Co-funded by the Erasmus+ Programme of the European Union

SOFIA UNIVERSITY "ST. KLIMENT OHRIDSKI" EST. 1888

ICT-TEX

ICT-TEX course on Digital skills

Topic 6: Introduction to programming

(in Python)

The course is developed under Erasmus+ Program Key Action 2:
Cooperation for innovation and the exchange of good practices Knowledge Alliance

ICT IN TEXTILE AND CLOTHING HIGHER EDUCATION AND BUSINESS

Project Nr. 612248-EPP-1-2019-1-BG-EPPKA2-KA

Co-funded by the
Erasmus+ Programme
of the European Union

SOFIA UNIVERSITY
"ST. KLIMENT OHRIDSKI
EST. 1888

ICT-TEX

Contents

Foreword

- In this module you are going to learn about basics of programming
- The target programming language used is Python
  - One of the most popular programming languages in the world
  - Considered the best to start for beginners
  - Python is a high-level general purpose programming language
  - Has a wide variety of applications – from everyday automation of operations of a computer to machine learning and big data analytics

Fundamentals

- Each computer program executes a sequence of operations:
  - Input operations, for example – from keyboard or a file
  - Output operations, for example – to the computer screen
  - Data processing operations, for example – finding the average of a sequence of numbers
- Fundamental concept in programming is that data and information is stored into computer memory and is accessed via *variables*
- Variable is a unique name, across the program, of a specific storage of data

Basic Python features

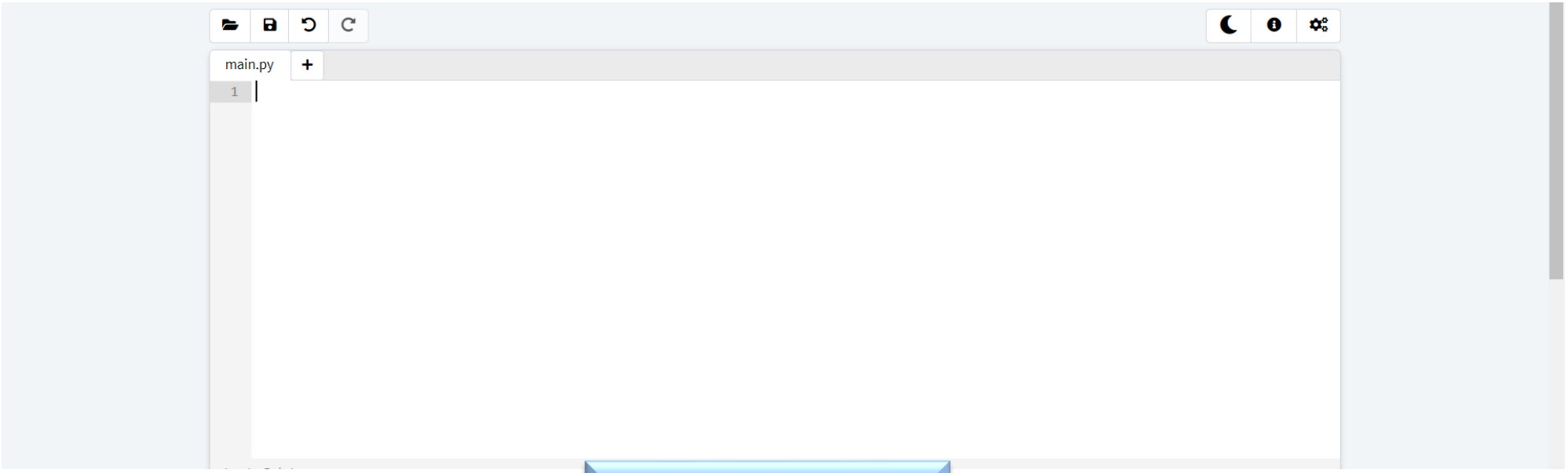
- Built-in high level data types: strings, lists, dictionaries, etc.
- Can be extended in other programming languages
- May be used for object-oriented programming
- Python is a representative of the so called dynamic-typed languages, which means that you don't need to bother about what exactly kind of information (like text or number) you store into the variables

Getting started

- In order to develop computer programs, you will need to set up the environment for the chosen programming language
- If you want to set up your computer to use Python follow the instructions given here
- For the purposes of this module, you do not need to install anything – we are going to use a web-based IDE (Integrated Development Environment) to get started with Python

