Technical University of Lodz
Institute of Electronics

Algorytmy i struktury danych

6. Łańcuchy i przetwarzanie tekstu (I)

Łódź 2019

Ćwiczenie

Zapisz program jako **textManipulation.py;** Uruchom skrypt

```
1  # string and text manipulation 1
2
3
4  def countChar(text, myChar):
5      """Functions counts appearence of myChar in the text"""
6      count = 0
7      for c in text:
8          if c == myChar:
9              count += 1
10     print('Character %s appeares %d time(s) in the string "%s"' %(myChar, count, text))
11
12
13 def insertTextNTimes(text, pos, insertText, n):
14     """Function insert insertText n times into Text at a position pos"""
15     return text[:pos] + insertText*n + text[pos:]
16
17
18 def main():
19     sentence = 'Ala ma kota'
20     charToFind = 'a'
21     textToInsert = ' i Ola'
22
23     countChar(sentence, charToFind)
24     print(insertTextNTimes(sentence, 3, textToInsert, 4))
25
26
27 main()
```

Łańcuchy

- **String** (łańcuch) jest jednym z podstawowych typów danych w języku Python.
- **String** jest przykładem struktury danych.
- **String** jest także kolekcją danych, zorganizowanych w celu wydajnej realizacji dostępu i operacji na nich.
- **String** jest sekwencją znaków między cudzysłowami
  1. Pojedynczymi (‘)
  2. Podwójnymi (“)
  3. Potrójnymi – dla łańcuchów w wielu wierszach (“””)

```
In[1]: subject = ‘Algorithms and Data Structures’
In[2]: name =  “Grzegorz”
In[3]: surname = “Brzeczyszczykiewicz”
In[4]: multi_line_string = “””This is a very long string, that is spread across many, many lines, for
       example a description of some newly created function “””
```

Podstawowe operacje

Konkatenacja

```
In[5]: letters = “ab” + “cd”
In[6]: letters += ‘ab’
In[7]: “Ala” + ‘ ma ‘ + “kota!”
```

s1 = s1 + s2 Po łańcuchu s1 następuje s2
s1 += s2: skrót dla s1 = s1 + s2

Powtórzenie

```
In[8]: “Hello”*4
In[9]: “bye! “*7
In[10]: 10*“-*-?-*”
```

s*n Łańcuch s+s+…+s+s,n razy
n*s oznacza to samo co s*n

Akumulator łacuchów

```
In[11]: s = ‘a’
In[12]: for i in range(4):
In[13]:  s += ‘b’
```

<accumulator> = <string value>
pętla:
<accumulator> += <tekst który dodajemy>

Porównanie

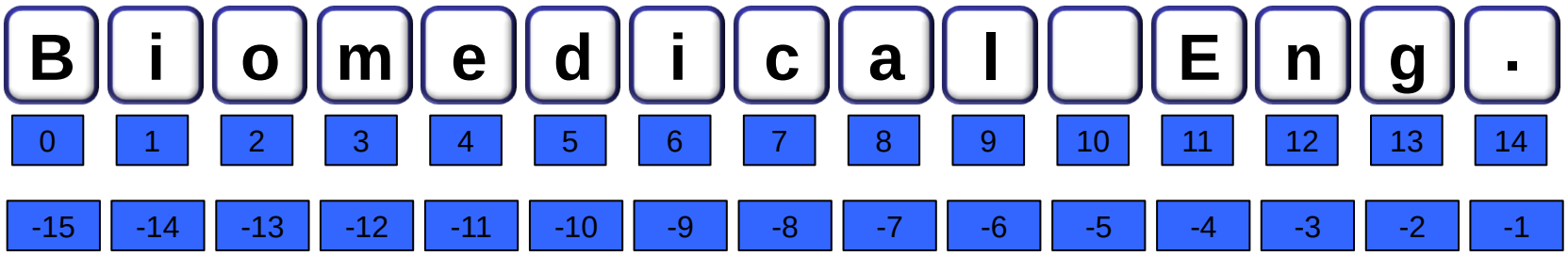
```
In[14]: s1 = ‘ala’; s2 = ‘ala’; s3 = ‘ALA’
In[15]: s1 == s2
In[16]: s1 == s3
```

