

## Ciągi znaków(łańcuchy)- ang. string

Python posiada szereg wbudowanych metod do obsługi tekstu. M.in. STRING.

Dla tego typu danych możemy:

- obliczać długość łańcucha – len(łańcuch),
- zliczyć wystąpienie jakiegoś znaku bądź frazy – łańcuch.count(znak),
- zamienić jeden znak bądź frazę na inny znak bądź frazę: łańcuch.replace(znak\_stary, znak\_nowy),

### 1.Stworzenie łańcucha

```
dna="TGGAAGATTATATCTAATATCCTCTCTATGGTGGGGTTTAGTAGGGTTGTCATTAAGAAT"  
print dna
```

Efekt jest oczywiście następujący:

```
TGGAAGATTATATCTAATATCCTCTCTATGGTGGGGTTTAGTAGGGTTGTCATTAAGAAT
```

```
>>>
```

### 2a. Zliczenie wystąpienia znaku „T” w łańcuchu:

```
print dna.count("T")
```

```
>>>
```

Efekt:

```
23
```

```
>>>
```

### 2b. Zliczenie wystąpienia frazy „GGTT” w łańcuchu:

```
print dna.count("GGTT")
```

Efekt:

```
2
```

```
>>>
```

### 3. Zamiana znaku „T” na „U”:

```
print dna.replace("T", "U")
```

Efekt:

```
>>>
```

```
UGGAAGAUUAUAUCUAAUAUCCUCUCUAUGGUGGGUUUAGUAGGGUUGUCAUUAAGAAU
```

### 4. Pobranie tylko wybranego znaku z łańcucha (0- to indeks pierwszego znaku w łańcuchu, 1- drugiego...itd):

```
print dna[0]
```

Efekt:

```
>>>
```

```
T
```

```
>>>
```

### 5. Zliczenie długości łańcucha:

```
dna = "TGGAAGATTATATCTAATATCCTCTCTATGGTGGGGTTTAGTAGGGTTGTCATTAAGAAT"
```

```
print len(dna)
```

Efekt:

```
60
```

```
>>>
```

### Polecenia:

Proszę sprawdzić do czego służy metoda:

- upper()

-Lower()

### Zadania do wykonania:

Za pomocą interpretera Python wykonaj na sekwencji

```
DNATGGAAGATTATATCTAATATCCTCTCTATGGTGGGGTTTAGTAGGGTTGTCATTAAGAAT
```

Następujące operacje:

1. Zmiennej o nazwie „dna” przypisz wartość w postaci powyższej sekwencji.
2. Oblicz długość sekwencji przy użyciu odpowiedniej funkcji.
3. Oblicz, ile razy w sekwencji występuje każda z zasad.
4. Oblicz, ile razy występuje sekwencja „GG”.
5. Oblicz, ile razy występuje sekwencja „TAT”.
6. Oblicz, ile razy występuje sekwencja „ATA”.
7. Przekształć sekwencję DNA w mRNA, zamieniając tyminę na uracyl , a wynik tego działania przypisz nowej zmiennej „mrna”.
8. Zbadaj, ile kodonów fenyloalaniny (UUU lub UUC) znajduje się w otrzymanej sekwencji mRNA ?
9. Zbadaj, ile kodonów leucyny (UUA, UUG, CUU, CUC, CUA lub CUG) znajduje się w sekwencji mRNA ?

### LISTY

Listy w języku Python to uporządkowane zbiory dowolnych obiektów, włączając w to inne listy. Elementy list mogą być dowolnie wstawiane, usuwane i podmieniane na inne. Listy tworzone są jako serie obiektów oddzielone przecinkami i zawarte w prostokątnych nawiasach, np. możemy zdefiniować listę zasad i ją wywołać podobnie jak dla łańcuchów:

```
zasady=["A", "C", "G", "T"]
```

```
print zasady
```

Efekt:

```
['A', 'C', 'G', 'T']
```

```
>>>
```

Możemy dla list:

- |  |   |
|--|---|
| • Dopisać na końcu jakiś element: metoda               | <code>łańcuch.append(znak lub łańcuch)</code> |
| • Usunąć jakiś element z list: metoda                  | <code>łańcuch.remove(znak lub łańcuch)</code> |
| • Wstawić na określoną pozycję w liście jakiś element: | <code>łańcuch.insert(indeks,znak)</code>      |
| • Zliczyć wystąpienie jakiegoś element                 | <code>łańcuch.count(znak)</code>              |
| • Zliczyć długość listy                                | <code>len(łańcuch)</code>                     |
| • Odwrócić elementy w listach                          | <code>łańcuch.reverse()</code>                |
| • Posortować listę                                     | <code>łańcuch.sort()</code>                   |

### 1. Dołączenie elementu na końcu listy:

```
zasady.append("U")  
  
print zasady
```

efekt:

```
>>>  
  
['A', 'C', 'G', 'T', 'U']  
  
>>>
```

### 2. Usunięcie elementu „U” na liście:

```
zasady.remove("U")  
  
print zasady
```

Efekt:

```
['A', 'C', 'G', 'T']
```

### 3. Wstawienie znaku „U” na indeksie nr 2 a więc pozycji nr 3:

```
zasady.insert(2, "U")  
  
print zasady
```

Efekt:

```
['A', 'C', 'U', 'G', 'T']
```

### 4. Zliczenie długości listy:

```
print len(zasady)
```

Efekt:

```
5  
  
>>>
```

### 5. Zliczenie „U” w liście:

```
print zasady.count("U")
```

Efekt:

```
1  
  
>>>
```

### 6. Potraktowanie łańcucha znaków jako listy:

```
print list(dna)
```

Efekt:

```
['T', 'G', 'G', 'A', 'A', 'G', 'A', 'T', 'T', 'A', 'T', 'A', 'T', 'C', 'T', 'A', 'A',  
'T', 'A', 'T', 'C', 'C', 'T', 'C', 'T', 'C', 'T', 'A', 'T', 'G', 'G', 'T', 'G', 'G', 'G',  
'G', 'T', 'T', 'T', 'A', 'G', 'T', 'A', 'G', 'G', 'G', 'T', 'T', 'G', 'T', 'C', 'A', 'T',  
'T', 'A', 'A', 'G', 'A', 'A', 'T']  
  
>>>
```

### 7. Odwrócenie elementów w liście:

```
zasady.reverse()  
  
print zasady
```

Efekt:

```
['T', 'G', 'U', 'C', 'A']  
  
>>>
```

### 8. Posortowanie listy:

```
zasady.sort()  
  
print zasady
```

Efekt:

```
['A', 'C', 'G', 'T', 'U']  
  
>>>
```

### 9. Zastąpienie w liście każdej frazy „tymina” na „uracyl”

```
zasady=["guanina","cytozyna","adenina","tymina"]  
  
for x,element in enumerate(zasady):  
    if element=="tymina":  
        zasady[x]="uracyl"  
  
print zasady
```

Efekt:

```
['guanina', 'cytozyna', 'adenina', 'uracyl']  
  
>>>
```

### Zadania do wykonania:

1. Utwórz listę czterech zasad występujących w DNA ( stosując ich pełne nazwy, tj. „adenina”, „cytozyna” itd) w zmiennej o nazwie „zasady”.
2. W liście „zasady” dokonaj zamiany elementu „tymina” na „uracyl”.
3. Odwróć kolejność zasad.
4. Nowej zmiennej „dnarev” przypisz listę utworzoną z sekwencji DNA (zmienna „dna” z poprzedniego zestawu zadań) , a następnie odwróć listę.

# Zastosowanie języka Python do komunikacji z arkuszem kalkulacyjnym excel

Niezbędne są 2 pakiety:

- xlwt – do zapisu danych w arkuszu Excel (trzeba ściągnąć pakiet z internetu)
- xlrd - do odczytu danych z arkusza Excel (trzeba ściągnąć pakiet z internetu)

Kopie umieściłam na swojej stronie z przedmiotem:

Dla xlrd: <http://zsi.tech.us.edu.pl/~nowak/python/xlrd-0.7.1.win32>

Dla xlwt: <http://zsi.tech.us.edu.pl/~nowak/python/xlwt-0.7.2.win32>

## Przykład 1.

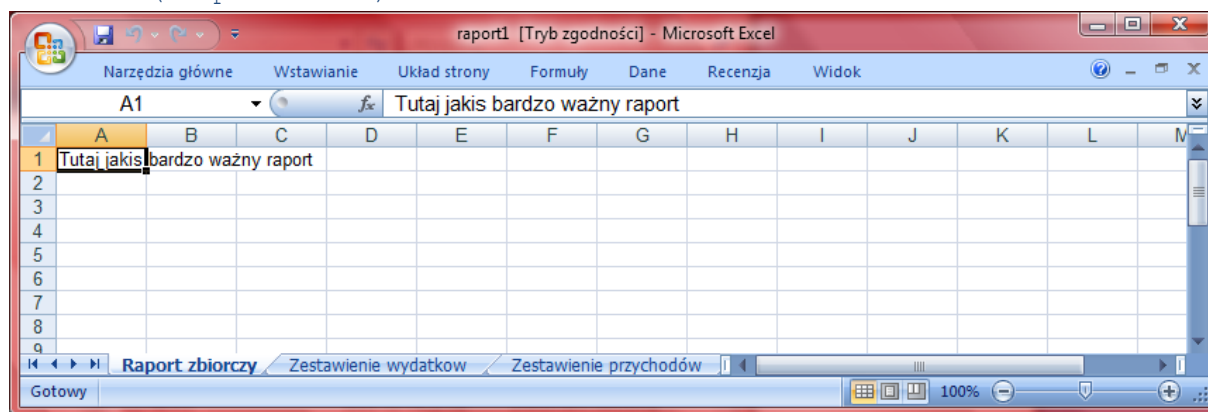
```
import xlwt

# kodowanie arkusza
book = xlwt.Workbook(encoding="cp1250")

#tworzymy dowolną ilość arkuszy (zakładek)
sheet1 = book.add_sheet("Raport zbiorczy")
sheet2 = book.add_sheet("Zestawienie wydatków")
sheet3 = book.add_sheet("Zestawienie przychodów")

#umieszczamy w nich dane
sheet1.write(0, 0, "Tutaj jakiś bardzo ważny raport")
sheet2.write(1, 10, "Wydaliśmy dużo")
sheet3.write(0, 2, "Ale zapłacili nam więcej")
sheet3.write(1, 2, "I jeszcze więcej nam zapłaca")
sheet3.write(2, 2, "Bedzie fajowo")

#zapisujemy do pliku
book.save("raport1.xls")
```



## Przykład 2

```
import xlrd
# otwarcie pliku
book = xlrd.open_workbook("raport.xls")

# dla każdego arkusza wyświetlamy zawartość pola 0,0
for sheet_name in book.sheet_names():
    arkusz = book.sheet_by_name(sheet_name)
    print arkusz.row_values(0)[0]
```

#Metoda `sheet_by_name` zwraca obiekt danego arkusza. Za pomocą tego obiektu możemy odczytać dane w nim zawarte. Metoda `row_values` zwróci wszystkie wartości dla podanego wiersza, lub zwróci wyjątek jeżeli żadne pole w danym wierszu nie zawiera danych.