Writing your first Python program

- Start the Python IDE: https://www.online-python.com/
- Clear all the initial code that appears in the "main.py" part of the window:



Your first Python program

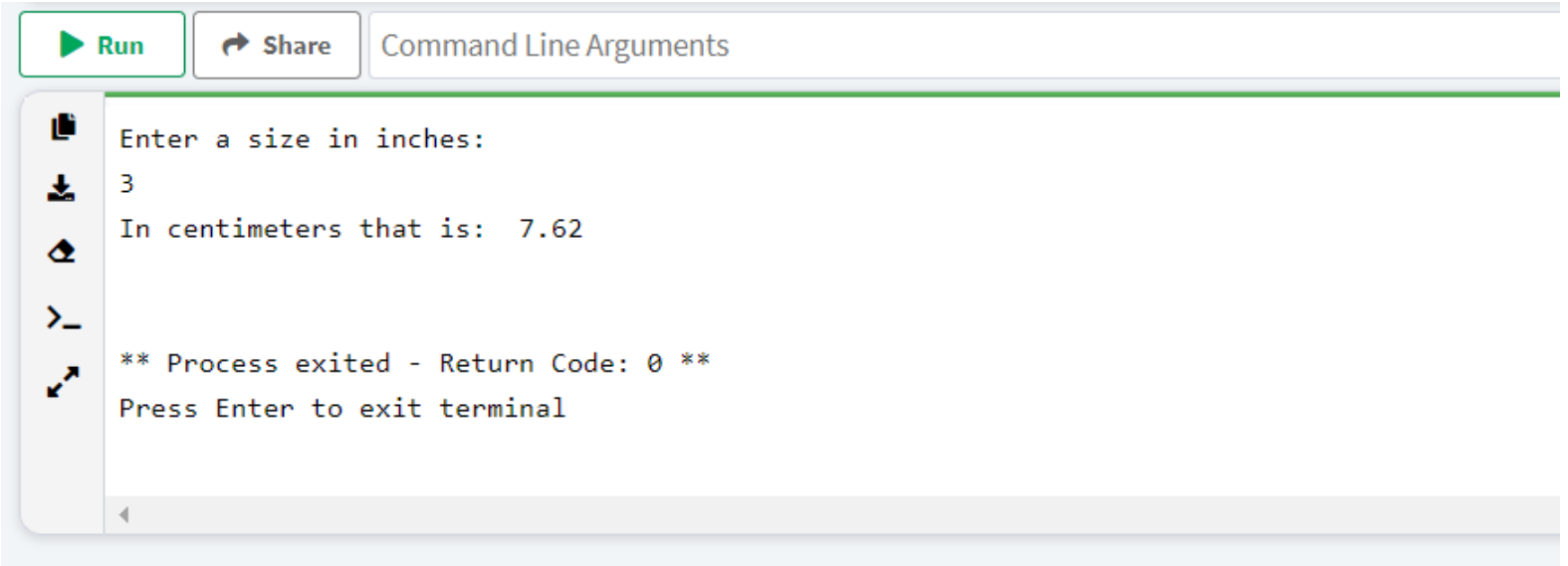
- Instead of the typical *"Hello world"* program we are going to implement a program that converts size measurement units
  - For example, inches into centimeters
  - Let's assume that for a conversion from inches into centimeters, the user should be prompted for a number
  - The output of the program should be the converted measurement

Your first Python program

- Type in the following code into the "main.py" window:

```
inches = eval(input('Enter a size in inches: '))
print('In centimeters that is: ', inches*2.54 )
```

- Click on Run



Arithmetic operations

- Python language supports these arithmetic operations:

Operation	Meaning	Example
+	Addition	5+6 = 11
-	Subtraction	8-3 = 5
*	Multiplication	7*3 = 21
/	Division (real numbers)	13/4 = 3.25
//	Division (integer numbers)	13//4 = 3 (throws away the decimal part of the calculation)
%	Remainder of integer division	13%4 = 1
**	Exponentiation (power of)	4**3 = 64

Program Input

- Input means the process of getting data into the computer memory, where it can be accessed by programs
- The basic usage of the **input** function is:
  - *<variable name>* = **input**(*message to user*)
    - Stores user input **as text** into the variable *<variable name>*
  - *<variable name>* = **eval**(**input**(*message to user*))
    - Stores user input **as a number** into the variable *<variable name>*

