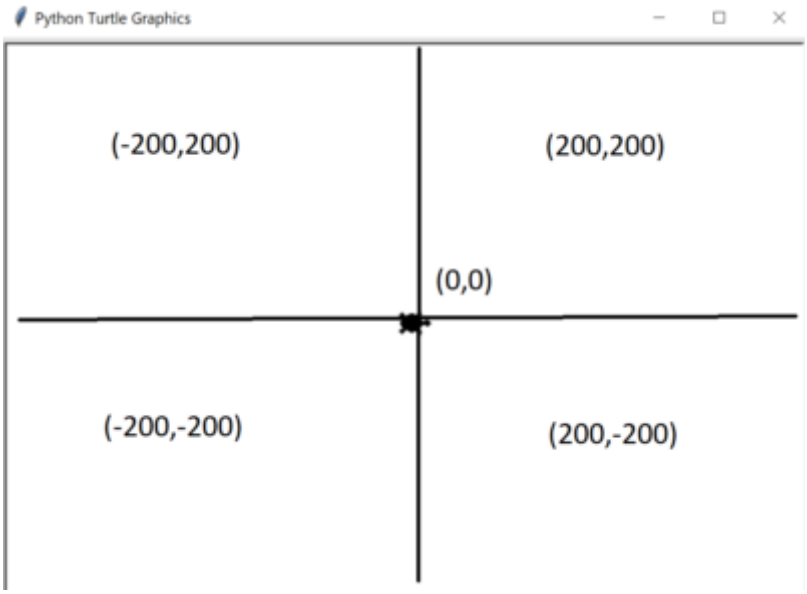
```
from turtle import*
bgcolor('black')
t = Turtle()
t.shape('turtle')
t.color('yellow')
for j in range(10):
    t.begin_fill()
    for i in range(4):
        t.forward(100)
        t.right(90)
    t.left(36)

    t.end_fill()
exitonclick()
```



5 Lokalizacja żółwia

Jak już mogliśmy zauważyć, 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:

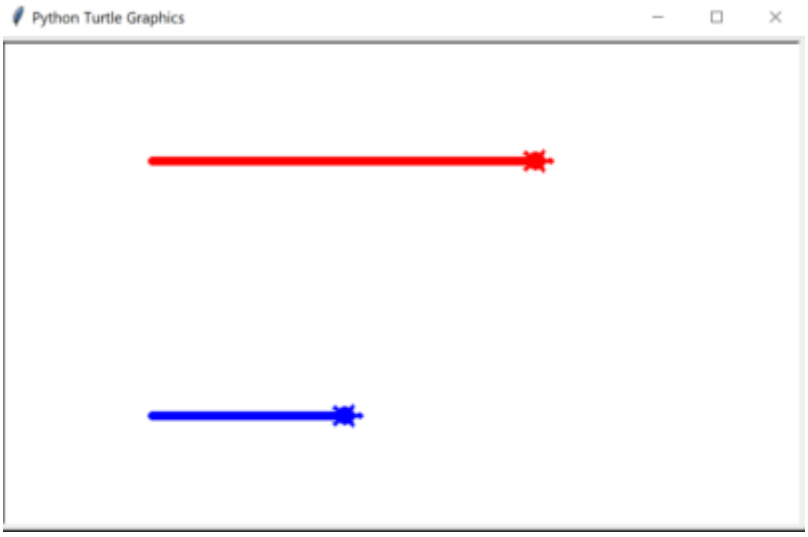


Dodatkowo, w każdym momencie, możemy uzyskać położenie naszego żółwia, za pomocą funkcji *position()*.

6 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.

```
import turtle
t = turtle.Turtle()
t.shape('turtle')
t.color('red')
t.pensize(7)
t.penup()
t.goto(-200,100)
t.pendown()
t.forward(300)
t2 = turtle.Turtle()
t2.shape('turtle')
t2.color('blue')
t2.pensize(7)
t2.penup()
t2.goto(-200,-100)
t2.pendown()
t2.forward(150)
turtle.exitonclick()
```



