Laboratorium programistyczne — Python: Rysowanie na płaszczyźnie

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2020-03-23

1 Grafika żółwia

Koncepcja proceduralnego rysowania różnych kształtów przy pomocy poleceń typu „zrób X kroków na przód”, „obróć się o kąt X stopni”, itd. przy pomocy *„żółwia”*, któremu wydajemy te polecenia pochodzi z języka LOGO. Język ten stworzony został w 1967 roku, głównie z myślą o zastosowaniach edukacyjnych.

Dzięki modułowi *turtle* możemy „bawić” się w ten sposób w współczesnym, profesjonalnym i jednym z najpopularniejszych języków programowania jakim jest Python. Aby rozpocząć pracę z żółwiem należy zaimportować ten moduł, można też ustawić kształt używany do rysowania na żółwia:

```
from turtle import *
shape('turtle')
```

Po wydaniu tych poleceń powinno pokazać się okienko, po środku którego widzimy żółwia. Nie zamykaj tego okienka – ustaw je obok naszej konsoli pythonowej, tak aby widzieć oba okna równocześnie.

1.1 Podstawowe ruchy

Wywołując w konsoli pythonowej odpowiednie funkcje możemy teraz sterować żółwiem. Podstawowe jego ruchy to:

- `forward(x)` - ruch na przód o `x` pixeli
- `left(x)` - obrót w lewo o `x` stopni
- `right(x)` - obrót w prawo o `x` stopni
- `circle(x)` - zrobienie kółka w lewo o promieniu `x` pixeli,
  jeżeli $x < 0$ to kółko w prawo o promieniu `-x`
- `circle(x, y)` - łuk o promieniu `x` pixeli i długości `y` stopni
- `reset()` - czyści obrazek i ustawia żółwia w pozycji wyjściowej

Zadanie 1.1.1

Narysuj przy pomocy żółwia i powyższych poleceń kwadrat.

Jak wiesz, większe programy wygodniej jest pisać (i poprawiać, zmieniać) w pliku tekstowym, który uruchamiamy przy pomocy interpretera Pythona, niż w bezpośrednio w samym interpreterze. W przypadku rysowania żółwiem warto dodać opcję `-i` aby python nie zakończył pracy (po wykonaniu wszystki instrukcji z pliku), dzięki czemu okienko z rysunkiem pozostanie widoczne. Czyli kod z pliku o nazwie `żółw.py` uruchamiamy poleceniem:

```
python3 -i żółw.py
```

Zadanie 1.1.2

Napisz program który narysuje 5 kwadratów, jeden wewnątrz drugiego, każdy o boku o połowę mniejszym niż poprzedni

Na poprzednich zajęciach dowiedzieliśmy się, że aby unikać powtarzania podobnego kodu, stosuje się w programowaniu funkcje. Pozwalają one wywołać nazwany fragment kodu z innego miejsca i przekazać do niego jakieś argumenty.

Zadanie 1.1.3

Napisz funkcję, która rysuje wielokąt foremny o podanej jako jej argumenty długości i ilości boków. Użyj jej do narysowania 7 kąta, 13 kąta oraz 21 kąta o boku długości 30.

Zadanie 1.1.4

Napisz program który będzie rysował kolejne kwadraty, jeden wewnątrz drugiego, każdy o boku o połowę mniejszym niż poprzedni, do momentu kiedy ma to sens (bok kwadratu > 2) i wypisze na ekranie liczbę narysowanych kwadratów.

1.2 Przerywanie rysowania i kolory

Możliwe jest także wykonywanie ruchów żółwia bez rysowania oraz zmiana koloru:

- `penup()` - podniesienie mazaka (kolejne ruchy nie będą zostawiać śladów)
- `pendown()` - opuszczenie mazaka (kolejne ruchy będą zostawiać ślady)
- `pencolor(r, g, b)` - ustawia kolor RGB mazaka (wartości `r`, `g` oraz `b` określają składowe RGB koloru, czyli ilość czerwonego, zielonego i niebieskiego)
- `pencolor(x)` - ustawia kolor mazaka o nazwie określonej przez `x`, gdzie `x` jest napisem (np. `pencolor('red')` - czerwony, `pencolor('blue')` - niebieski, `pencolor('black')` - czarny)