Program Output

- The **print** function requires parenthesis around its arguments.
- Anything inside quotes will (with a few exceptions) will be printed exactly as it appears.
  - Try to run the following **print**('Hello world') into the IDE
- However, in the following, the first statement will output 3+4, while the second will output 7.
  - **print**('3+4')
  - **print**(3+4)

Important things to have in mind

- **Case matters**. For Python, print, Print, and PRINT are all different identifiers. For now, lowercase is recommended as most Python statements are in lowercase
- Spaces are extremely important for beginning of lines. For example, this code will not work:

```
inches = eval(input('Enter a size in inches: '))
  print('In centimeters that is: ', inches*2.54 )
```

  - On the other hand, spaces in most other places don't matter.

Algorithms

- Each computer program implements an algorithm
- An algorithm in computer science can be defined as: *"A finite set of unambiguous programmatically-implementable instructions that, can perform a certain computation."*
- For example, our conversion algorithm had the following steps:
  - User input of a number (inches)
  - Calculation of the conversion from inches to centimeters
  - Output of the converted number (centimeters) onto the screen

Algorithms

- Basic constructs of algorithms are:
  - Instructions
  - Iterations implemented by loop operators)
  - Branches (implemented by conditional statements)
- We have already got acquainted with instructions, lets now delve into other two constructs

Conditional statements

- Conditional statements provides opportunity to check if something is ***True*** or ***False*** and depending on it to take different paths for execution of the program
  - Traditionally in programming *True* and *False* are called Boolean constants and are subject to Boolean algebra rules
- **if** is a conditional statement operator in Python, as well as in a large variety of other programming languages

Conditional statements

Expression	True when
if x == 3:	x is equal to 3
if x > 3:	x is greater than 3
if x < 3:	x is less than 3
if x >= 3:	x is greater than or equal to 3
if x <= 3:	x is less than or equal to 3
if x == 3:	x is equal to 3
if x != 3:	x is not equal to 3

Note the equality operator. It consists of two equals signs!

Example with conditional statements

- Let's consider a user wants a program that takes as input man's chest size in centimeters and outputs if it is suitable for clothing (i.e., T-Shirt) size of XS (less than 81 centimeters)

```
chestSize = eval(input("Enter chest size in centimeters: "))
if (chestSize < 81):
    print('You should wear XS size')
if (chestSize >= 81):
    print('You should wear larger than XS size')
```

More Boolean operators

- **and**, **or** and **not** keywords of Python language provide means to make more complex evaluation for deciding the branch statements
- This is called Boolean algebra
- Next slide shows more details about it

