Technical University of Lodz
Institute of Electronics

Algorithms and Data Structures

6. Strings and Text Manipulation I

Łódź 2019

Exercise

- Type in the program; Save it as **textManipulation.py;** Run the script

```python
# string and text manipulation 1


def countChar(text, myChar):
    """Functions counts appearence of myChar in the text"""
    count = 0
    for c in text:
        if c == myChar:
            count += 1
    print('Character %s appeares %d time(s) in the string "%s"' %(myChar, count, text))


def insertTextNTimes(text, pos, insertText, n):
    """Function insert insertText n times into Text at a position pos"""
    return text[:pos] + insertText*n + text[pos:]


def main():
    sentence = 'Ala ma kota'
    charToFind = 'a'
    textToInsert = ' i Ola'

    countChar(sentence, charToFind)
    print(insertTextNTimes(sentence, 3, textToInsert, 4))


main()
```

Strings

- **String** is a basic type in Python
- **String** is an example of data structure
- **String** is a collection of data organized to efficiently support a particular set of access methods and operations
- **String** is a sequence of characters inside quotation marks
  1. Single quotes (‘)
  2. Double quotes (“)
  3. Triple quotes – for multi-line strings (“””)

```
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 “””
```

Basic operations

Concatenation

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

s1 = s1 + s2 The string s1 followed by s2
s1 += s2 Shorthand for s1 = s1 + s2

Repetition

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

s*n The string s+s+…+s+s, n times
n*s The same as s*n

String Accumulators

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

<accumulator> = <string value>
loop:
<accumulator> += <string to add>

Comparison

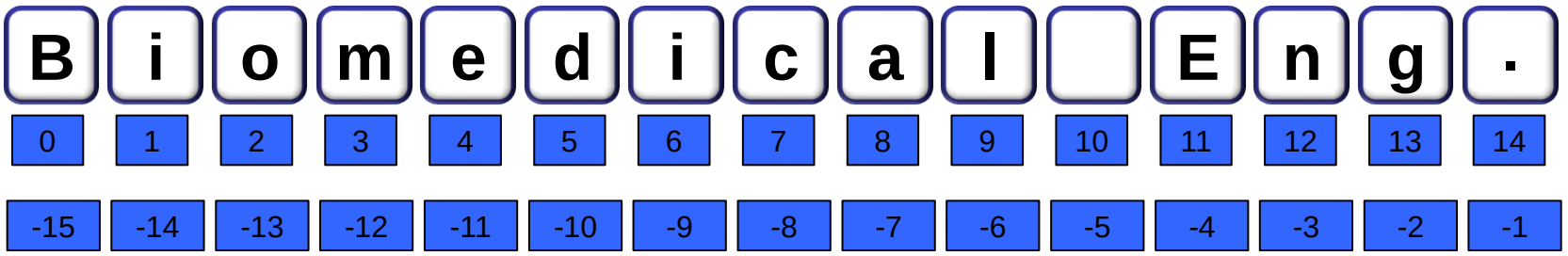
```
In[14]: s1 = ‘ala’; s2 = ‘ala’; s3 = ‘ALA’
In[15]: s1 == s2
In[16]: s1 == s3
```

s1 == s2 returns True
s1 == s3 returns False

Indexing

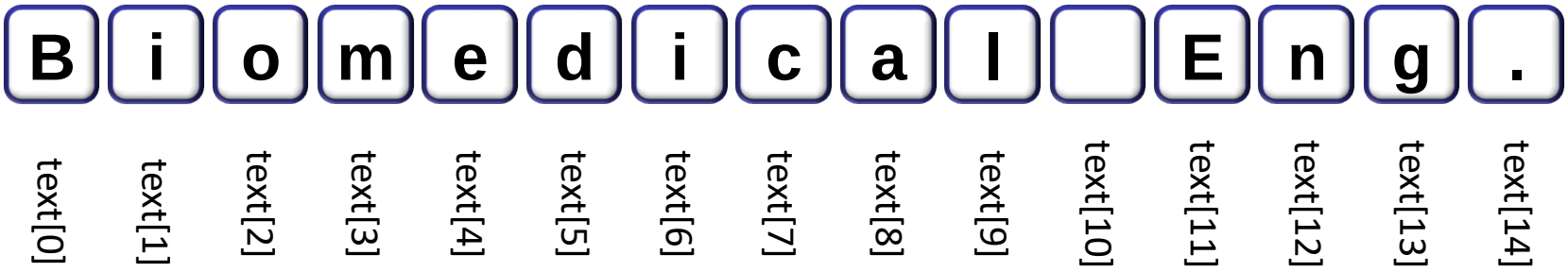
```
In[17]:  text = 'Biomedical Eng.'
```

The *text* variable is stored in a memory like below. Each, individual character is stored in consecutive memory location



Each number along the bottom row is called **index** of the character above it.

