

Instytut Informatyki i Matematyki Komputerowej UJ
opracowanie: Ewa Matczyńska, Jacek Śmietański

DNA – RNA – białko

1. Wprowadzenie do Pythona

Python to interpretowany, interaktywny język programowania stworzony przez Guido van Rossuma w 1990. Nazwa pochodzi od Monty Pythona.

Główne cechy:

- dynamiczny system typów - zmienne nie mają przypisanych na sztywno typów, typ zmiennej wynika z wartości jaką dana zmienna przechowuje;
- automatyczne zarządzanie pamięcią - garbage collector;
- wszystko jest obiektem: system typów w pythonie jest silnie powiązany z systemem klas, klasa może dziedziczyć z dowolnego typu. Można więc dziedziczyć klasy z napisów czy słowników, a nawet z liczb całkowitych, możliwe jest dziedziczenie wielokrotne;
- wszystkie wartości przekazywane są przez referencję;
- bloki kodu są wydzielane przez wcięcia.

Na zajęciach będziemy korzystać również z pakietu biopython: <http://biopython.org/>

1. Podstawowe typy danych:

(a) logiczny

(b) liczby : int, float, long i complex np.

```
compl = 3+2.7j
```

(c) string

```
str = 'Hello World!'
```

(d) lista np.

```
list = ['banana', 'apple', 'mango']
```

(e) krotka, ang. *tuple* - w pewnym stopniu podobna do listy, ale dostęp do jej elementów jest szybszy, jest niemodyfikowalna

```
tuple = ('abcd', 786 , 2.23, 'john', 70.2)
```

(f) słownik np.

```
dict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
```

2. Instrukcja warunkowa

```
if expression:
    statement(s)
else:
    statement(s)
```

```
if expression1:
    statement(s)
```

```
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

```
if n>0:
    n+=2
else:
    n-=2
```

3. Pętla while:

```
while expression:
    statement(s)
```

```
while n>0:
    n-=1
```

4. Pętla for

```
for iterating_var in sequence:
    statement(s)
else:
    statement(s)
```

```
for n in range(1, 10):
    print(n)
else:
    print("done")
```

```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
    print('Current fruit :', fruit)
```

5. Definiowanie funkcji

```
def functionname(parameters):
    ...
    return [expression]
```

```
def fibonacci(n):
    a, b = 0, 1
    while a < n:
        print(a)
```

```
        a, b = b, a + b
    return b
```

Zadanie 1:

Co wypisuje wskazana wyżej funkcja „fibonacci” i jaką wartość zwraca?

6. Importowanie modułów:

```
import module1[, module2[,... moduleN]

---

import math
from math import sin, cos
```

7. Podstawowe operacje na plikach:

```
with open(file_name [, access_mode][, buffering])) as file_object:
    file_object.write(string)
    file_object.read([count])
    file_object.readline()

---

for line in file_object:
    print(line)
```

8. Standardowe wejście i wyjście

```
import sys
parameter = sys.argv[0]
s = sys.stdin.readline()
name = input("What is your name? ")
print(name)
```

9. „Hello world”

```
print("Hello bio world!")
w funkcji:
```

```
def hello():
    print("Hello bio world!")
```

10. Proste operacje na stringach, listach i słownikach

```
str = 'AGGTTACT '
print(str)           # complete string
print(str[0])        # first character of the string
```

```

print(str[2:5])          # characters starting from 3rd to 6th
print(str[2:])           # string starting from 3rd character
print(str[-1])           # last character
print(str[0:len(str):2]) # starting from first to last character with step 2
print(str[::-1])         # reverse string
print(str * 2)           # string two times
print(str + "TEST")      # concatenated string
print(len(str))          # length of string
print(min(str))          # smallest item
print(max(str))          # largest item
print(str.index("G"))     # index of the first occurrence of G in string
print(str.count("T"))     # total number of occurrences of T in string
print(str.rstrip())      # trailing whitespaces removed
print(str.split('A'))     # split using a delimiter
#i dużo innych

---

list = [ 'ACCTGC', 786 , 3.14, 'Adenine' ]
tinylist = [123, 'DNA']

print(list)              # complete list
print(list[0])           # first element of the list
print(list[1:3])         # elements starting from 2nd to 4th
print(list[2:])          # elements starting from 3rd element
print(tinylist * 2)       # list two times
print(list + tinylist)   # concatenated lists
print(list.append('1234')) # appends new element
print(list.insert(2, "new")) # inserts new element
print(list.extend(["two", "elements"])) # extends with elements

---

complementary_bases = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
print(complementary_bases['A']) # value for 'A' key
print(complementary_bases)     # complete dictionary
print(complementary_bases.keys()) # all the keys
print(complementary_bases.values()) # all the values

#adds pair to the dictionary:
complementary_bases['human DNA length in bp'] = 3.2*10**9

```

Zadanie 2:

Niech a będzie dowolną sekwencją DNA (ciąg niezerowej długości, składający się z liter A, C, G i T), np.: $a = \text{'TATATAAAAAAAGGGGGAT'}$.

Wypisz:

- ciąg w odwróconej kolejności;
- pierwsze pięć nukleotydów w sekwencji
- co trzeci nukleotyd
- kolejne fragmenty ciągu pogrupowane po trzy znaki w jednym wierszu

2. Transkrypcja, translacja

Na platformie <http://rosalind.info> znajduje się kilka zadań nawiązujących do kluczowych procesów zachodzących w organizmach żywych.

Zadanie 3: (4 pkt)

Zarejestruj (zaloguj) się na portalu <http://rosalind.info> i dołącz do klasy „Bioinformatyka 2018”. Wykonaj zadania oznaczone dzisiejszą datą.