Boolean algebra

X	Y	result
True	True	True
False	True	False
True	False	False
False	False	False

AND

X	Y	result
True	True	True
False	True	True
True	False	True
False	False	False

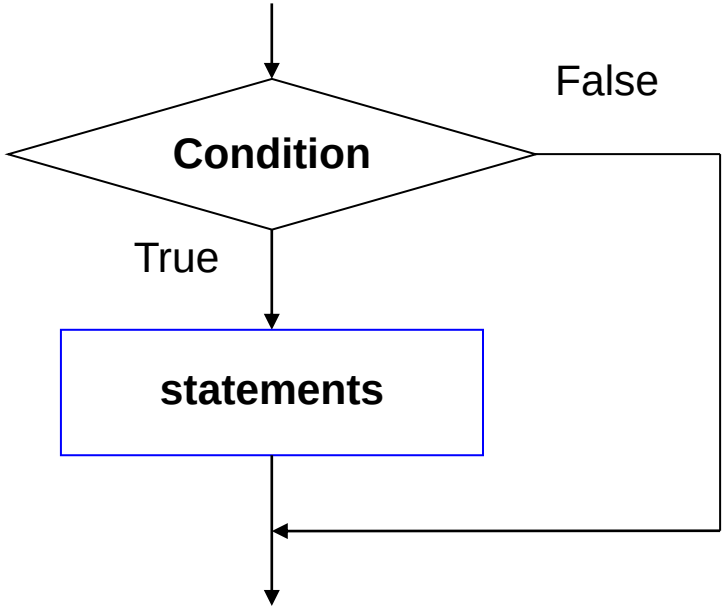
OR

X	result
True	False
False	True

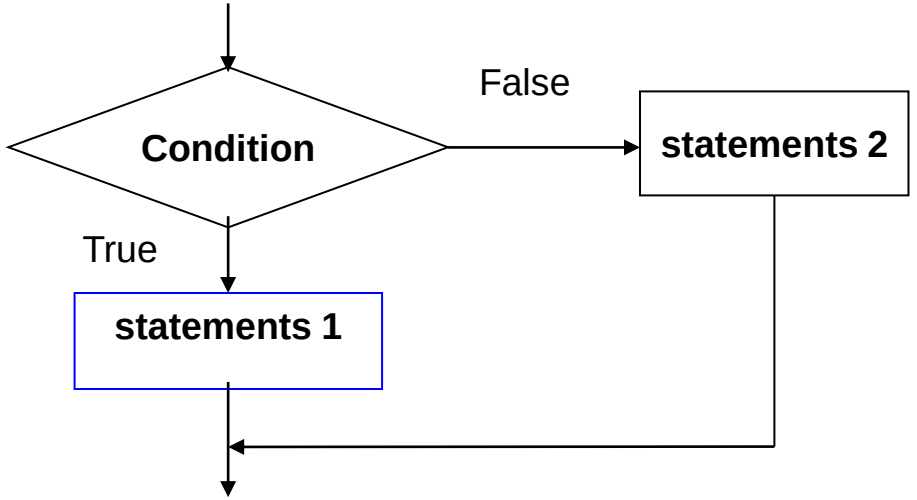
NOT

if-else operator

if operator



if-else operator



Example with conditional statements

- Let's now consider a user wants another program that also takes as input man's chest size in centimeters and outputs if it is suitable for clothing (i.e., T-Shirt) size of XL (between 114 and 125 centimeters)

```
chestSize = eval(input("Enter chest size in centimeters: "))
if (chestSize >= 114 and chestSize <= 125):
        print('You should wear XL size')
else:
    print ('You should wear different than XL size')
```

Conditional statements example

- Lets consider we want to classify clothing (i.e., T-Shirt) sizes, depending on the man's chest size in cm
  - Less than 86 – XS
  - From 87 to 93 – S
  - From 94 to 103 – M
  - From 104 to 115 – L
  - From 116 to 125 – XL
  - And so on

Clothing size calculation program

```
chestSize = eval(input("Enter chest size in centimeters: "))
if (chestSize <= 86):
    print('You should wear XS size')
elif (chestSize <= 93):
    print('You should wear S size')
elif (chestSize <= 103):
    print('You should wear M size')
elif (chestSize <= 115):
    print('You should wear L size')
elif (chestSize <= 125):
    print('You should wear XL size')
else:
    print('You should wear larger than XL size')
```

Loops

- Type the following program into the IDE and execute it

```
for i in range (30): print(i)
```

- The result should be printed numbers from 0 to 29 (a total count of 30 numbers)
- Operator **for** is called the loop operator and **i** is called – loop control variable
  - **i** is just a common name for the loop control variable. In fact, its name is not constrained
  - It is important to note that in programming counting of numbers usually start from zero

Loops

- Loops can also be nested, the following program will print 5 lines with numbers from 1 to 20 on each of them

```
for i in range (5):
    print()
    for j in range (20): print(j+1, end=', ')
```

- The **print**() with no parameters is used to print a new line
- The **end** parameter denotes an alternative to be printed instead of the new line

Loops

- range() can also take an interval range:

```
for j in range (4,20):
    print(j+1, end=", ")
```

- Note the indent in the beginning of the line. It is important as it marks a block for iterative execution. All statements after the for operator that are indented will be part of the loop

The **range**() function

- Range function returns sequences of numbers of a fixed stepping
  - It starts at 0 by default
  - Increments by 1 by default
  - Stops before the specified number

The **range**() function

- General syntax
  - **range**(start, stop, step)
- The parameters are
  - **start** (optional): An integer that denotes the start number for the range sequence.
  - **stop** (required): An integer that denotes the final number for the range sequence (actually the last number is stop-1).
  - **step** (optional): An integer that defines the step for the range range sequence.

