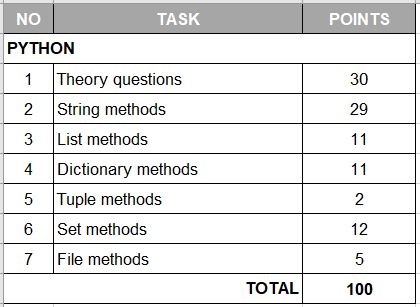
**THEORY QUESTIONS ASSIGNMENT**

Python based theory

To be completed at student’s own pace and submitted before given deadline



|  |  |  |
| --- | --- | --- |
| **1.** | **Python theory questions** | **30 points** |

1. What is Python and what are its main features?

Python is a programming language which was created by Guido van Rossum and released in 1991.

Main features:

* Object oriented programming ( OOP )
* Open source. It is available for everyone.
* Code will work in Mac, Windows or Linux which makes it platform-independent.
* Relatively simple syntax
* It does not need to compile code

1. Discuss the difference between Python 2 and Python 3

They are two different versions of the same programming language. Python 2 was released in the year 2000 and Python 3 in 2008 to fix problems that the old version had. The newest version is Python 3.7 since June 2018. Python 3 has more options of rich libraries and can be easily integrated with other languages.

Some main differences:

* Python 3’s syntax is easier to understand.
* Python 3 treats print() as a function, when Python 2 treated is as a statement.
* Python 3 is much faster than Python 2.
* Python 3 makes ordering comparisons easier than in Python 2.

1. What is PEP 8?

PEP8 is a document that provides a guidelines and the best practices on how to write Python code. It is designed to help understand the code by provided structures and makes it easier to read.

* 4 spaces per indentation level.
* It is better to use spaces than tabs.
* Limit of characters in the line is 79.
* Being consistant with the breaks next to binary operators.
* Using blank lines to indicate logical sections.
* Imports should usually be on separate lines
* Be consistent with spaces.

1. In computing / computer science what is a program?

It is a specific set of orders for computer to perform. Program contains a one-at-a-time sequences of instructions that computer can follow.

1. In computing / computer science what is a process?

A process is a program running in a computer. It can be a small task but also running a application, like a Microsoft Word. Operating systems have many background tasks running which cause that there will be many running processes even if there is just one open program.

1. In computing / computer science what is cache?

The cashe stores recently used information for quick access later. There are different types of sashe, for example: browser cashe, memory cashe, dick and processor cashe.

1. In computing / computer science what is a thread and what do we mean by multithreading?

A thread is an independed unit of execution created within the context of process. Multithreading is a widespread programming and execution model that allows multiple threads to exist within the context of one process.

1. In computing / computer science what is concurrency and parallelism and what are the differences?

Concurrency is when multiple tasks can run in overlapping period. Parallelism is when tasks actually run in parallel in multiple CPUs.

1. What is GIL in Python and how does it work?

GIL (Python Global Interpreter ) is the mutex that allows only one thread to hold the control of the Python interpreter.

1. What do these software development principles mean: DRY, KISS, BDUF

DRY – Don’t Repeat Yourself

KISS – Keep It Simple, Stupid

BDUF – Big Design Up Front

1. What is a Garbage Collector in Python and how does it work?

Garbage Collector keeps track of all objects in memory. It ensures that RAM does not fill up. It has 3 generations. Once one generation is “full”, its objects move to an older generation.

1. How is memory managed in Python?

Thanks to Garbage Collector, python deletes data that is no longer needed.

1. What is a Python module?

A Python module is a .py file.

1. What is docstring in Python?

It is a comment that is inserted to document code. We use triple quote for that.

1. What is pickling and unpickling in Python? Example usage.

Pickle is a module that converts objects I to byte streams. Unpickling converts byte streams into Python objects.

1. What are the tools that help to find bugs or perform static analysis?

Pychecker and Pylint. They can both be added using pip.

1. How are arguments passed in Python by value or by reference? Give an example.

By reference. It means that the argument given to a function refers to a variable that already exists and the changes that are made inside the function have an effect outside the function.

1. What are Dictionary and List comprehensions in Python? Provide examples.
2. What is namespace in Python?

It is a collection of currently defined names along with information about the object.

20.What is pass in Python?

It is a placeholder for future code that is used in functions.

1. What is unit test in Python?

Unit test checks if a single component of code works.

1. In Python what is slicing?

Slicing is used to return just a specific characters from the variable.

It starts withs 0 and for example:

x = “CodeFirstGirls”

print(x[4:13])

result: FirstGirls

1. What is a negative index in Python?

There are values at the end of the object. We start indexing with (0,1,2…) but we also can index -1 which will be the last value and -2 as a second to last etc.

