Python - początki 2

"You can't have a standard education
and expect to have extraordinary life"
(Nie możesz mieć standardowego wykształcenia i spodziewać się niezwykłego życia)
~ Tony Robbins

1. Rysowanie z Żółwiami

Żółw (turtle) w Pythonie jest mniej więcej tym samym żółwiem co w świecie rzeczywistym. Wiemy, że żółwie są powolnymi zwierzakami z własnym domem na plecach. W świecie Pythona, żółw to mała, czarna strzałka powoli poruszająca się po ekranie, która zostawia ślad po swoich ruchach (mniej więcej jak ślimak). Żółw jest świetną opcją dla chcących nauczyć się podstawy komputerowej grafiki, więc pokażemy jak za jego pomocą narysować proste kształty lub linie.

Moduły w Pythonie

Moduł to sposób zapewnienia użytecznego kodu, który może zostać użyty w innym programie. Może zawierać funkcje, których chcemy użyć.
Moduł *turtle* to sposób na programowanie grafiki wektorowej, czyli proste linie, krzywe i kropki.
Żeby użyć jakiegoś modułu potrzebujemy komendy **import**:

```
import turtle
```

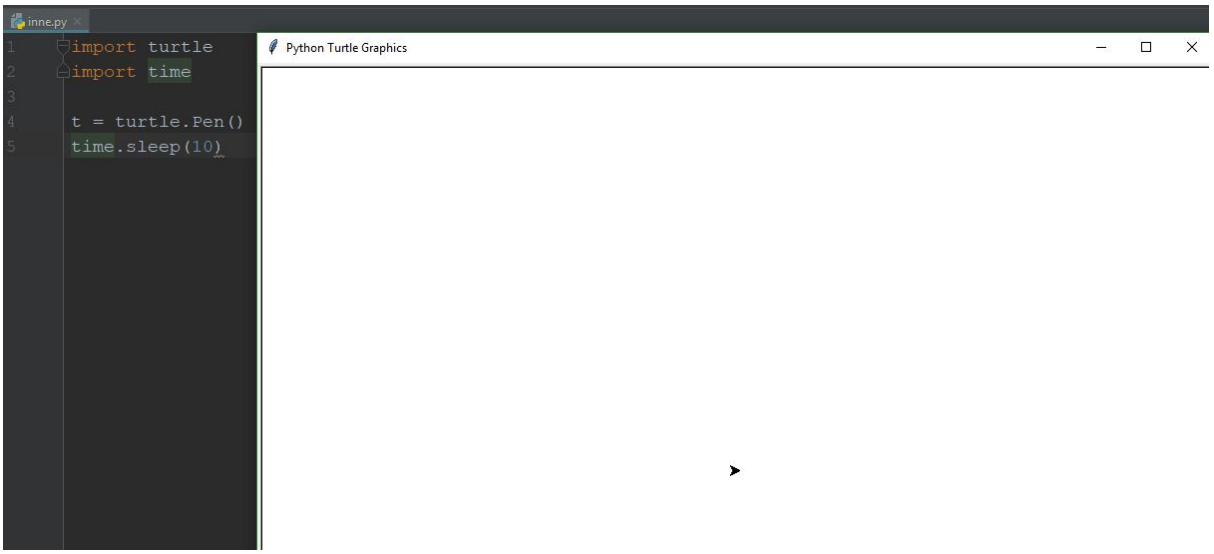
Importowanie modułu mówi Pythonowi o tym, że chcesz go użyć.

Tworzenie Canvasu

Po zaimportowaniu modułu *turtle*, musimy stworzyć canvas(ang. płótno) - puste miejsce do rysowania, jak płótno malarza. Żeby to zrobić, musimy wywołać funkcję *Pen* z modułu *turtle*, która automatycznie tworzy dla nas ów canvas.

```
t = turtle.Pen()
```

Powinieneś teraz zobaczyć puste okienko (canvas) ze strzałką na środku:



Strzałka, którą widzisz na środku ekranu jest naszym żółwiem. Tak, wiemy - nie przypomina żółwia.
Po kliknięciu na nasze okienko z żółwiem, okienko może nie działać poprawnie. Zależnie od środowiska, w jakim piszemy nasz kod (Python konsolowy lub PyCharm) ma to różne przyczyny:

- Python konsolowy - żółw czeka na kolejne ruchy do wykonania,
- PyCharm (okienko znika) - nie ustawiliśmy zatrzymania ekranu (sleep(), jak na kodzie powyżej) lub czeka na wpisanie czegoś z konsoli.