Indexing



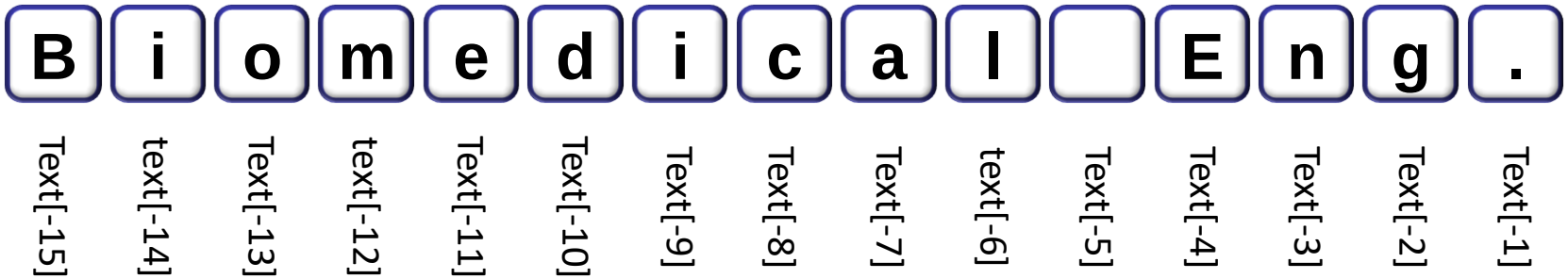
Get the letters at following positions (indices):

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

string[i] Character at index i.

Indexing a string always returns a single character

Indexing



Type in following commands and check the result:

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

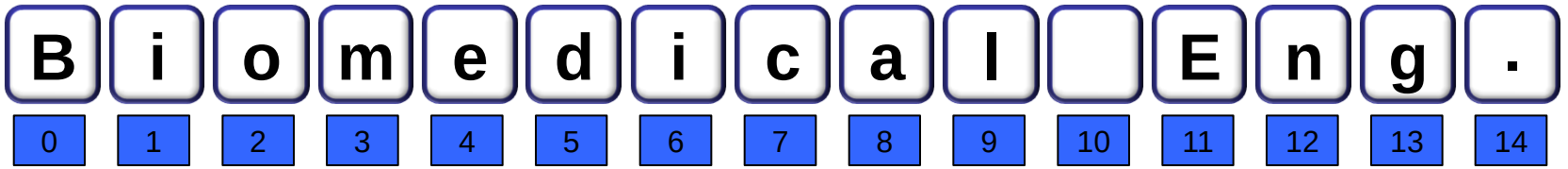
string[i] Character at index i.

Indexing a string always returns a single character

Slicing

Accessing a group of consecutive characters

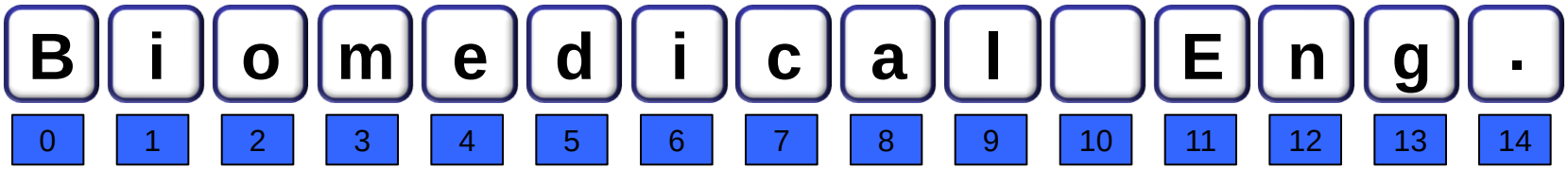
text[i:j] Slice from i to j-1




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

Does not include text[j]!