24.How can the ternary operators be used in python? Give an example.

Ternary operations are conditional expressions such as if, elif, else.

num = float(input("Enter a number: "))

if num >= 0:

if num == 0:

print("Zero")

else:

print("Positive number")

else:

print("Negative number")

1. What does this mean: \*args, \*\*kwargs? And why would we use it?
2. How are range and xrange different from one another?
3. What is Flask and what can we use it for?

Flask is a web framework. It is a Python module that lets you develop web applications.

1. What are clustered and non-clustered index in a relational database?

Clustered index stored the rows physically on the disk in the same order as the index. There can be only one clustered index.

A non-clustered index points to the physical rows where the information can be found. There can be many non-clustered index in the table.

1. What is a ‘deadlock’ a relational database?

A deadlock is a situation in which two or more transactions are waiting for one another to give up locks.

1. What is a ‘livelock’ a relational database?

A Live lock is one, where a request for an exclusive lock is denied continuously because a series of overlapping shared locks keeps on interfering each other.

|  |  |
| --- | --- |
| **2. Python string methods: describe each method and provide an example** | **29 points** |

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | It change the first character of the word to upper case. | example = ("how are you today?") x = example.capitalize() print(x)  Result: How are you today? |
| **casefold()** | It converts all characters in the string into lower case | example = ("How Are You Today?") x = example.casefold() print(x)  Result: how are you today? |
| **center()** | This will align the string to the center using specified characters. | example =( "How are you today?") x = example.center(50) print(x)  Result: How are you today? |
| **count()** | It returns how many times a specified value appears in a string | example = ("I love dogs, dogs are my favourite animals") x = example.count("dogs") print(x)  Result: 2 |
| **endswith()** | Methods return “True” if the string ends with the value that we wrote in (). If the string does not ends with the same value, it return False. | example =( "How are you") x = example.endswith("u") print(x)  Result: True  example = ("How are you") x = example.endswith("o") print(x)  Result: False |
| **find()** | The methods return the position of a specified string. | example = ("How are you?") x = example.find("you") print(x)  Return: 8 |
| **format()** | This method formats the specified values and insert them into the placeholders which are defined using {}. | example = ("Hello") next\_example ="How are you?" x = "{} , {}".format(example, next\_example) print(x)  Return: Hello , How are you? |
| **index()** | It is very similar to find() but it shows the error when the program cannot find a string. It cannot be used with conditional statements. | example = ("How are you?") x = example.index("blue")  print(x)  Return: error |
| **isalnum()** | It return True if the strings are alphanumeric ( a -z and 0-9). It will be always True if there is only one string. False will be always when there will be a space or any other sign that is not a letter or a number. | example =( "Howareyou ") x = example.isalnum() print(x)  Return: True  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  example = ("How are you") x = example.isalnum() print(x)  Return : false |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | It is similar to isalnum() but it return false if there are also number. That will return True only if the string contains character ( a – z ) | example =( "Howareyou0 ") x = example.isalpha () print(x)  Return: False |
| **isdigit()** | This method returns True if the string contains only digits. If there are any letter, it return False. | example = ("1234") x = example.isdigit() print(x)  Return: True |
| Islower() | These methods check if all characters in the string are lower case. If they are all lower case, it returns True, if there are any upper case letters it returns False. | example = ("how are you") x = example.islower() print(x)  Return: True  example = ("How are you") x = example.islower() print(x)  Return: False |
| **isnumeric()** | This method returns True if all characters in the string are numeric. It does not include floating number or negative numbers. | example =( "2022") x = example.isnumeric() print(x)  Return: True |
| **isspace()** | This method return True is there are only white spaces in the string. Otherwise, it is False. | example =( " ") x = example.isspace() print(x)  Return: True |
| **istitle()** | This method returns True if every first letter in the string is uppercase. | example =( "How Are You") x = example.istitle() print(x)  Return: True |
| **isupper()** | This method returns True if all letter in the string are uppercase. | example = ("HOW ARE YOU") x = example.isupper() print(x)  Return: True |
| **join()** | This method put together strings with a separator of our choice into one string. | example = ("a", "b") together = " and ".join(example) print(together)  Return: a and b |
| **lower()** | These methods change every letter in the string into lowercase | example =( "How Are You?") x = example.lower() print(x)  Return: how are you? |
| **lstrip()** | This method removes all the spaces to the left of the string. | example = (" there is a lot of free space") x = example.lstrip() print(x)  Return: there is a lot of free space |
| **replace()** | This method replaces a specified phrase with another specified phrase. | example =( “ I like oranges and apples” )  x = example.replace (“apples” , “pears” )  print(x)  Return: I like oranges and pears |
| **rsplit()** | This method converts a string into a list. | example=( “ How are you ? “ )  x = example.rsplit()  Print(x)  Result: [“How”, “are”, “you, “?”] |
| **rstrip()** | This method removes any white spaces at the end of the string | example=( “ How are you? “)  x = example.rstrip()  print(x)  Return: How are you? |
| **split()** | This method split a string into a list where each word is a list item. The only difference between the split() and rsplit() is when the maxsplit parameter is specified: with rsplit() starts splitting a string from the right side and split() starts splitting from the left side. | example = (“How are you ? “)  x = example.split()  print(x)  Return: [ “How” , “are”, “you”, “?”] |
| **splitlines()** | This method split a string into a list with two string after eliminating \n character. | example = (“First line \n Second line”)  x = example/slitlines()  print(x)  Result : [“First line “ , “Second line”] |
| **startswith()** | This method return True if the string starts with a specific prefix. | example = (“ How are you? “ )  x = example.startswith(“How are”)  print(x)  Return: True |
| **strip()** | This method removes spaces at the beginning and at the end of the string. | example = (“ how are you? “)  x= example.string()  print(x)  Return: how are you? |
| **swapcase()** | This method make the lower case letters upper case and upper case letters lower case. | example = “HELLO, How are you?”  x = example.swapcase()  print(x)  Return: hello, hOW ARE YOU? |
| **title()** | This method makes the first letter in each word upper case. | example = (“how are you?”)  x = example.title()  print(x)  Return: How Are You? |
| **upper()** | This method makes the string upper case | example = (“ How are you?)  x = example.upper()  print(x)  Return: HOW ARE YOU? |