Poruszanie żółwiem

Aby przekazać instrukcje dla żółwia, musimy użyć funkcji na zmiennej *t*, którą wcześniej stworzyliśmy, podobnie jak użyliśmy *Pen()* z modułu *turtle.*
Na przykład: funkcja *forward()* każe żółwiowi poruszać się prosto. By powiedzieć żółwiowi, by poruszył się o 50 pixeli, wpisz komendę:

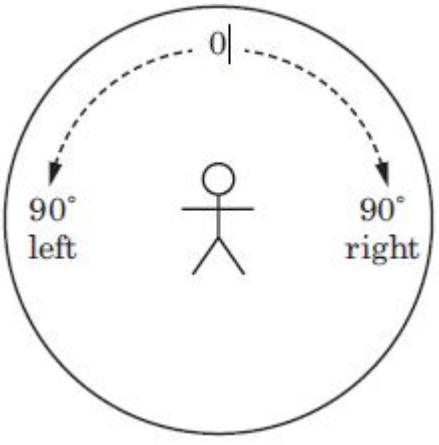
```
t.forward(50)
```

Teraz jeśli chcemy, by nasz żółw obrócił się w lewo o 90 stopni, używamy poniższej komendy:

```
t.left(90)
```

Jeśli nie uczyłeś się jeszcze czym są stopnie - czas na mini-poradnik.

- Kierunek do którego zmierzasz to 0 stopni,
- To, co jest po Twojej lewej stronie, jest na 90 stopni na lewo,
- To, co jest po Twojej prawej stronie, jest na 90 stopni w prawo.



Jeśli będziesz obracać się wokół własnej osi cały czas w prawo, pełen obrót, który wykonasz, będzie składał się z 360 stopni. Tyle z podstawówki na dziś.

Oto przykład kodu rysujący kwadrat:

```
t.forward(50)
t.left(90)
t.forward(50)
t.left(90)
t.forward(50)
t.left(90)
t.forward(50)
t.left(90)
```

Spis przydatnych funkcji

Poruszanie żółwiem:

```
t.forward(100) #w przód o 100 pixeli
t.backward(50) #w tył o 50 pixeli
t.left(90) #obrót w lewo o 90 stopni
t.right(45) #obrót w prawo o 45 stopni
```

reset() - Wymazuje canvas i ustawia żółwia na wyjściowej pozycji.

```
t.reset()
```

clear() - Wymazuje canvas i zostawia żółwia tam gdzie był.

```
t.clear()
```

up() - Dosłownie "podnosi" żółwia, by ten nie zostawiał po sobie śladu. Możemy wtedy przemieścić żółwia na pożądaną przez nas pozycję…

```
t.up()
```

down() - ...i "opuścić" go w danym miejscu po jego przemieszczeniu. Żółw będzie teraz ponownie zostawiał ślad.

```
t.down()
```

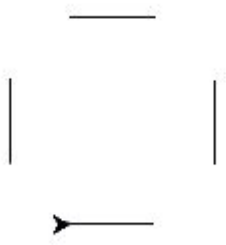
2. Podsumowanie

W tym rozdziale dowiedziałeś się, czym jest moduł oraz jak go użyć we własnym programie. Nauczyłeś się teraz rysować z wykorzystaniem modułu *turtle* i jak użyć podstawowych funkcji w nim zawartych.

3. Zagadki programistyczne

#1 Trójkąt
Napisz program, który narysuje trójkąt.

#2 Kwadrat bez rogów
Napisz program, który narysuje cztery oddalone od siebie linie, które będą tworzyć kwadrat bez rogów jak na załączonym obrazku:



Wielkość nie jest ważna, chodzi o kształt ;)

(Zaawansowana wersja - zrób pętlę wypisującą podaną przez użytkownika ilość kwadratów bez rogów, gdzie każdy kolejny jest większy od poprzedniego (jpg przykładowego wyniku na githubie))

4. Bycie eko i recykling

W świecie programowania ponowne używanie rzeczy (w tym wypadku kodu) jest tak samo ważne jak w realnym świecie. Oczywiście, nasz program nie zniknie pod stertą śmieci, ale jeśli nie będziemy wykorzystywać jego części ponownie to w końcu zaczną boleć nas opuszki palców od ciągłego pisania tego samego. W tym rozdziale dowiesz się o wielu metodach ponownego użycia kodu.