The **range**() function – examples

Range	Returns	Description
range(4,7)	4,5,6	Starting from 1 to 6 with default step of 1
range(1,12,3)	1, 4, 7, 10	Starting from 1 to 11 with step of 3
range(12,1,-3)	12, 9, 6, 3	Starting from 12 backwards to 1 with step of 3

Finding the sum of elements

- Let's find the sum of the first 100 integer numbers

```
s = 0
for i in range(1,101):
    s = s + i
print('The sum is', s)
```

- Note setting the initial value of the s variable to zero

While loop

- Let's now assume that we want the user to enter a number that lies within a given interval (let's say between 70 and 200)
- One way to this is to use an iterative algorithm that continuously prompts the user for a number until they enter the desired value

```
num = 0;
while (num < 70 or num > 200):
    num = eval(input('Enter a size in cm (between 70 and 200)'))
print('You have entered', num)
```

While loop

- The expression after the **while** keyword is called a *stop condition.* When its value equals to **False** the loop stops its execution
- Beware – if by some reason the stop condition is never evaluated to false, the loop will become endless – it will never stop by itself

Endless loop

- The following loops will never stop, because their stop condition always evaluates to True

```
i=0
while True:
    print(i)
    i+=1
```

```
i = 0
while i < 4:
    print(i)
    i = i - 1
```

- However, sometimes (as in the left example) you may intentionally make the loop endless

Break statement

- The **break** statement can be used to stop the execution of a for or while loop before the loop is finished (the stop condition to become True)
- We will modify the program for calculation of clothing size so that the user should enter number as long as he wants to (until they enter a negative value)

```
chestSize = eval(input("Enter chest size in centimeters: "))
while True:
    if(chestSize<=0):
        break
    elif (chestSize <= 86):
        print('You should wear XS size')
    elif (chestSize <= 93):
        print('You should wear S size')
    elif (chestSize <= 103):
        print('You should wear M size')
    elif (chestSize <= 115):
        print('You should wear L size')
    elif (chestSize <= 125):
        print('You should wear XL size')
    else:
        print('You should wear larger than XL size')
    chestSize = eval(input("Enter chest size in centimeters: "))
print('Bye, bye')
```

break statement

- The program above executes a **break** in the first if, so if the entered number is negative, the loop immediately ends
  - This is an example when you intentionally make the stop condition to always evaluates to True

Strings

- In programming a string is a variable (or object) that represents a sequence of characters
- Strings are used by programs to work with text
- The **print** and **input** functions used so far in this course use strings
- Strings are any sequence of characters, enclosed by either double quotes (“”) or single quotes (‘’)
- A triple double quote can be used for multiple-line strings

Strings in Python

```
str1 = 'This is a string'
str2 = "This is another string"
str3 = """This is a very long string that spreads over
multiple lines"""

print(str1)
print(str2)
print(str3)
```

Strings in Python

- **Length** – to get the length of a string (how many characters it has), use the built-in function **len()**.
  - For example, **len**('Hello') will return 5
- **Concatenation and repetition**
  - The operators **+** and ***** can be used on strings.
  - The **+** operator combines two strings. This operation is called concatenation.
  - The ***** operator repeats a string a certain number of times.

Concatenation and repetition

Expression	Result
'AB'+'cd'	'ABcd'
'A'+'7'+'B'	'A7B'
'Hi'*4	'HiHiHiHi'

- For example, to print a long row of dashes, use the following:
  - `print('-'*75)`

The in operator

- The in operator is used to check if a string contains something:

```
string = input('Please, enter a sequence of characters: ')
if 'a' in string:
    print('Your string contains the letter "a".')
else:
    print('Your string does not contain the letter "a".')
```

The in operator

- Any Boolean expression is allowed

```
string = input('Please, enter a sequence of characters: ')
if ('a' in string) and ('b' not in  string):
    print('Your string contains the letter "a" and does not contain the letter "b".')
else:
    print('Your string does not contain the letter "a" or it contains the letter "b".')
```

- Note how you can use double quotes to “quote” something into a string defined by single quotes
  - Vice-versa is also applicable

```
string = input('Please, enter a sequence of characters: ')
if 'a' in string:
    print("Your string contains the letter 'a'.")
else:
    print("Your string does not contain the letter 'a'.")
```

String comparison

- Operators > (greater than) and < (less then) work on strings and they compare them according to their alphabetical order
  - Capital letters are considered before lowercase letters

The find method of strings()

- Strings have a method that can be used to find a character within a string and get its position (index)
  - For example, this program:

```
string = 'Hello world'
print(string.find('o'))
```

  - Returns 4
    - Remember, we start counting from zero
    - Actually, the first occurrence of the character (in reading order) is found

Indexes of characters within a string

- Square brackets may be used to get a specific character, by its position in the string
- We can also get characters by "revers counting them" with negative index

Indexes of characters within a string

Let's consider the following string: s = 'Hello there'

Statement	Result	Description
s[0]	H	first character of s
s[4]	o	fifth character of s
s[-1]	e	last character of s
s[-2]	r	second-to-last character of s

String slices

- A slice is used to pick out part of a string. It behaves very similar to the range function.
- The syntax of slices is
  - string name[start index: end index+1]
- Slices return the characters within a string without the ending index. For instance, **str**[2:5] gives the characters located at indexes 2, 3, and 4, in the string **str**, but not the character at index 5.