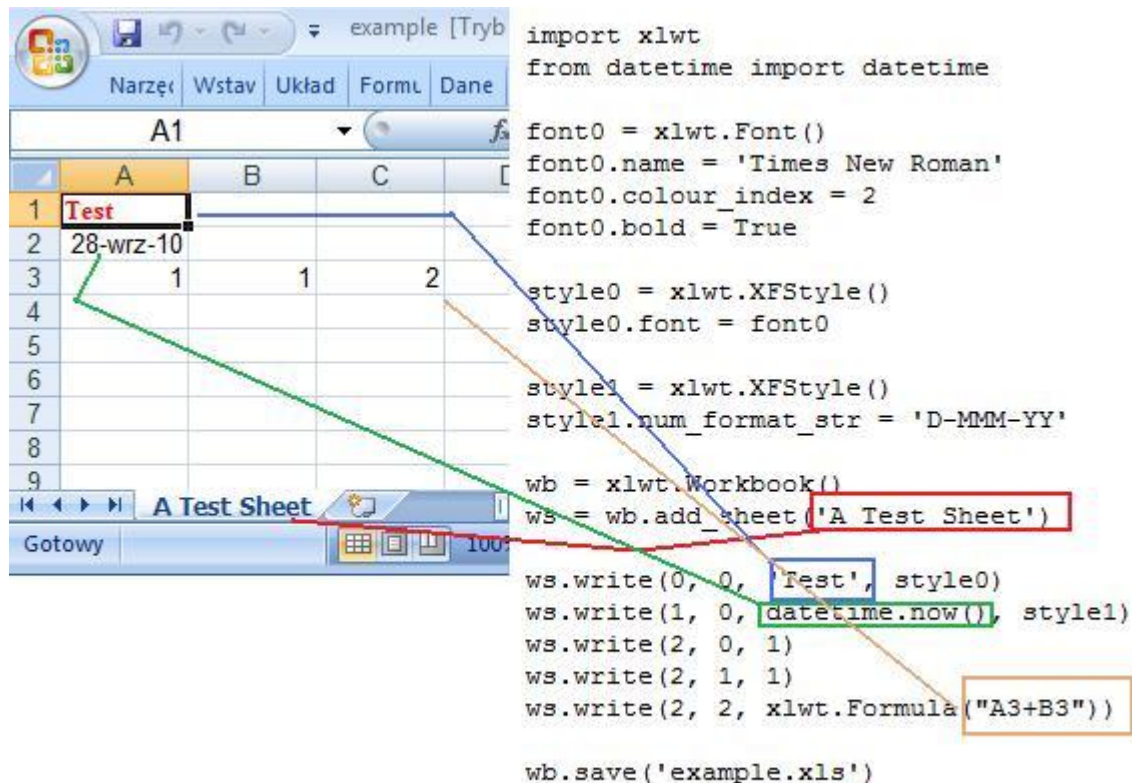
Efekt:

>>>

Tutaj jakis bardzo ważny raportXXXXXX

>>>

### Przykład 3 + efekt



```
import xlwt
from datetime import datetime

font0 = xlwt.Font()
font0.name = 'Times New Roman'
font0.colour_index = 2
font0.bold = True

style0 = xlwt.XFStyle()
style0.font = font0

style1 = xlwt.XFStyle()
style1.num_format_str = 'D-MMM-YY'

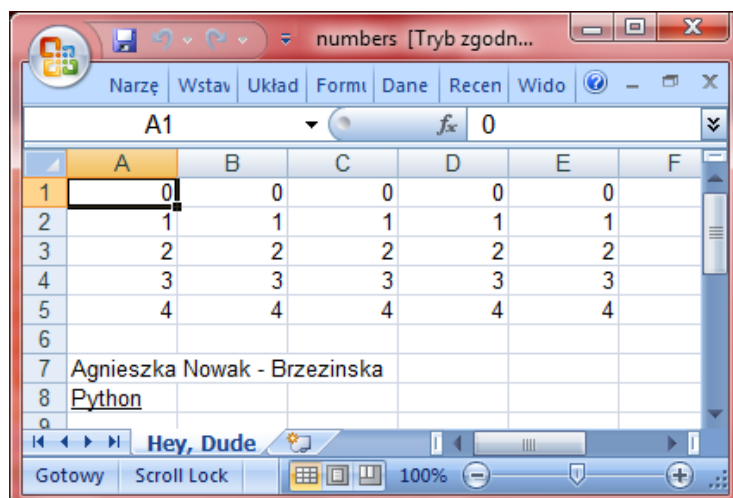
wb = xlwt.Workbook()
ws = wb.add_sheet('A Test Sheet')

ws.write(0, 0, 'Test', style0)
ws.write(1, 0, datetime.now(), style1)
ws.write(2, 0, 1)
ws.write(2, 1, 1)
ws.write(2, 2, xlwt.Formula("A3+B3"))

wb.save('example.xls')
```

### Zadanie do zrobienia:

Proszę spróbować osiągnąć podobny efekt:



	A	B	C	D	E	F
1	0	0	0	0	0	
2	1	1	1	1	1	
3	2	2	2	2	2	
4	3	3	3	3	3	
5	4	4	4	4	4	
6						
7	Agnieszka Nowak - Brzezinska					
8	Python					