

# Co było ostatnio?

Wyścigi żółwi:

- obiekty:
  - definiowanie obiektu: `zenek = turtle.Turtle()`
  - metody (funkcje związane z obiektem): `zenek.goto(100,-20)`
- pętla: `while warunek:`  
    ...

Rząd kwadratów i szachownica:

- pętla: `for zmienna in range(n):`  
    ...
- funkcje:
  - definiowanie funkcji: `def nazwanowejfunkcji():`  
        ... (wcięcie, treść funkcji)
  - użycie nowej funkcji: `nazwanowejfunkcji()`

# Co będzie dziś?

## Wektory

- co to są wektory i gdzie się ich używa? i jak :)
- jak je narysować w pythonie
- definiowanie funkcji z argumentami:

```
def nazwanowejfunkcji(x,y):
```

## Bajka o liczbach

...

[https:](https://github.com/radoslawwieczorek/Kwantowanie-w-pythonie)

[//github.com/radoslawwieczorek/Kwantowanie-w-pythonie](https://github.com/radoslawwieczorek/Kwantowanie-w-pythonie)

Następujący kod rysuje układ współrzędnych

```
import turtle
xmax=180
ymax=180
zolw = turtle.Turtle()
zolw.speed(0)
for i in range(2):
    zolw.fd(-xmax)
    zolw.fd(2*xmax)
    zolw.rt(135)
    zolw.fd(10)
    zolw.bk(10)
    zolw.rt(90)
    zolw.fd(10)
    zolw.bk(10)
    zolw.rt(135)
    zolw.home()
    zolw.lt(90)
zolw.up()
```

```
zolw.home()
for x in [-150,-100,-50,50,100,150]:
    zolw.goto(x,0)
    zolw.down()
    zolw.write(x//50)
    zolw.setheading(-90)
    zolw.fd(10)
    zolw.up()
for y in [-150,-100,-50,0,50,100,150]:
    zolw.goto(4,y)
    zolw.write(y//50)
    zolw.goto(0,y)
    zolw.down()
    zolw.setheading(180)
    zolw.fd(10)
    zolw.up()
zolw.ht()
```

# Definiowanie funkcji (przypomnienie).

Definiujemy:

```
def kwadrat():  
    """Rysuje kwadrat"""  
    for i in range(4):  
        turtle.left(90)  
        turtle.fd(100)
```

Używamy:

```
kwadrat()
```

(narysuje nam kwadrat o boku 100)

Jeśli zdefiniujemy:

```
def kwadratoboku(dlugoscroku):  
    """Rysuje kwadrat o boku dlugoscroku"""  
    for i in range(4):  
        turtle.left(90)  
        turtle.fd(dlugoscroku)
```

Gdy użyjemy:

```
kwadratoboku(50)
```

narysuje kwadrat o boku 50.

## Wektory. Zadania.

1. Narysuj wektory  $[1, 3]$  i  $[-2, -1.5]$  (osobnymi żółwiami).

*Uwaga:* Możesz ustawić kierunek żółwia w stronę  $(x,y)$  za pomocą:  
`nazwazolwia.seth(nazwazolwia.towards(x,y))`.

2. Napisz funkcję `rysujwektor(v)`, która rysuje wektor o współrzędnych  $v = [x, y]$ .

*Uwaga:*  $x$  i  $y$  są liczbami typu `float`,  $[x,y]$  jest **listą** dwóch liczb.

- Narysuj wektory  $v_1 = [1, 2]$  i  $v_2 = [2.5, 0.5]$  za pomocą funkcji.

3. Napisz funkcję `rysujwektorod(v, poczatek)`, która rysuje wektor o współrzędnych  $v = [x, y]$  zaczepiony w punkcie `poczatek`  $= [x_0, y_0]$ .

- Przesuń wektor  $v_1$  na koniec wektora  $v_2$ .
- Narysuj sumę wektorów  $v_1$  i  $v_2$ .

# Liczby zespolone.

1. Odpowiedz na pytania ze strony: [Khan Academy. Lekcja 2: Wprowadzenie do liczb zespolonych](#):  
(*Sprawdź, czy rozumiesz, Pytanie do zastanowienia, Teraz spróbuj sam!*)
2. Niech  $z_1 = 1 - 1i$ ,  $z_2 = 3 + 4i$ . Policz (na kartce):
  - 1  $z_1 + z_2$
  - 2  $z_1 - z_2$
  - 3  $z_1 z_2$
  - 4  $\frac{z_1}{z_2}$
3. Zaznacz liczby  $z_1$ ,  $z_2$  oraz  $z_1 + z_2$  na płaszczyźnie Gaussa.

# Liczby zespolone w Pythonie

1. Sprawdź, jak działają operatory `+`, `-`, `*` i `/` na liczbach zespolonych.
2. Sprawdź, że liczby `1`, `1.0`, oraz `1.0 + 0j` są równe. Zobacz, jakiego są typu. Użyj `print(type())`.
3. Sprawdź atrybuty `.real` (część rzeczywista) i `.imag` (część urojona): Zobacz, ile to jest `(2-3j).real`, `(2-3j).imag` oraz `z.imag`, jeśli `z = (3 - 2) * (3 + 2j)`.
4. Zobacz, ile to jest  $i^2$ ,  $(2 + 1i)^2$ , oraz  $(2 + i)(2 - i)$ .