Używanie funkcji

Najprościej mówiąc funkcja to fragment kodu, który może być wykonywane wielokrotnie z różnych miejsc programu.

W jednym z poprzednich rozdziałów wykorzystaliśmy funkcję **range()**. Gdy piszesz proste programy funkcje są przydatne. Gdy zaczniesz pisać długie i skomplikowane programy takie jak np. gry - funkcje są niezbędne (zakładając, że chcesz skończyć pisanie programu w tym stuleciu).

Funkcja składa się z trzech części: nazwy, list parametrów i treści.

Przykład prostej funkcji:

```
def hello(name):
    print('Hello ' + name)
```

W powyższym przykładzie nazwą tej funkcji jest *hello*. Przyjmuje ona jeden parametr o nazwie *name*, a treścią funkcji jest blok kodu. Nazwa naszej funkcji musi być poprzedzona wyrazem **def** (skrót od define), a jej parametr istnieje tylko wewnątrz treści funkcji.
Możesz uruchomić funkcję wywołując jej nazwę i używając nawiasów wokół wartości parametru:

```
hello('Kinga')
# wypisane zostanie 'Hello Kinga'
```

Funkcja może przyjmować więcej niż jeden parametr (może ich przyjmować nieskończenie wiele). Wystarczy oddzielić je od siebie przecinkami. Jak parametry możemy też przekazywać zmienne.

```
firstname = 'Mikołaj'
lastname = 'Korbanek'
hello(firstname, lastname)
# Hello Mikołaj Korbanek
```

Funkcje często wykorzystywane są do zwracania wartości. Wykorzystujemy do tego wyrażenie **return**.

Na przykład możemy napisać funkcję, która obliczy ile pieniędzy zaoszczędziliśmy:

```
def savings(earnings, expenses):
    return earnings - expenses
```

Zmienne i zakresy

Zmienne, które znajdują się w ciele (treści) funkcji, nie mogą być ponownie użyte gdy funkcja zakończy działanie, ponieważ istnieje tylko wewnątrz tej funkcji.

Spójrzmy na tą prostą funkcje. Wykorzystuje ona kilka zmiennych i nie przyjmuje żadnych parametrów.

```
def multiply():
    firstNumber = 10
    secondNumber = 20
    return firstNumber * secondNumber
```

Kiedy wywołamy funkcję dostaniemy rezultat, ale gdy spróbujemy wypisać jakiś numer wyskoczy nam błąd.

```
print(multiply)
# 200
print(firstNumber)
NameError: name 'first_variable' is not defined
```

Funkcje można również pogrupować w moduły.

Używanie modułów

Czym jest moduł dowiedzieliście się czytając o żółwiku :).Służą one do grupowania funkcji, zmiennych i innych rzeczy w większe i potężniejsze programy. Niektóre moduły wbudowane są w Pythona, a inne można pobrać osobno. Znajdziesz moduły, które pomogą Ci tworzyć gry (takie jak tkinter, który jest wbudowany i PyGame, który nie jest wbudowany), odpowiedzialne za manipulacje zdjęciami (np. PIL), moduły odpowiedzialne za grafikę 3d (np. Panda3D) i wiele więcej. Moduł może być używane do robienia różnych rzeczy. Na przykład jeśli tworzyłeś symulator, a chciałeś świat gry zmienić ma się zmieniać zależnie od aktualnej godziny możesz użyć wbudowanego modułu o nazwie time:

```
import time
```

Tutaj polecenie **import** służy do powiadomienia pythona o tym, że chcemy użyć modułu **time**. Od teraz możemy wywoływać funkcje dostępne w tym module. Na przykład możemy wywołać funkcję asctime z modułu time:

```
print(time.asctime())
# Sun Oct 14 17:47:28 2018
```

Funkcja asctime jest częścią modułu time i zwraca bieżącą datę i czas jako string.

5. Podsumowanie

W tym rozdziale dowiedziałeś się, jak tworzyć funkcje i korzystać z funkcji dostarczanych przez moduły. Dowiedziałeś się, że zmienna mają zakresy i może je deklarować wewnątrz lub na zewnątrz funkcji.

6. Zagadki programistyczne

#3 Naj z 3

Stwórz prostą funkcję, która znajdzie największą z dwóch liczb oraz funkcję która znajdzie największą z 3 liczb (wykorzystując funkcję odpowiedzialną za znalezienie największej z dwóch).

#4 Suma listy

Napisz funkcję, która będzie zwracać sumę wszystkich elementów podanej listy.