|  |  |  |
| --- | --- | --- |
| **3.** | **Python list methods:**  **describe each method and provide an example** | **11 points** |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**append()**](https://www.w3schools.com/python/ref_list_append.asp) | This method add an element at the end of the list. | example = [“orange”, “apple”, “pear”]  example.append(“banana”)  print(example)  Return: [“orange”, “apple”, “pear”, “banana”] |
| [**clear()**](https://www.w3schools.com/python/ref_list_clear.asp) | This method gets rid of all elements in the list | example = [“orange”, “apple”, “pear”]  example.clear()  print(example)  Return: [ ] |
| [**copy()**](https://www.w3schools.com/python/ref_list_copy.asp) | This method copies a list. | example = [“orange”, “apple”, “pear”]  example.copy()  print(example)  Result: [“orange”, “apple”, “pear”] |
| [**count()**](https://www.w3schools.com/python/ref_list_count.asp) | This method counts the number of times an item in the list appears. | example = [“orange”, “apple”, “pear”, “apple”]  x=example.count(“apple”  print(x)  Return: 2 |
| [**extend()**](https://www.w3schools.com/python/ref_list_extend.asp) | This method add elements of the list to the another already existing list. | example = [“orange”, “apple”, “pear”]  new\_example = [“banana”, “cherry”, “blueberries”]  example.extend(new\_example)  print(example)  Return: ['orange', 'apple', 'pear', 'banana', 'cherry', 'blueberries'] |
| [**index()**](https://www.w3schools.com/python/ref_list_index.asp) | This method returns the index of the specified element in the list. | example = [“orange”, “apple”, “pear”]  x = example.index(“apple”)  print(x)  Return: 1 |
| [**insert()**](https://www.w3schools.com/python/ref_list_insert.asp) | This method inserts an element to the list at the specified index. | example = [“orange”, “apple”, “pear”]  example.insert(2, “banana”)  print(example)  Return: [“orange”, “apple”, “banana”, “pear”] |
| [**pop()**](https://www.w3schools.com/python/ref_list_pop.asp) | This method method removes the item at the given index from the list and returns the removed item | example = [“orange”, “apple”, “pear”]  x = example.pop(2)  print(x)  Return: pear |
| [**remove()**](https://www.w3schools.com/python/ref_list_remove.asp) | This method method removes the first matching element from the list. | example = [“orange”, “apple”, “pear”]  example.remove(“pear”)  print(example)  Return: [“orange”, “apple”] |
| [**reverse()**](https://www.w3schools.com/python/ref_list_reverse.asp) | This method reverses the elements of the list. | example = [“orange”, “apple”, “pear”]  example.reverse()  print(example)  Return: [“pear”, “apple”, “orange”] |
| [**sort()**](https://www.w3schools.com/python/ref_list_sort.asp) | This method sort the list alphabetically. | example = [“orange”, “apple”, “pear”]  example.sort()  print(example)  Return: [“apple”, “orange”, “pear”] |