Slicing




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

text[:j]	Slice from beginning to j-1
text[i:]	Slice from i to the end
text[:]	Full slice, which creates a **copy** of s

Optional third parameter k specifies a stepsize other 1 (see the range() function)

```
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] Slice from i to j-1 with steep k

If k is negative, the index i will still is the starting point. If i is omitted with k<0, the beginning is considered the right end (index-1), and if j is omitted with a k<0, the end is at the left (index 0)

in and not in

To test whether or not a characters in x are a substring of s. Substring means that the characters of x occur as a consecutive slice of s

x in s	True if x is a substring of s; otherwise False
x not in s	True if x is not a substring of s; otherwise 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 =* ?

Length

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)

What character is at the index 15?

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

```
Out[56]: 1 text[15]
IndexError: string index out of range
```

The "type" of the error that occurred

Short message about why it occurred

The first character of text is text[0], not text[1]. Remember that in programming, counting usually starts at 0, not 1.

Changing characters

Suppose we have a string

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

Python strings are immutable

We want to repair spelling mistake

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

```
Out[58]: 1 day[4]='y'
TypeError:  'str' object does not support item assignment
```

The "type" of the error that occurred

Short message about why it occurred

We have to create a string and assign it to day

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

Looping

If we want to scan through a string one character at a time we can use a for loop

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

```
for <variable> in <string>: # loop over each character in the string
        <body>
```

The easiest way is as follows

(without holding the current location in the string)

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

Use *end* paremeter of *print()* function to change character at the end of printed sentence.

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

Task: create string like below:

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

Looping

If we want to scan through a string with information about current location

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

Let's recall the for loop syntax

```
for <variable> in <sequence>:
        <body>
```

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

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

How define a scope of a *range()* function?

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

The scope of a *range()* is the length of a string

```
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
```

How to get a single character form the string?

```
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
```

Task: create strings (below, and on the right):

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

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

Exercise

6.0 Fill in the red gaps in the body of the function 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
```

Exercises

6.1 Write a Python script to print a patterns like shown below
(Hint: use for loop and string repetition, use 10 stars).

A)

B)

Exercises

6.2 Determine the values of these expressions:

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 Crate Python expressions using variable subject that have these values:

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.

Exercises

6.4 Write a script **email.py** that builds e-mail address for a student. Ask the user for booth first and second name and surname. Generate e-mail address with the use of pattern: initials@p.lodz.pl, e.g. mmk@p.lodz.pl

6.5 Write a script that prints word and its lengths from a list (e.g. week=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])

6.6 Write a script that prints average number of letters per word from a list.

6.7 Write a script stringAccumlator1.py that accumulates characters until 'q' appear

6.8 Write a script stringAccumlator2.py that accumulates characters until string 'quit' appear

Exercises

6.9 Write a function stringAccumlator(someText) that accumulates every 2nd (or 3rd) character from someText

6.10 Write a function stringReverse(someText) that returns reversed someText

6.11 Write a function getSpaces(someText) that returns number of spaces and index of the last space

6.12 In the script *myStringFunctions.py* write a function vowelsConsonantsAndSpaces(someText) that returns number of vowels, consonants and spaces in the someText

6.13 In the script *myStringFunctions.py* create a function replaceSpaces(someText, newChar) that replaces the spaces with the newChar given as an argument.

Exercises

6.14 In the script *myStringFunctions.py* create a function replaceCharacter(someText, charToReplace, newChar) that replaces the specified char (charToReplace) in someText with the newChar given as an argument

6.15 In the script *myStringFunctions.py* create a function deleteReplaceTripleCharacter(someText, charToChange, flag) that depends on flag value replaces/delete/triples given charToChange. The help of the function is as follows:

""" Function deletes, replace with space or triple given letter (character)
depends on flag value:
d -> delete
s -> replace with space
t -> triple
Usage:
1)deleteReplaceTripleCharacter('Biomedical','m','d') -> 'Bioedical'
2)deleteReplaceTripleCharacter('Biomedical','m','s') -> 'Bio edical'
3)deleteReplaceTripleCharacter('Biomedical','m','t') -> 'Biommmedical'
"""

Summary

1. String is a sequence of characters inside quotation marks
2. Strings can be added, repeated, accumulated and compared
3. A indexing method is used to get individual character from a strong
4. Python strings can have positive and negative indexes
5. To get several characters from a string we use slicing
6. Boolean values are returned by In and Not In operators
7. To change a character inside string we have to create a new
8. To scan through a string we use for loop

Literature

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.