s1 == s2 zwraca True
s1 == s3 zwraca False

Indeksowanie

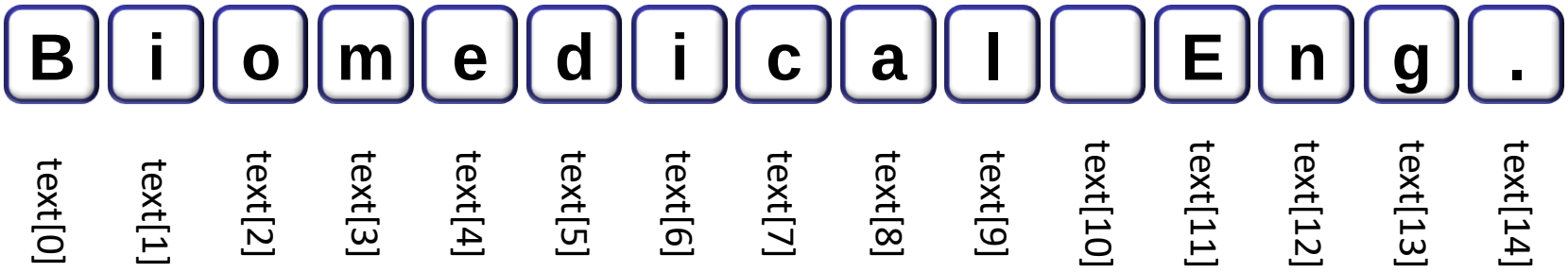
```
In[17]: text = 'Biomedical Eng.'
```

Zmienna *text* jest zapisywana w pamięci wg poniższego wzoru. Każdy znak jest zapisany w kolejnych komórkach pamięci.



Każda z liczb w dolnych wierszach jest nazywana **indeksem** znaku powyżej niej.

Indeksowanie



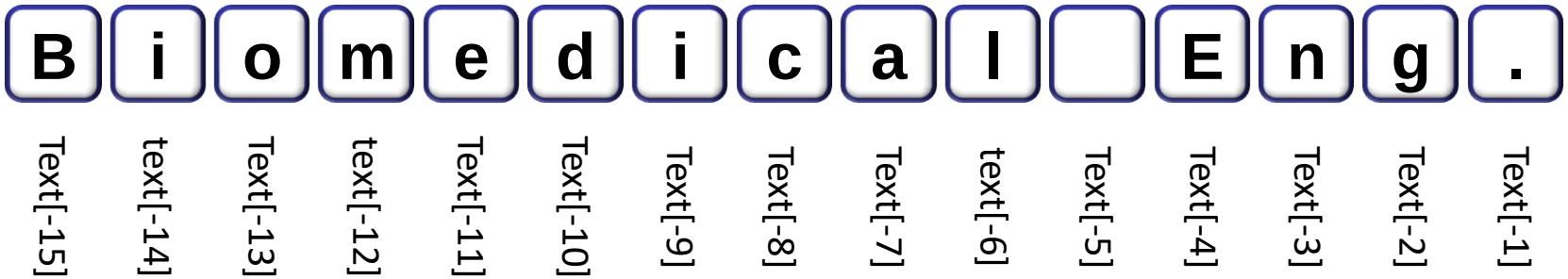
Wpisz poniższe polecenia i sprawdź wyniki:

```
In[18]: text[0] =    ?
In[19]: text[1] =    ?
In[20]: text[5] =    ?
In[21]: text[8] =    ?
In[22]: text[11] =   ?
```

string[i]: znak o indeksie i.

Indeksowanie łańcucha zwraca pojedynczy znak.

Indeksowanie



Wpisz poniższe polecenia i sprawdź wyniki:

```
In[23]: text[-1] =    ?
In[24]: text[-5] =    ?
In[25]: text[-7] =    ?
In[26]: text[-11] =   ?
In[27]: text[-15] =   ?
```

string[i]: znak o indeksie i.

Indeksowanie łańcucha zwraca pojedynczy znak.

Wycinanie (slicing)

Dostęp do zbioru kolejnych znaków łańcucha.

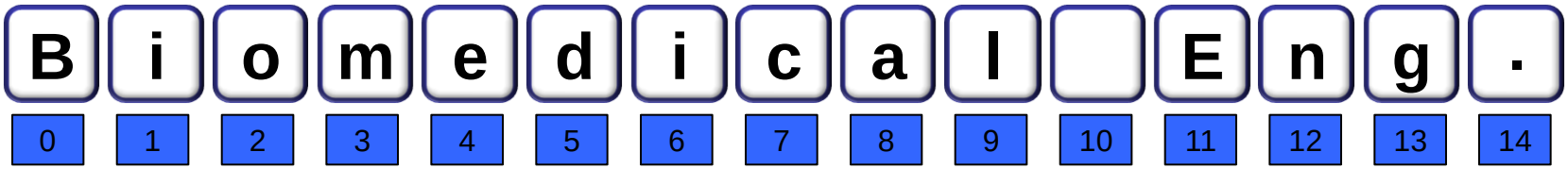
text[i:j] wycinek (slice) od i do j-1

B	i	o	m	e	d	i	c	a	l		E	n	g	.
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
In[28]: text[0:3] =      ?
In[29]: text[3:6] =      ?
In[30]: text[11:14] =    ?
In[31]: text[0:15] =     ?
```

Wycinek text[i:j] nie zawiera text[j]!

Wycinanie




```
In[32]: text[:3] =     ?
In[33]: text[3:] =     ?
In[34]: text[:] =      ?
```

text[:j]	Wycinek od początku do j-1
text[i:]	Wycinek od i do końca łańcucha
text[:]	Pełny wycinek - **kopia** zmiennej text

Trzeci, opcjonalny parametr k określa krok indeksu, różny od 1 (por. funkcja range())

```
In[34]: text[2:12:2] =     ?
In[35]: text[3::3] =       ?
In[36]: text[::2] =        ?
In[37]: text[::-1] =       ?
In[38]: text[9:2:-1] =     ?
In[39]: text[:5:-1] =      ?
In[40]: text[5::-1] =      ?
```

text[i:j:k] Wycinek od i do j-1 z krokiem k

Jeśli k < 0, indeks i jest nadal punktem startowym.
Jeśli i jest pominięte a k<0, początkiem jest prawy koniec.
Jeśli j jest pominięte a k<0, koniec znajduje się po lewej stronie (indeks 0)

W łańcuchu lub nie

Sprawdzenie, czy ciąg znaków x jest częścią s. Znaki tekstu x są lub nie są wycinkiem s.

x in s	True jeśli x zawiera się w s; jeśli się nie zawiera to False
x not in s	True jeśli x nie zawiera się w s; w przeciwnym razie False

In[41]: *s = 'ala ma kota'*

In[42]: *'a' in s =* ?
In[43]: *'ala' in s =* ?
In[44]: *'a ma k' in s =* ?
In[45]: *'alama' in s =* ?
In[46]: *'kot' in s =* ?

In[47]: *'a' not in s =* ?
In[48]: *'ala' not in s =* ?
In[49]: *' ma ' not in s =* ?
In[50]: *'alama' not in s =* ?
In[51]: *'ola' not in s =* ?

Liczba elementów i typ obiektu

Length and type of the string

```
In[52]: len(subject)
In[53]: len(text)
In[54]: type(name)
In[55]: type(student1)
```

len(string)

type(object)

Jaki znak ma indeks 15?

```
In[56]: text[15]
```

Pierwszym znakiem tekstu jest text[0], a nie text[1]. W programowaniu liczenie zaczyna się zwykle od 0, nie 1.

```
----> 1 text[15]
IndexError: string index out of range
```

Rodzaj błędu **Krótki opis przyczyny**

Zmiana znaków

Rozważmy łańcuch tekstowy

```
In[57]: day = 'Todai is Monday'
```

Chcemy poprawić pomyłkę

```
In[58]: day[4] = 'y'
```

Łańcuchy w języku Python są niezmienne (immutable)

```
----> 1 day[4]='y'
TypeError:  'str' object does not support item assignment
```

Obiekty typu ‚str' nie mogą być modyfikowane w ten sposób.

Trzeba utworzyć nowy łańcuch i przypisać go do zmiennej day

```
In[59]: day = day[:4] + 'y' + day[5:]
In[60]: print(day)
```

Pętla For

Jeśli chcemy przeglądać znaki łańcucha tekstowego po kolei, możemy utworzyć pętlę

```
In[61]: day = 'Today is Monday'
```

```
for <variable> in <string>: # pętla, znak po znaku
        <body>
```

Przykład

```
In[62]: for c in day:
            print(c)
```

Użyj dodatkowego parametru *end* w funkcji *print()*. Co się zmieniło?

```
In[63]: for c in day:
            print(c, end=' ')
```

Ćwiczenie: Utwórz następujący łańcuch:

*T * o * d * a * y * * i * s * * M * o * n * d * a * y **

Pętla For

Skanowanie znaków łańcucha z informacją o ich położeniu

```
In[64]: day = 'Today is Monday'
```

Przypomnienie składni

```
for <zmienna> in <sekwencja>:
        <ciało>
```

```
In[65]: for i in range(5):
            print(i, end=' ')

Out[65]: 0 1 2 3 4
```

Jak określić funkcję *range()*?

```
>>> for i in range(???):
        print i,
```

Zakres funkcji *range()* określa długość łańcucha znaków

```
In[66]: for i in range(len(day)):
            print (i, end=' ')

Out[66]: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

Dostęp do poszczególnych znaków według indeksu.

```
In[67]: for i in range(len(day)):
            print(day[i], end=' ')
Out[67]: T o d a y  i s  M o n d a y
```

Ćwiczenie: utwórz łańcuch

```
Out[68]: i s  M o n d a y
```

```
0 -> T
1 -> o
2 -> d
3 -> a
4 -> y
5 ->
6 -> i
7 -> s
8 ->
9 -> M
10 -> o
11 -> n
12 -> d
13 -> a
14 -> y
```

Ćwiczenie

6.0 Wypełnij przerwy w ciele funkcji deleteChar(text, myChar)

```
1   # string and text manipulation 1 - exercises
2
3   def deleteChar(text, myChar):
4       """Function deletes first appearance of a given character"""
5       for i in range(len(text)):
6           if [redacted] == myChar:
7               text1 = text[ ] + text[ ]
8               break #comment this line, what is the difference?
9           else:
10              text1 = text
11      return text1
```

Ćwiczenie

6.1 Napisz skrypt w języku Python, który wyświetli poniższe wzory (Wskazówka: użyj pętli).

A)

B)

Ćwiczenia

6.2 Określ wartości wyrażeń:

In[68]: subject = “Algorithms and Data Structures”

subject[0]
subject[5:]
subject[:5]
subject[::2]

subject[1:5]
subject[:-4]
subject[-15:]
subject[-20:20]

6.3 Napisz wyrażenia w języku Python z uzyciem zmiennej subject

In[69]: subject = “Algorithms and Data Structures”

Algorithms
Data
Structures
Alg

serutcurtS ataD dna smhtiroglA
AotsnDatcr #every third
suust amig #every third negative dir.
smhtiroglA #algorithms neg.

Ćwiczenia

6.4 Napisz skrypt **email.py**, który automatycznie konstruuje adres e-mail. Poproś użytkownika o podanie obu imion (piewrwszego i drugiego) i nazwiska. Skonstruuj adres według następującego wzoru: inicjaly@p.lodz.pl,

np. Jan Maria Nowak -> jmn@p.lodz.pl

6.5 Napisz skrypt który wypisuje element listy oraz jego długość (lista: tydzien=['Poniedziałek', 'Wtorek', 'Sroda', 'Czwartek', 'Piatek', 'Sobota', 'Niedziela'])

6.6 Napisz skrypt który podaje średnią liczbę litera przypadająca na każdy wyraz w liście.

6.7 Napisz skrypt stringAccumlator1.py który wykorzystuje akumulator łańcuchów (dodaje kolejne litery) aż do momentu wciśnięcia litery 'q'. Skrypt powinien wyświetlić zawartość akumulatora.

6.8 Napisz skrypt stringAccumlator2.py który wykorzystuje akumulator łańcuchów aż do momentu podania sekwencji znaków 'quit'. Skrypt powinien wyświetlić zawartość akumulatora.

Ćwiczenia

6.9 Napisz skrypt myStringFunctions.py. Wewnątrz skryptu napisz funkcję stringAccumlator(someText) , która zapisuje do akumulatora łańcuchów co 2 (lub co 3) znak z podanego tekstu someText.

6.10 W skrypcie myStringFunctions.py napisz funkcję stringReverse(someText) która zwraca zmienną tekstową someText w odwrotnej kolejności (od końca).

6.11 W skrypcie myStringFunctions.py napisz funkcję getSpaces(someText) która zwraca liczbę spacji występujących w zmiennej someText oraz indeks ostatniej spacji.

6.12 W skrypcie myStringFunctions.py napisz funkcję vowelsConsonantsAndSpaces(someText) która zwraca liczbę samogłosek, spółgłosek oraz spacji zawartych w zmiennej tekstowej someText

6.13 W skrypcie myStringFunctions.py napisz funkcję replaceSpaces(someText, newChar) która zastępuje spacje nowym znakiem newChar który podajemy jako argument funkcji.

Ćwiczenia

6.14 W skrypcie myStringFunctions.py napisz funkcję

replaceCharacter(someText, charToReplace, newChar) która zastępuje określony znak (charToReplace) w zmiennej tekstowej someText podanym jako argument funcji znakiem newChar

6.15 W skrypcie myStringFunctions.py napisz funkcję

deleteReplaceTripleCharacter(someText, charToChange, flag) która w zależności od wartości parametru funkcji flag zastępuje/usuwa/potraja podany jako argument funkcji znak (charToChange). Opis działania funkcji (Doc String) wygląda następująco:

Funkcja kasuje, zastępuję spacją lub potraja podany znak (literę)
w zależności od wartości parametru flag:
jeśli flag ma wartość 'd' -> kasujemy,
jeśli flag ma wartość 'r' - > zastępujemy spacją,
jeśli flag ma wartosć 't' -> potrajamy
Składnia funkcji:
1)deleteReplaceTripleCharacter('Biomedical','m','d') -> 'Bioedical'
2)deleteReplaceTripleCharacter('Biomedical','m','r') -> 'Bio edical'
3)deleteReplaceTripleCharacter('Biomedical','m','t') -> 'Biommmedical'
"""

Podsumowanie

1. String (łańcuch) jest sekwencją znaków zawartych wewnątrz znaków cudzysłowu
2. Łańcuchy mogą być dodawane, powtarzane, akumulowane i porównywane
3. Indeksowanie umożliwia uzyskanie pojedynczego znaku z łańcucha
4. W języku Python rozróżniamy indeksy dodatnie i ujemne
5. Aby uzyskać kilka znaków ze zmiennej tekstowej stosujemy wycinanie (slicing)
6. Wartości boolowskie są zwracane przy stosowaniu operatorów In and not In
7. Aby zmienić znak wewnątrz łańcucha, musimy stworzyć nowy łańcuch
8. Iterowanie przez łańcuch znaków umożliwia pętla for.

Literatura

Brian Heinold, Introduction to Programming Using Python, Mount St. Mary's University, 2012 (http://faculty.msmary.edu/heinold/python.html).

Brad Dayley, Python Phrasebook: Essential Code and Commands, SAMS Publishing, 2007 (dostępne też tłumaczenie: B. Dayley, Python. Rozmówki, Helion, 2007).

Mark J. Johnson, A Concise Introduction to Programming in Python, CRC Press, 2012.