|  |  |  |
| --- | --- | --- |
| **4.** | **Python tuple methods:**  **describe each method and provide an example** | **2 points** |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**count()**](https://www.w3schools.com/python/ref_tuple_count.asp) | This method counts the number of times the item appears in the tuple. | example = ( 3, 1,3,5,7,9,2,3,6,3,8)  x = example.indes(3)  print(x)  Return: 4 |
| [**index()**](https://www.w3schools.com/python/ref_tuple_index.asp) | This method returns the position of the element in the tuple. | example = ( 1,3,5,7,9,2,4,6)  x = example.index(7)  print(x)  Return: 3 |

|  |  |  |
| --- | --- | --- |
| **5.** | **Python dictionary methods:**  **describe each method and provide an example** | **11 points** |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**clear()**](https://www.w3schools.com/python/ref_dictionary_clear.asp) | This method gets rid of all elements in a dictionary. | example = {  “cinnamon roll : 2”,  “chocolate twist: 1.5”,  “croissant: 2 “  }  example.clear()  print(example)  Return: {} |
| [**copy()**](https://www.w3schools.com/python/ref_dictionary_copy.asp) | This method copies the dictionary | example = {  “cinnamon roll : 2”,  “chocolate twist: 1.5”,  “croissant: 2 “  }  example.copy()  print(example)  Return: {‘ cinnamon roll’ : 2, ‘ chocolate twist’ : 1.5, ‘ croissant’ : 2 } |
| [**fromkeys()**](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | It returns a new dictionary with the keys that we specified in the variable. Also, we can specify a value that will be passing to all the keys. | example = (“cinnamon roll”, “chocolate twist”, “croissant”)  value = 2  x = direct.fromkeys(example, value)  print(x)  Result: { “cinnamon roll” : 2, “chocolate twist” : 2, “croissant”: 2} |
| [**get()**](https://www.w3schools.com/python/ref_dictionary_get.asp) | It returns the value of the specific key. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  x = example.get(“chocolate twist”)  print(x)  Result: 1.5 |
| [**items()**](https://www.w3schools.com/python/ref_dictionary_items.asp) | It returns all objects with their values from the dictionary. It returns it into a list. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  x = example.items()  print(example)  Result: dict\_items([‘cinnamon roll’, 2), (‘chocolate twist’, 1.5), (‘croissant’, 2)]) |
| [**keys()**](https://www.w3schools.com/python/ref_dictionary_keys.asp) | It returns all keys from the dictionary and put them into a list. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  x = example.keys()  print(x)  Result: dict\_items([‘cinnamon roll’ , ‘ chocolate twist’ , ‘croissant’]) |
| [**pop()**](https://www.w3schools.com/python/ref_dictionary_pop.asp) | It removes a specific key and its value from the dictionary. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  example.pop(“chocolate twist”)  print(example)  Result: { “cinnamon roll”:2, “croissant”:2} |
| [**popitem()**](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | It removes the last key and its value from the dictionary. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  example.popitem()  print(example)  Result: {“cinnamon roll” : 2, “chocolate twist” : 1.5} |
| [**setdefault()**](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | It returns the value of the item with the specified key. But if the item does not exist in the dictionary, insert the key and its value and it still will work. | example = {  “cinnamon roll : 2”,  “chocolate twist: 1.5”,  “croissant: 2 “  }  x = example.setdefault(“bacon bun “, “3” )  print(x)  Result : 3 |
| [**update()**](https://www.w3schools.com/python/ref_dictionary_update.asp) | It updates the dictionary with the specified key and its value and adds it at the end. | example = {  “cinnamon roll : 2”,  “chocolate twist: 1.5”,  “croissant: 2 “  }  example.update({“bacon bun” : “3”})  print(example)  Result: Result: { “cinnamon roll” : 2, “chocolate twist” : 1.5, “croissant”: 2, “bacon bun” : 3} |
| [**values()**](https://www.w3schools.com/python/ref_dictionary_values.asp) | It returns the list of all values in the dictionary. | example = {  “cinnamon roll : 2”  “chocolate twist: 1.5”  “croissant: 2 “  }  x = example.values()  print(x)  Result: dict\_values([ 2 , 1.5 , 2)] |