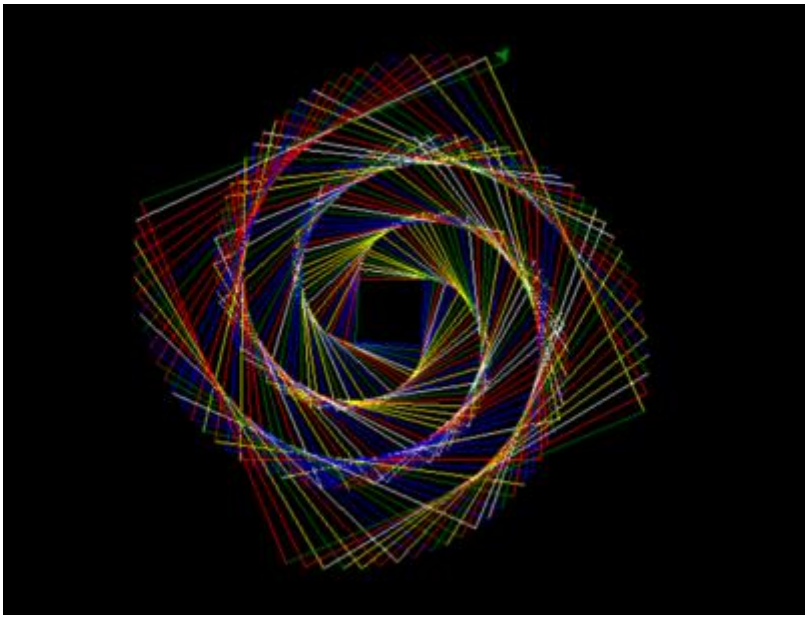
7 Pierwsza, bardziej złożona figura w Python turtle

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
- przemieszczamy żółwia o x
- obracamy żółwia o 91 stopni
- powtarzamy pętlę

```
import turtle
import random
turtle.bgcolor('black')
t = turtle.Turtle()
t.speed(0)
for x in range(50, 300):
t.color(random.choice(['white','red','blue','green','yellow']))
t.forward(x)
t.right(91)
turtle.exitonclick()
```

W rezultacie otrzymujemy:



8 Obsługa klawiszy w turtle

W turtle, mamy możliwość nasłuchiwania na zdarzenia, takie jak naciśnięcie klawisza, i jego obsłużenie. 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, oraz całość ma być powtarzana.

turtle.listen()

turtle.mainloop()

program zakończy się, w momencie kiedy zostanie uruchomiona funkcja turtle.bye()

Zdecydowanie prościej jest to pokazać, niż wytłumaczyć, tak więc przyjrzyjmy się prostemu programowi.

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

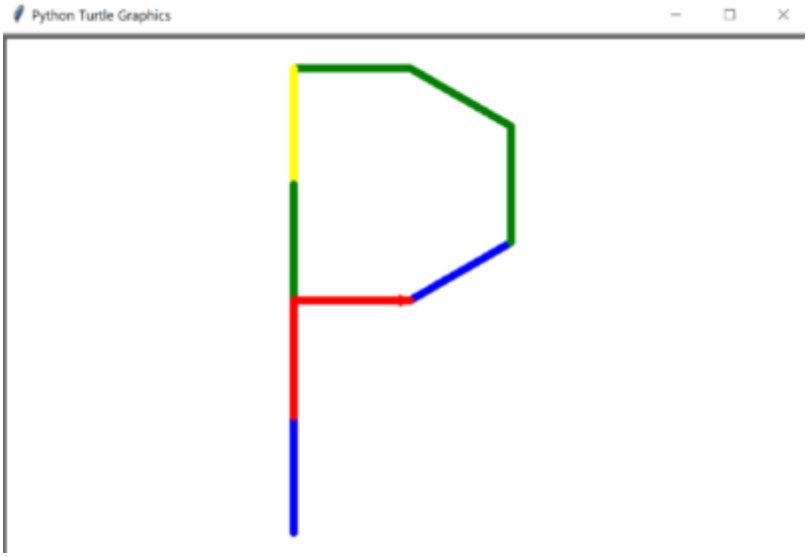
- strzałka do góry, przemieszcza żółwia o 100
- strzałka w prawo, obraca go w prawo o 30 stopni
- strzałka w lewo, obraca go w lewo o 30 stopni
- 'q' kończy działanie naszego programu

W tym celu, zdefiniowaliśmy 4 funkcje, które wykonują odpowiednie akcje. Aby program był atrakcyjniejszy, przy każdym ruchu, żółw losuje swój kolor.

```
import turtle
t = turtle.Turtle()
def up():
t.forward(100)
def left():
t.left(30)
def right():
t.right(30)
def bye():
turtle.bye()
```

