



Technical University of Lodz
Institute of Electronics

Algorytmy i struktury danych

6. Łącuchy i przetwarzanie tekstu (I)

Łódź 2013





Ćwiczenie

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

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

```
>>> subject = 'Algorithms and Data Structures'
```

```
>>> name = "Grzegorz"
```

```
>>> surname = "Brzeczyszczykiewicz"
```

```
>>> 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

```
>>> letters = "ab" + "cd"  
>>> letters += 'ab'  
>>> "Ala" + ' ma ' + "kota!"
```

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

Powtórzenie

```
>>> "Hello"*4  
>>> "bye!"*7  
>>> 10*" - *-? - *"
```

$s*n$ łańcuch $s+s+...+s$, n razy
 $n*s$ oznacza to samo co $s*n$

Akumulator łańcuchów

```
>>> s = 'a'  
>>> for i in range(4):  
>>>     s += 'b'
```

$\langle \text{accumulator} \rangle = \langle \text{string value} \rangle$
loop:
 $\langle \text{accumulator} \rangle += \langle \text{string to add} \rangle$

Porównanie

```
>>> s1 = 'ala'; s2 = 'ala'; s3 = 'Ala'  
>>> s1 == s2  
>>> s1 == s3
```

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



Indeksowanie

```
>>> text = 'Biomedical Eng.'
```

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

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
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

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



Indeksowanie

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

Wpisz poniższe polecenia i sprawdź wyniki:

```
>>> text[0] = ?  
>>> text[1] = ?  
>>> text[5] = ?  
>>> text[8] = ?  
>>> text[11] = ?
```

string[i]: znak o indeksie i.

Indeksowanie łańcucha zwraca pojedynczy znak.



Indeksowanie

B	i	o	m	e	d	i	c	a	l		E	n	g	.
Text[-15]	text[-14]	Text[-13]	text[-12]	Text[-11]	Text[-10]	Text[-9]	Text[-8]	Text[-7]	text[-6]	Text[-5]	Text[-4]	Text[-3]	Text[-2]	Text[-1]

Wpisz poniższe polecenia i sprawdź wyniki:

```
>>> text[-1] = ?  
>>> text[-5] = ?  
>>> text[-7] = ?  
>>> text[-11] = ?  
>>> text[-15] = ?
```

string[i]: znak o indeksie i.

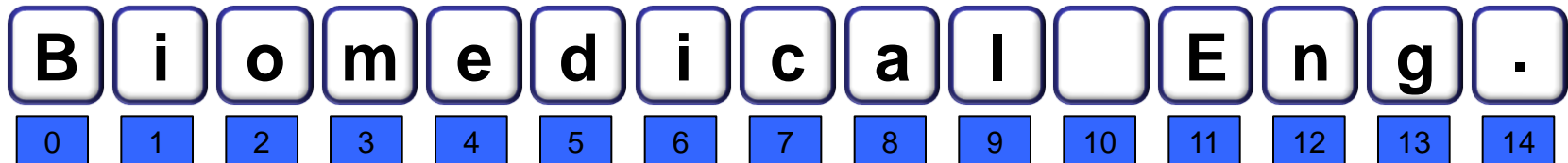
Indeksowanie łańcucha zwraca pojedynczy znak.



Wycinanie (slicing)

Dostęp do zbioru kolejnych znaków łańcucha.

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



```
>>> text[0:3] = ?  
>>> text[3:6] = ?  
>>> text[11:14] = ?  
>>> text[0:15] = ?
```

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



Wycinanie

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

```
>>> text[:3] = ?  
>>> text[3:] = ?  
>>> text[:] = ?
```

text[:j]	Wycinek od początku do j-1
text[i:]	Wycinek od i do końca łańcucha
text[:]	Pełny wycinek - <u>kopia</u> zmiennej text

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

```
>>> text[2:12:2] = ?  
>>> text[3::3] = ?  
>>> text[::2] = ?  
>>> text[::-1] = ?  
>>> text[9:2:-1] = ?  
>>> text[:5:-1] = ?  
>>> text[5::-1] = ?
```

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

```
>>> s = 'ala ma kota'
```

```
>>> 'a' in s = ?  
>>> 'ala' in s = ?  
>>> 'a ma k' in s = ?  
>>> 'alama' in s = ?  
>>> 'kot' in s = ?
```

```
>>> 'a' not in s = ?  
>>> 'ala' not in s = ?  
>>> 'ma' not in s = ?  
>>> 'alama' not in s = ?  
>>> 'ola' not in s = ?
```



Długość [len()]

Długość i typ zmiennej

```
>>> len(subject)
>>> len(text)
>>> type(name)
>>> type(student1)
```

Jaki znak ma indeks 15?

```
>>> text[15]
```

```
----> 1 text[15]
```

IndexError: string index out of range

Rodzaj błędu

Krótki opis przyczyny

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



Zmiana znaków

Rozważmy łańcuch tekstowy

```
>>> day = 'Todai is Monday'
```

Chcemy poprawić pomyłkę

```
>>> day[4] = 'y'
```

```
----> 1 day[4]='y'
```

```
TypeError: 'str' object does not support item assignment
```

Łańcuchy w języku Python są
niezmienne (immutable)

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

Trzeba utworzyć nowy łańcuch i przypisać go do zmiennej day

```
>>> day = day[:4] + 'y' + day[5:]
```

```
>>> print day
```



Pętla

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

```
>>> day = 'Today is Monday'
```

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

Przykład

```
>>> for c in day:  
    print c
```

Użyj przecinka po argumencie funkcji `print()`. Co się zmieniło?

```
>>> for c in day:  
    print c,
```

Ćwiczenie: Utwórz następujący łańcuch:

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



Pętle

Skanowanie znaków łańcucha z informacją o ich położeniu

```
>>> day = 'Today is Monday'
```

Przypomnienie składni

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

```
>>> for i in range(5):  
    print i,
```

```
>>> 0 1 2 3 4
```

Jak określić funkcję `range()`?

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

Zakres funkcji `range()` określa długość łańcucha znaków

```
>>> for i in range(len(day)):  
    print i,
```

```
>>> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

Dostęp do poszczególnych znaków według indeksu.

```
>>> for i in range(len(day)):  
    print day[i],
```

```
>>> T o d a y   i s   M o n d a y
```

Ćwiczenie: utwórz łańcuch

```
>>> 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[redacted] + text[redacted]
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:

```
>>> 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 użyciem zmiennej subject

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

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

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



Ćwiczenia

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

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

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

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



Ćwiczenia

6.12 W skrypcie myStringFunctions.py napisz funkcję

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

6.13 W skrypcie myStringFunctions.py napisz funkcję

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

*Funkcja kasuje, zastępuje spacją lub potrąja 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 wartość 't' -> potrąjamy*

Składnia funkcji:

- 1) deleteReplaceTripleCharacter('Biomedical', 'm', 'd') -> 'Bioedical'
- 2) deleteReplaceTripleCharacter('Biomedical', 'm', 'r') -> 'Bio edical'
- 3) deleteReplaceTripleCharacter('Biomedical', 'm', 't') -> 'Biommmmedical'



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.