PROJEKT D

7. 11.2018

TERMIN ROZLICZENIA 19.11.2018

ALGORYTMY LOSOWE

PYTHON:
FUNKCJE, MODUŁY, TESTY MODUŁÓW

https://www.python-course.eu/python3_**functions**.php

https://www.python-course.eu/python3_**passing_arguments**.php

https://www.python-course.eu/python3_**global_vs_local_variables**.php

https://www.python-course.eu/python3_**lambda**.php

https://www.python-course.eu/python3_**modules_and_modular_programming**.php

Funkcje

Składnia funkcji:

```
def nazwa_funkcji (lista_argumentow_funkcji) :
    dokumentacja_funkcji
    blok_instrukcji
    return wynik_funkcji
```

tzw. *docstring:*
""" Opis działania .
Opis użycia .
Opis testów. """
Opis ten jest dostępny:
nazwa_funkcji.__doc__

Zauważ, argumenty funkcji nie mają typu!

```
def times(x, y):
    """Zwraca iloczyn argumentów."""
    return x * y

times(2, 3)          # 6
times(2, 3.14)       # 6.28
times("Bum!", 3)     # "Bum!Bum!Bum!"
```

Przy okazji: **Rekurencja Fibonacciego**:

$$F(n) = \begin{cases} 0 & dla\ n = 0 \\ 1 & dla\ n = 1 \\ F(n-1) + F(n-2) & dla\ n > 1 \end{cases}$$

0_funkcje.py

Przesyłanie argumentów do funkcji

Dowolna lista argumentów

```
def print_multiple_items(separator, *arguments): # arguments to krotka
    """Sklejenie napisów podanym separatorem."""
    print( separator.join(arguments) )


print_multiple_items("_", "ab", "cde", "f")          # ab_cde_f
```

1_funkcje.py

Moduły

Podstawowe (użyteczne) moduły:

- numpy : do obliczeń naukowych
- matlibplot.pyplot: biblioteka wykreślania 2D obrazów.
- random: biblioteka implementacji pseudo-losowych generatorów o różnych rozkładach

2_main.py

```
import fibonacci
print (fibonacci.rec_fib(8),fibonacci.iter_fib(8))  #funkcje wywołujemy z nazwą modułu


from  fibonacci  import  rec_fib , iter_fib        # importujemy wybrane funkcje
print (rec_fib(8),  iter_fib(8))                   #wywołujemy  bezpośrednio nazwy
```

fibonacii.py

5_plot.py

funkcje zip(), map(), filter(), wyrażenie lambda,

Funkcja map(function, sequence) - funkcja przetworzy po kolei każdy element listy *sequence* zgodnie z *function*. Funkcja map() zwraca tzw. generator.

3_map.py

Funkcja zip(seq1 [, seq2, ...]) zwraca jedną listę krotek utworzoną z kolejnych elementów sekwencji

Funkcja filter(function, sequence) - z listy *sequence* pozostaną elementy, dla których funkcja *function* zwróci True.

4_filter.py

lambda jest wyrażeniem (*lambda expression*), a nie instrukcją, dzięki czemu może pojawić się w miejscu niedostępnym dla instrukcji def.
Lambda zwraca obiekt funkcja, do którego można opcjonalnie przypisać nazwę.
Ciałem lambda jest pojedyncze wyrażenie, a nie blok instrukcji.

Składnia wyrażenia:

lambda lista_argumentow **:** wyrażenie

```
suma  =  lambda x, y : x+y
ciag  = map(rec_fib, [1,2,5,8,13])
```

Zakresy zmiennych

Każda zmienna żyje w określonej przestrzeni nazw. Kiedy mówimy o szukaniu wartości nazwy w odniesieniu do kodu, do przestrzeni nazw odnosi się zakres (scope).
Funkcje definiują zakres lokalny, a moduły zakres globalny.
Rozwiązywanie konfliktów w zakresie nazw w Pythonie nazywane jest regułą LEGB (local, enclosing, global, builtin)

- local - zakres lokalny
- enclosing - zakres lokalny instrukcji *def* lub wyrażeń *lambda* zawierających daną funkcję [działa przy funkcjach zagnieżdżonych]
- global - zakres globalny (moduł)
- builtin - zakres wbudowany

6_zakresy.py

PROJEKT D 5. 11.2018

Zaimplementować następujące funkcje:

1. Funkcja zwraca współrzędne kartezjańskie punktu leżącego w jednostkowej kuli. Rozważyć kule w przestrzeni 2-,3-,…,k- wymiarowej.
   Dla przypadku 2-wymiarowego wyznaczyć rozkład rozkład gęstości wylosowanych punktów. Czy jest jednostajny? Uzasadnić.

2. Funkcja zwraca współrzędne biegunowe (r, θ) punktu leżącego w jednostkowym okręgu. Wyznaczyć rozkład gęstości zestawu wylosowanych punktów. Czy jest jednostajny? Uzasadnić.

3. Wylosować zestaw (1000) motywów 8-elemetowych, w których prawdopodobieństwo pojawienia się danego nukleotydu na danym miejscu określa następująca macierz:

$$\text{Profile} = \begin{matrix} 0.2 & 0.1 & 0.5 & 0.1 & 0.1 & 0.4 & 0.2 & 0.7 \\ 0.4 & 0.6 & 0.3 & 0.3 & 0.1 & 0.4 & 0.0 & 0.2 \\ 0.3 & 0.3 & 0.1 & 0.5 & 0.6 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.1 & 0.2 & 0.1 & 0.6 & 0.0 \end{matrix}$$

7_projekt.py

Wszystkie funkcje powinny być oddzielnymi modułami z dobrze przygotowaną sekcją testów.