Zadanie 1.2.1

Zmodyfikuj rozwiązanie zadania 1.1.3 tak aby n-kąty o parzystym n rysowała w kolorze czerwonym, a o nieparzystym w niebieskim

2 Rysowanie z użyciem współrzędnych

Rysowanie żółwia odbywa się w kartezjańskim układzie współrzędnych, a w chwili początkowej żółw znajduje się w punkcie (0,0). Możliwe jest pobranie aktualnej pozycji żółwia oraz nakazanie mu przejścia do wskazanego punktu:

- `pos()` - zwraca aktualną pozycję żółwia (wynik można pobrać do osobnych zmiennych `x, y = pos()` lub w postaci wektora `v = pos()` do elementów którego odwołujemy się `x = v[0], y = v[1]`)
- `goto(x, y)` - przesunięcie żółwia do punktu `x, y`
- `goto(v)` - przesunięcie żółwia do punktu wskazanego przez wektor (dwuelementową listę) `v`

Zadanie 2.0.1

Narysuj odcinek o początku w punkcie (-30, -30) i końcu w punkcie (0, 30).

Przy rysowaniu w taki sposób bardziej złożonych kształtów przydatne może się okazać użycie listy do

przechowywania punktów (czyli wykorzystanie „listy list”[1]):

```
punkty = [(10, 10), (20, 10), (50, 20)]

for p in punkty:
        print(p, "x =", p[0], "y =",
         ↪  p[1])
```

```
(10, 10) x = 10 y = 10
(20, 10) x = 20 y = 10
(50, 20) x = 50 y = 20
```

Do elementów listy możemy odwoływać się w obu kierunkach:

```
punkty = [(10, 10), (20, 10), (50, 20)]

print("pierwszy:",       punkty[0])
print("ostatni:",        punkty[-1])
print("przed ostatni:",  punkty[-2])
```

```
pierwszy: (10, 10)
ostatni: (50, 20)
przed ostatni: (20, 10)
```

Zadanie 2.0.2

Narysuj kwadrat o wierzchołkach w punktach (30, 30), (-30, -30), (30, -30), (-30, 30).

2.1 Obroty

Zauważ że polecenie `goto()` nie obraca żółwia, obrotami bezwzględnymi możemy manipulować przy pomocy poleceń:

- `heading(x)` - zwraca aktualny obrót żółwia
- `setheading(x)` - ustawia obrót żółwia na x stopni

Zadanie 2.1.1

Korzystając z polecenia `setheading()` zmodyfikuj rozwiązanie zadania 2.0.1, tak aby po narysowaniu odcinka orientacja żółwia była zgodna z kierunkiem wskazywanym przez ten odcinek.
Wskazówka: funkcję arctg znajdziesz w module math jako ***`math.atan()`***

3 Zadania dodatkowe

Zadanie 3.0.1

Narysuj trójkąt prostokątny, w którym przeciwprostokątna będzie koloru czerwonego, a przyprostokątne będą czarne.

Zadanie 3.0.2

Napisz funkcję która będzie rysować krzywą łamaną z wierzchołkami w podanych punktach. Zbiór punktów powinien być przekazywany jako argument funkcji w formie listy.

1. W poniższym przykładzie użyta jest tak naprawdę lista krotek, czyli lista niemodyfikowalnych list. Krotkę tworzymy `x = (1, 2, 3)` i od normalnej listy różni się tym iż operacje typu `x[1]=0`, czy też dodania elementu do krotki są niedozwolone.

Zadanie 3.0.3

Napisz funkcję, która będzie rysować spiralę Fibonacciego, czyli krzywą złożoną z kolejnych fragmentów łuków o długości 90 stopni i promieniu proporcjonalnym do kolejnych wyrazów ciągu Fibonacciego. Funkcja powinna przyjmować dwa argumenty określające odpowiednio:

1. ilość wyrazów ciągu użytych do rysowania spirali.
2. skalę, czyli to jaka wartość promienia, odpowiada wyrazowi ciągu o wartości 1