|  |  |  |
| --- | --- | --- |
| **6.** | **Python set methods:**  **describe each method and provide an example** | **12 points** |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**add()**](https://www.w3schools.com/python/ref_set_add.asp) | This methods adds a new item to the set. | example = {“orange”, “apple”, “pear”}  example.add(“cherry”)  print(example)  Return: {'cherry', 'apple', 'orange', 'pear'} |
| [**clear()**](https://www.w3schools.com/python/ref_set_clear.asp) | This methods removes all elements from the set. | example = {“orange”, “apple”, “pear”}  example.clear()  print(example)  Result : set() |
| [**copy()**](https://www.w3schools.com/python/ref_set_copy.asp) | It copies the set. | example = {“orange”, “apple”, “pear”}  example.copy()  print(example)  Return: {“pear” , “apple”, “orange”} |
| [**difference()**](https://www.w3schools.com/python/ref_set_difference.asp) | Return a set that contains items that only exist in the first set and does not exist in the second one. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  x = example\_one.defference(example\_two)  print(example\_one)  Return: { “ apple” , “pear”} |
| [**intersection()**](https://www.w3schools.com/python/ref_set_intersection.asp) | Return the items that exist in both sets. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  x = example\_one.intersection(example\_two)  print(example\_one)  Return: {“apple”} |
| [**issubset()**](https://www.w3schools.com/python/ref_set_issubset.asp) | Return True if all item from the first ( main ) set are present in the second set. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “apple”, “pear” “strawberry” , “ blueberry” }  x = example\_one.issubset(example\_two)  print(example\_one)  Return: True |
| [**issuperset()**](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns true if all items from the second set are present in the first ( main ) set. | example\_one = {“orange”, “apple”, “pear”, “strawberry” , “ blueberry”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  x = example\_one.issuperset(example\_two)  print(example\_one)  True |
| [**pop()**](https://www.w3schools.com/python/ref_set_pop.asp) | It removes a random element from the set. | example = {“orange”, “apple”, “pear”}  example.pop()  print(example)  Return: { “orange” , “pear” } |
| [**remove()**](https://www.w3schools.com/python/ref_set_remove.asp) | It removes a specific item from the set. | example = {“orange”, “apple”, “pear”}  example.remove(“orange”)  print(example)  Return: { “apple” , “pear” } |
| [**symmetric\_differ ence()**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Return all items from both sets exept ones that are present in both sets. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  x = example\_one.symmetric\_difference(example\_two)  print(example\_one)  Return: { “ apple” , “pear” , “strawberry” , “ blueberry”} |
| [**union()**](https://www.w3schools.com/python/ref_set_union.asp) | Return all items that are in two sets, but does not duplicate these that are in both of them. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  x = example\_one.union(example\_two)  print(example\_one)  Return: {“orange”, “apple”, “pear”, “strawberry”, “ blueberry”} |
| [**update()**](https://www.w3schools.com/python/ref_set_update.asp) | Insert all elements from the second set into the first ( main ) set. | example\_one = {“orange”, “apple”, “pear”}  example\_two = {“orange” , “strawberry” , “ blueberry” }  example\_one.update(example\_two)  print(example\_one)  Return: = {“orange”, “apple”, “pear”, “strawberry”, “ blueberry”} |

|  |  |  |
| --- | --- | --- |
| **7.** | **Python file methods:**  **describe each method and provide an example** | **5 points** |

**Example file:**

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**read()**](https://www.w3schools.com/python/ref_file_read.asp) | This method reads a specific file. | f = open(“zen\_of\_python.txt” , “r” )  print(f.read)  Result:  Beautiful is better than ugly.  Explicit is better than implicit.  Simple is better than complex |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | This method read only one line from the file. | f = open(“zen\_of\_python.txt” , “r” )  print(f.readline())  Result:  Beautiful is better than ugly |
| [**readlines()**](https://www.w3schools.com/python/ref_file_readlines.asp) | This method returns lines from the file as if they were a list. | f = open(“zen\_of\_python.txt” , “r” )  print(f.readlines())  Result:  [‘Beautiful is better than ugly. \n ‘  ‘Explicit is better than implicit.\n’  ‘Simple is better than complex.\n’] |
| [**write()**](https://www.w3schools.com/python/ref_file_write.asp) | This method writes text in the file. We can choose “a” which will add a text at the end of the existing file or “w” which will delete previous text and replace it with the new text. | f = open(“zen\_of\_python.txt” , “w” )  f.write(“Flat is better than nested.”)  Result: Flat is better than nested. |
| [**writelines()**](https://www.w3schools.com/python/ref_file_writelines.asp) | It writes the items of a list to the file.  “a” and “w” works exactly the same as in write() | f = open(“zen\_of\_python.txt” , “a” )  f.writelines(“\n Flat is better than nested”)  Result:  Beautiful is better than ugly.  Explicit is better than implicit.  Simple is better than complex  Flat is better than nested |