String slices

- Either the start or end index may be left blank.
  - A blank start index, defaults to the start of the string.
  - A blank end index defaults to the end of the string.
  - Negative indexes refer to the ending characters of the string.
- There is an optional third argument (like in the range statement), that can specify the step.
  - A step of -1, steps backwards through the string, reversing the order of the characters.

String slices

- Let's see some examples with the following string

Slice	Result	Description
s[2:5]	cde	characters at indices 2, 3, 4
s[:5]	abcde	first five characters
s[5:]	fghij	characters from index 5 to the end
s[-2:]	ij	last two characters
s[:]	abcdefghij	entire string
s[1:7:2]	bdf	characters from index 1 to 6, by twos
s[: :-1]	jihgfedcba	a negative step reverses the string

Lists

- Let's say we have a small data set of thirty different sizes in centimeters and want not only to recommend appropriate T-Shirt size, but also to make further analysis on them
- If we make a thirty variables, size1, size2, …. etc., this could be very ineffective
- The solution is to use a list, instead

Lists

- List can be declared like that:
  - L = [70, 80, 120, 30]
  - Square brackets are used to mark start and end of a list
  - Lists can be entered via the **input**() function – remember to separate values by commas
- Lists can contain all kinds of things, even other lists. For example, the following is a valid list:
  - [1, 2.718, 'abc', [5,6,7]]

Some features of lists

- **len**(L) – function that returns the numbers of items within the list L
- **in** – operator that returns a Boolean value denoting if a list contains something

```
if 2 in L:
    print('Your list contains the number 2.')
if 0 not in L:
    print('Your list has no zeroes.')
```

Some features of lists

- + - operator that adds one list to the end of another
- * – operator that repeats a list

Expression	Result
[7,8]+[3,4,5]	[7,8,3,4,5]
[7,8]*3	[7,8,7,8,7,8]
[0]*5	[0,0,0,0,0]

Some features of lists

- **sum**(L) returns the sum of the elements in the list L
- **min**(L) returns the minimum of the elements in the list L
- **max**(L) returns the maximum of the elements in the list L

List methods

- L.**append**(x) adds the element x to the end of the list L
- L.**sort**() sorts the list L
- L.**count**(x) returns the number of times x occurs in the list L
- L.**index**(x) returns the index of the first occurrence of x in the list L if x exists in L
- L.**reverse**() reverses the list L
- L.**pop**(p) removes the item at index p from the list L and returns its value
- L.**insert**(p,x) inserts x at index p of the list L

Code comments

- A comment is a message to someone reading your program.
- Comments are often used to describe what a section of code does or how it works, especially with tricky sections of code.
- It is considered a good programming practice to comment non-obvious parts of your code
- Comments have no effect on your program.
  - You can temporarily comment a part of the code that you don't want to execute

Code comments

- There are two types of comments
  - Single line comments – the commented line starts with # symbol
  - Multiple line comments – the commented block that spans over several lines should start and end with triple quotes

Code comments in Python

```
"""
print('This line and the next are inside a comment.')
print('These lines will not get executed.')
"""
#This line is also a comment and will not get executed
print('This line is not in a comment and it will be executed.')
```

References

- Following sources are used for creation of this material
  - https://python.org
  - Heinold, Brian. "A Practical Introduction to Python Programming." *Creative Commons Attribution* (2012).

CONTACTS

Coordinator:
Technical University of Sofia

Project coordinator:
assoc. prof. Angel Terziev, PhD
aterziev@tu-sofia.bg

Author:
Assoc. Professor Aleksandar Dimov
Sofia University "St. Kliment Ohridski"
aldi@fmi.uni-sofia.bg

Web-site: ICT-TEX.eu

Co-funded by the Erasmus+ Programme of the European Union

KNOWLEDGE ALLIANCE
ICT-TEX
ICT IN TEXTILE AND CLOTHING HIGHER EDUCATION AND BUSINESS