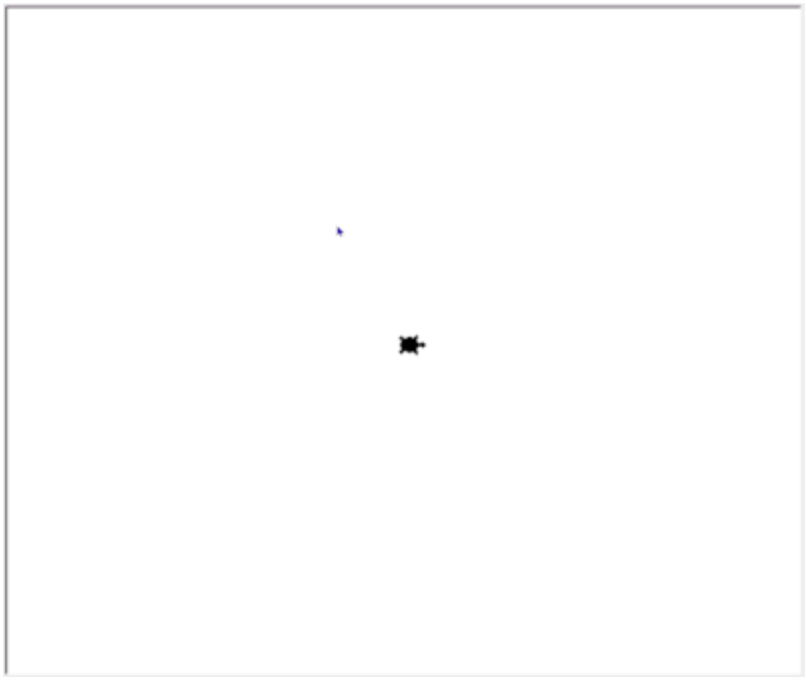
```
turtle.onkey(up,'Up')
turtle.onkey(left,'Left')
turtle.onkey(right,'Right')
turtle.onkey(bye,'q')
turtle.listen()
turtle.mainloop()
```

W rezultacie, otrzymaliśmy możliwość sterowania naszym żółwiem, co wykorzystaliśmy do próby napisania literki 'P'. 'P' jak Python



8.1 obsługa zdarzeń

Do tej pory pokazywaliśmy podstawowe operacje w Turtle, które umożliwiały nam rysowanie różnych kształtów. Tym razem zobaczymy jak obsłużyć zdarzenia. Przykładowo naciśniecie klawisza lub kliknięcie myszki. Tym samym nasze aplikacji będą mogły wchodzi w interakcję z użytkownikiem.



9 Przypisanie zdarzenia do klawisza

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

```
import turtle
window = turtle.Screen()
t = turtle.Turtle()
def up():
pass
window.onkey(up,'Up')
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 strzałka do góry.

Jako drugi parametr, przyjmuje funkcję która ma się wykonać. Czyli w momencie kiedy użytkownik naciśnie strzałkę do góry, wykona się funkcja up().

Na początku kodu, importujemy bibliotekę turtle, oraz tworzymy obiekt window = turtle.Screen()

Na końcu kodu, wykonujemy jeszcze 2 ważne funkcje:

- window.listen() – informujemy turtle, że ma nasłuchiwać na zdarzenia
- window.mainloop() – informujemy turtle, aby nie zamykał programu, po pojawieniu się klawisza, ale by dalej program działał

9.1 Obsługa wielu klawiszy w turtle

Jeżeli chcemy obsługiwać wiele klawiszy, robimy to analogicznie. Za pomocą funkcji onkey(), przypisujemy więcej funkcji do większej ilości klawiszy. Przykładowo:

```
def up():
t.forward(100)
def left():
t.left(45)
def right():
t.right(45)
def bye():
turtle.bye()
window.onkey(up,'Up')
window.onkey(left,'Left')
window.onkey(right,'Right')
window.onkey(bye,'q')
```

Powyższy kod:

- przypisuje funkcje do naciśnie strzałek
- przypisuje funkcje która zamyka program do klawisza 'q’
- Jeżeli użytkownik naciśnie strzałkę do góry, to żółw pójdzie do przydu. Przy strzałkach w bok, żółw skręci.

9.2 Obsługa kliknięcia myszy w turtle

Do tej pory używaliśmy funkcji window.onkey(), aby przypisać funkcję która ma się wykonać do konkretnego klawisza.

Aby obsłużyć kliknięcie myszy, mamy do dyspozycji funkcję window.onclick(), która przyjmuje jako parametr funkcję która ma się wykonać jeżeli użytkownik kliknie. Dodatkowo, funkcja ta, otrzymuje jako parametry, miejsce kliknięcia myszy, w postaci współrzędnych x, y.
Zobaczmy jak to wygląda w kodzie:

```
def click(x,y):
t.goto(x,y)
window.onclick(click)
```

1. Użyliśmy funkcji window.onclick(), i przypisaliśmy nowo utworzoną funkcję click
2. Funkcja klik, przyjmuje parametry x, y
3. W momencie kiedy użytkownik kliknie, miejsce kliknięcia jest przekazywane do funkcji
4. I w naszym przykładzie, żółw przemieści się do wskazanego miejsca

9.3 Całość kodu

Aby dodatkowo, uatrakcyjnić program, przed każdorazowym przemieszczeniem się żółwia, żółw zmienia kolor na losowo wybrany.

```
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')
window.onkey(left,'Left')
window.onkey(right,'Right')
window.onkey(bye,'q')
window.onclick(click)
window.listen()
window.mainloop()
```

A my w rezultacie otrzymujemy prosty program, reagujący na naciśnięcie strzałek, klawisza 'q’ oraz kliknięcie myszki: