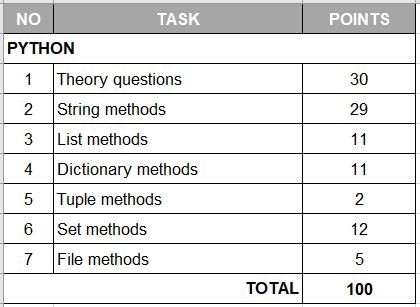
THEORY QUESTIONS ASSIGNMENT

Python based theory

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



**30 points**

**1. Python theory questions**

1. What is Python and what are its main features?
   * Python is a dynamic, high-level, free open source, and interpreted programming language. The main features are: Free and Open Source, easy to code and to read, Object-Oriented Language and high-level language.
2. Discuss the difference between Python 2 and Python 3

|  |  |
| --- | --- |
| Python2 | Python3 |
| Legacy: still used in older company  Libraries are not compatible with new version  use ASCII  Division rounds to the nearest whole number  print"works like this" | Future: will take over Python2 by 2022  Libraries are created striclty to work only with Python3  Use UNICODE  Division return exact float number  print("works like this") |

1. What is PEP 8?
   * PEP 8 provides guidelines and best practices on how to write Python code.
2. In computing / computer science what is a program?
   * A program is a set of ordered operation for a computer to perform a function.
3. In computing / computer science what is a process?

A process happens when a program is being executed.

1. In computing / computer science what is cache?
   * Cache is a hardware or software that store data temporarily in a computing environment.
2. In computing / computer science what is a thread and what do we mean by multithreading?
   * Multithreading refers to the ability of executing multiple threads by rapidly switching the control of the CPU between threads.
3. In computing / computer science what is concurrency and parallelism and what are the differences?
   * - Concurrency is the task of running and managing the multiple computations at the same time  
     - Parallelism tasks are divided into smaller sub-tasks that are processed simultaneously or parallel.
4. What is GIL in Python and how does it work?
   * In Python, the GIL is a mutex that allows only one thread at a time to have the control of the Python interpreter. In other words, the lock ensures that only one thread is running at any given time.
5. What do these software development principles mean: DRY, KISS, BDUF
   * - DRY -> Don’t Repeat Yourself - Never write the same logic more than once  
     - KISS –> Keep It Simple Stupid - Try to write simple code and instead of over-engineer a soluiton to a problem  
     - BDUF -> Big Design Up Front - Design the application before writing code
6. What is a Garbage Collector in Python and how does it work?
   * The garbage collector keeps track of all objects in memory. Python deletes unwanted objects and automatically to free the memory space when an object's reference count reaches zero.
7. How is memory managed in Python?
   * The Python memory is primarily managed by Python private heap space. All Python objects and data structures are managed by this private heap and it is ensured internally by the Python memory manager.
8. What is a Python module?
   * A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended.
9. What is docstring in Python?
   * Python docstrings appear right after the definition of a function, method, class, or module to provide a convenient documentation associated so programmers can understand what it does.
10. What is pickling and unpickling in Python? Example usage.
    * - Pickling: It is a process where a Python object hierarchy is converted into a byte stream.   
      - Unpickling: It is the inverse process
11. What are the tools that help to find bugs or perform static analysis?
    * Pychecker and Pylint are the static analysis tools that help to find bugs in python.
12. How are arguments passed in Python by value or by reference? Give an example.

. All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function.  
 > dogs={'Jet':28,'Rum':25,'Cookie':32,'Laszlo':25}   
 >   
 > def test(dogs):  
 >   
 > new={'Bobby':30,'Moon':28}   
 >   
 > student.update(new)  
 >   
 > print("Inside the function", dogs)  
 >   
 > return   
 >   
 > test(dogs) > Inside the function {'Jet':28,'Rum':25,'Cookie':32,'Laszlo':25, 'Bobby':30,'Moon':28}   
 >   
 > print("outside the function:",dogs)

> outside the function: {'Jet':28,'Rum':25,'Cookie':32,'Laszlo':25, 'Bobby':30,'Moon':28}

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

List comprehensions provide a concise way to create lists.

- Syntax: [expression for item in iterable if conditional]

- eg: list\_example=[i for i in range(1,11)]  
 A Dictionary comprehension works as the list but needs two expressions separated with a colon followed by “for” and “if” clauses.

- Syntax: {key:value for (key,value) in iterable if conditional}

- eg: dic\_example={num:i for i in range(1,11) if i%2==0}

1. What is namespace in Python?
   * 20. A namespace is a system that has a unique name for each and every object in Python. We can have built-in namespace, global namespace, local namespace.
2. What is pass in Python?
   * The pass statement is like a placeholder for future code to avoid getting an error when empty code is not allowed.
3. What is unit test in Python?
   * Unit testing is a method for testing software that checks individual units of source code.
4. In Python what is slicing?
   * The slice() function returns the extraction of a part of an object in any sequence can be applied on string, tuple, or list
5. What is a negative index in Python?
   * A Negative index starts from the end of the iterable. Hence, the last element can be accessed at index -1.
6. How can the ternary operators be used in python? Give an example.
   * - Ternary operators are conditional expressions are operators that evaluate something based on a condition being true or false.  
     - EG: name = input('write your name >>')  
     - EG: say\_hello = f"Hello, {name}" if name else "Hello stranger!"
7. What does this mean: \*args, \*\*kwargs? And why would we use it?
   * \*args refer to the number of arguments and \*\*kwargs to the length of arguments -> we use it when we are unsure of the formatting of the argument passed to the function.
8. How are range and xrange different from one another?
   * range : Returns a list of integers. Takes more memory as it keeps the entire list of elements in memory.  
     xrange: Returns a generator object. Takes less memory as it keeps only one element at a time in memory.
9. What is Flask and what can we use it for?
   * Flask is a Python web framework that provides tools and features that make creating web applications in Python easier.
10. What are clustered and non-clustered index in a relational database?
    * Clustered indexes sort and store the data rows in the table or view based on their key values. A non-clustered index collects the data at one place and records at another place.
11. What is a ‘deadlock’ a relational database?
    * A deadlock happens when two concurrent transactions cannot make progress because each one waits for the other to release a lock,
12. What is a ‘livelock’ a relational database?
    * In the case of a livelock, the states of the processes involved in a live lock scenario constantly change

**29 points**

**2. Python string methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | The capitalize() method converts the first character of a string to an uppercase letter and all other alphabets to lowercase. | name = ‘feDRrica’  name.capitalize()  Print(name)  >> ‘Federica’ |
| **casefold()** | The casefold() method converts all characters of the string into lowercase letters and returns a new string. Not change the first one. | text = "THIS Is a TeST"  lowercased\_string = text.casefold()  print(lowercased\_string) >> this is a test  print(text) >> THIS Is a TeST |
| **center()** | The center() method will align the string in the center, using a specified character passed to the function | text = "center me"  centered\_text = text.center(20)  print(centered\_text)  >> center me |
| **count()** | The count() method returns the number of how times a specified value appears in a string. | message = 'count how many c in this sentence'  print('Number of occurrence of c:', message.count('c'))  >> Number of occurrence of c: 3 |
| **endswith()** | The endswith() method returns a boolena if a string ends with the specified value. | message = 'this text and with a m'  print(message.endswith('m'))  >> True |
| **find()** | The find() method returns the index of first occurrence of the element passed to the function. | message = 'This is an example'  print(message.find('e'))  >> 11 |
| **format()** | We can use a placeholder with {} in a string and cal the method format() to insert a specified alue inside the string's placeholder. | text = "Hello {}!"  print(text.format('Fede'))  >> Hello Fede! |
| **index()** | The index() method returns the index of the first match of a substring inside the string. | text = 'Find me'  result = text.index('me')  print(result)  >> 5 |
| **isalnum()** | The isalnum() method returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False. | test1 = "Thisiscorrect1234"  print(test1.isalnum()) #True  test2 = "this is not !"  print(test2.isalnum()) #False |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | The isalpha() method checks if all the characters are alphabet letters (a-z) and if so return True. | txt = "Alllettersinsentence"  x = txt.isalpha()  print(x) #True |
| **isdigit()** | The isdigit() method checks if all the characters in a string are digits and if so return True. | txt = "123456"  x = txt.isdigit()  print(x) #True |
| **islower()** | The islower () method checks if all the characters in a string are lowercase and if so returns True. | txt = "all lowercase!"  x = txt.islower()  print(x) #True |
| **isnumeric()** | The isnumeric () method checks if all the characters in a string are numeric (0-9) and if so returns True. | txt = "345676543"  x = txt.isnumeric()  print(x) #True |
| **isspace()** | The isspace() method checks if all the characters in a string are whitespaces and if so returns True. | txt = " "  x = txt.isspace()  print(x) #True |
| **istitle()** | The istitle() method checks if all the words in a string are Capitalized and if so returns True. | txt = "This Sentence Is Capitalized!"  x = txt.istitle()  print(x) #True |
| **isupper()** | The isupper() method checks if all the chars in a string are UPPERCASE and if so returns True. | txt = "THIS IS UPPERCASE!"  x = txt.isupper()  print(x) #True |
| **join()** | The string join() method returns a string by joining all the elements of an iterable (list, string, tuple), separated by the given separator declared before the method. | text = ['Join', 'this', 'list', 'in', 'one', 'string']  print(' '.join(text))  > Join this list in one string |
| **lower()** | The lower() method returns the lowercase string from the given string. | message = 'LOWER ME'  print(message.lower())  > lower me |
| **lstrip()** | The Istrip() method returns a new and cleaned string by removing any leading whitespaces including tabs (\t). | string =' abc '  Clean\_string = string.Istrip()  >> 'abc ' |
| **replace()** | The replace() method replaces a substring in a string with another specified element. | txt = "I will be replaced"  x = txt.replace("replaced", "happy")  print(x)  >> I will be happy |
| **rsplit()** | The rsplit() method splits a string into a list, starting from the right. | txt = "we, become, a, list"  list = txt.rsplit(", ")  print(x)  >> ['we', 'become', 'a', 'list'] |
| **rstrip()** | The strip() method returns a new and cleaned string by removing any trailing whitespaces including tabs (\t). | String =' abc '  Clean\_string = string.rstrip()  >> ' abc' |
| **split()** | The split() method splits a string into a list. | txt = "we, become, a, list"  list = txt.rsplit()  print(x)  >> ['we', 'become', 'a', 'list'] |
| **splitlines()** | The splitlines() method splits a string into a list at line breaks. | txt = "First line\nSecond line"  x = txt.splitlines()  print(x)  >> ['First line', 'Second line'] |
| **startswith()** | The startswith() method checks if the string starts with the specified value, if so return True. | txt = "This sentence start with This."  x = txt.startswith("This")  print(x)  >> True |
| **strip()** | The strip() method returns a new and cleaned string by removing any leading and trailing whitespaces including tabs (\t). | string =' abc '  Clean\_string = string.strip()  >> 'abc' |
| **swapcase()** | The swapcase() method returns a string where all the upper case letters are swapped to lower case and lower case to upper case. | txt = "this will be upper and THIS LOWERED"  x = txt.swapcase()  print(x)  >> THIS WILL BE UPPER AND this lowered |

|  |  |  |
| --- | --- | --- |
| **title()** | The title() method returns a string where every word is Capitalized. | txt = "CaPITalize each worD"  x = txt.title()  print(x)  >> Capitalize Each Word |
| **upper()** | The upper() method returns a string with all characters upper cased. | txt = "uppercase each character"  x = txt.upper()  print(x)  >> UPPERCASE EACH CHARACTER |

**11 points**

**3. Python list methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**append()**](https://www.w3schools.com/python/ref_list_append.asp) | The append() method adds an item to the end of the list. | my\_dogs = ['Luna', 'Cody', 'Pound']  my\_dogs.append('Star')  print(my\_dogs)  >> ['Luna', 'Cody', 'Pound', 'Star'] |
| [**clear()**](https://www.w3schools.com/python/ref_list_clear.asp) | The clear() method removes all the elements from a list. | my\_dogs = ["Luna", "Cody", "Star"]  my\_dogs.clear()  print(my\_dogs)  >> [] |
| [**copy()**](https://www.w3schools.com/python/ref_list_copy.asp) | The copy() method returns a shallow copy of the list. | odd\_numbers = [3, 5, 7, 9]  numbers = odd\_numbers.copy()  print('Copied List:', numbers)  >> Copied List: [3, 5, 7, 9] |
| [**count()**](https://www.w3schools.com/python/ref_list_count.asp) | The count() method returns the number of times the specified element appears in the list. | numbers = [2, 3, 5, 2, 11, 2, 7]  count = numbers.count(2)  print('Count of 2:', count)  >> Count of 2: 3 |
| [**extend()**](https://www.w3schools.com/python/ref_list_extend.asp) | The extend() method extend the list by adding all the elements of the second list passed to the function. | first\_list = [1, 2, 3]  second\_list = [4, 5]  first\_list.extend(second\_list)  print('List after extend():', first\_list)  >> List after extend(): [1, 2, 3, 4, 5] |
| [**index()**](https://www.w3schools.com/python/ref_list_index.asp) | The index() method finds the index of the specified element in the list. | nums = ['0', '1', '2', '3']  index = nums.index('2')  print(index)  >> 2 |
| [**insert()**](https://www.w3schools.com/python/ref_list_insert.asp) | The insert() method inserts an element inside a list at the specified index. | letters = ['a', 'b', 'd', 'e']  letters.insert(2, 'c')  print('List:', letters)  >> List: ['a', 'b', 'c', 'd', 'e'] |
| [**pop()**](https://www.w3schools.com/python/ref_list_pop.asp) | The pop() method removes the item at the given index from the list and returns the removed item that we can then store inside a variable. | numbers = [1, 2, 3, 4]  removed\_number = numbers.pop(2)  print('Removed Element:', removed\_number)  print('Updated List:', numbers)  >> Removed Element: 3  >> Updated List: [1, 2, 4] |
| [**remove()**](https://www.w3schools.com/python/ref_list_remove.asp) | The remove() method removes the first matching element that is passed to the function from the list. | numbers = [1, 2, 3, 4, 2, 4, 7, 9]  numbers.remove(2)  print('Updated List: ', numbers)  >> Updated List: [1, 3, 4, 2, 4, 7, 9] |
| [**reverse()**](https://www.w3schools.com/python/ref_list_reverse.asp) | The reverse() method return a list with the elements reversed. | numbers = [1, 2, 3, 4, 5, 6]  numbers.reverse()  print('Reversed List:', numbers)  >> Reversed List: [6, 5, 4, 3, 2, 1] |
| [**sort()**](https://www.w3schools.com/python/ref_list_sort.asp) | The sort() method return a listh with the element sorted ascending by default. | animals = ['Dog', 'Cat', 'Monkey', 'Parrot']  animals.sort()  print(animals)  >> ['Cat', 'Dog', 'Monkey', 'Parrot'] |

**2 points**

**4. Python tuple methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**count()**](https://www.w3schools.com/python/ref_tuple_count.asp) | The count() method returns the number of times a specified value appears in the tuple. | nums = (1, 3, 7, 8, 7, 3, 4, 6, 8, 3)  x = nums.count(3)  print(x)  >> 3 |
| [**index()**](https://www.w3schools.com/python/ref_tuple_index.asp) | The index() method returns the index of the specified element in the tuple. | nums = (1, 3, 7, 8, 7, 3, 4, 6, 8, 3)  x = nums.index(8)  print(x)  >> 3 |

**11 points**

**5. Python dictionary methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**clear()**](https://www.w3schools.com/python/ref_dictionary_clear.asp) | The clear() method removes all the elements from a dictionary. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  my\_dog.clear()  print(my\_dog)  >> {} |
| [**copy()**](https://www.w3schools.com/python/ref_dictionary_copy.asp) | The copy() method returns a shallow copy of the dictionary | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  My\_dog\_copied = my\_dog.copy()  print(my\_dog\_copied)  >> {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  } |
| [**fromkeys()**](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | The fromkeys() method returns a dictionary with keys and values passed to the function | key = ('name', 'age', 'breed')  value = 0  my\_dog = dict.fromkeys(key, value)  print(my\_dog)  >> {'name': 0, 'age': 0, 'breed': 0} |
| [**get()**](https://www.w3schools.com/python/ref_dictionary_get.asp) | The get() method returns the value for the specified key. | my\_dog = {'name': 'Thor', 'age': 4, 'breed': 'Akita'}  print(my\_dog.get('name'))  >> Thor |
| [**items()**](https://www.w3schools.com/python/ref_dictionary_items.asp) | The items() method returns a view object that displays a list of dictionary's (key, value) tuple pairs. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  print(my\_dog.items())  >> dict\_items([('name', 'Thor'), ('age', 4), ('Breed', 'Akita')]) |
| [**keys()**](https://www.w3schools.com/python/ref_dictionary_keys.asp) | The keys() method returns a view object with the keys of the dictionary. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  keys = my\_dog.keys()  print(keys)  >> dict\_keys(['name', 'age', 'Breed']) |
| [**pop()**](https://www.w3schools.com/python/ref_dictionary_pop.asp) | The pop() method removes and returns an element from a dictionary having the key passed to the method. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  name = my\_dog.pop('name')    print('Popped:', name)  >> Popped: Thor |
| [**popitem()**](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | The popitem() method removes the inserted item into the dictionary. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  my\_dog.popitem()  print(my\_dog)  >> {'name': 'Thor', 'age': 4} |

|  |  |  |
| --- | --- | --- |
| [**setdefault()**](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | The setdefault() method returns the value of the item with the specified key. If the key does not exist, create a new key with the value passed to the method. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  x = my\_dog.setdefault("color", "Beige")  print(x)  print(my\_dog)  >> Beige  >> {'name': 'Thor', 'age': 4, 'Breed': 'Akita', 'color': 'Beige'} |
| [**update()**](https://www.w3schools.com/python/ref_dictionary_update.asp) | The update() method expands the dictionary with the elements from another dictionary | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  my\_dog\_color = {'Color': 'beige'}  my\_dog.update(my\_dog\_color)  print(my\_dog)  >> {'name': 'Thor', 'age': 4, 'Breed': 'Akita', 'Color': 'beige'} |
| [**values()**](https://www.w3schools.com/python/ref_dictionary_values.asp) | The values() method displays a list of all the values in the dictionary. | my\_dog = {  "name": "Thor",  "age": 4,  "Breed": 'Akita'  }  print(my\_dog.values())  >> dict\_values(['Thor', 4, 'Akita']) |

**12 points**

**6. Python set methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**add()**](https://www.w3schools.com/python/ref_set_add.asp) | The add() method adds a given element to a set if not present already. | my\_dogs = {'Luna', 'Doggy', 'Marta'}  my\_dogs.add('Thor')  print(my\_dogs)  >> {'Thor', 'Marta', 'Doggy', 'Luna'} |
| [**clear()**](https://www.w3schools.com/python/ref_set_clear.asp) | The clear() method removes all the elements from a set. | my\_dogs = {'Luna', 'Doggy', 'Marta'}  my\_dogs.clear()  print(my\_dogs)  >> {} |
| [**copy()**](https://www.w3schools.com/python/ref_set_copy.asp) | The copy() method returns a shallow copy of the set. | my\_dogs = {'Luna', 'Doggy', 'Marta'}  my\_dogs\_copy = my\_dogs.clear()  print(my\_dogs)  >> my\_dogs = {'Luna', 'Doggy', 'Marta'} |
| [**difference()**](https://www.w3schools.com/python/ref_set_difference.asp) | The function difference() returns a set that is the difference between the set on which the method is called compared to the one passed as an argument. | set\_a = {1, 2, 3, 4, 8}  set\_b = {10, 3, 8, 4, 6}  set\_a.difference(set\_b)  >> {1, 2} |
| [**intersection()**](https://www.w3schools.com/python/ref_set_intersection.asp) | The intersection() method returns a new set with elements that are present in both set compared. | set\_a = {1, 2, 3, 4, 8}  set\_b = {10, 3, 8, 4, 6}  set\_a.intersection(set\_b)  >> {3, 4, 8} |
| [**issubset()**](https://www.w3schools.com/python/ref_set_issubset.asp) | The issubset() compare two set and return True if the first set is a exact subset (contains all the element) of the second set. | set\_a = {3, 4, 8}  set\_b = {10, 3, 8, 4, 6}  set\_a.issubset(set\_b)  >> True |
| [**issuperset()**](https://www.w3schools.com/python/ref_set_issuperset.asp) | The issuperset() compare two set and return True if the first set is a exact superset (contains all the element) of the second set. | set\_a = {10, 3, 8, 4, 6}  set\_b = {3, 4, 8}  set\_a.issuperset(set\_b)  >> True |
| [**pop()**](https://www.w3schools.com/python/ref_set_pop.asp) | The pop() method removes the item at the index passed as argument and returns the removed item that can be stored in a variable. | set\_a = {10, 3, 8, 4, 6}  last\_num = set\_a.pop(-1)  last\_num >> 6  set\_a >> {10, 3, 8, 4} |
| [**remove()**](https://www.w3schools.com/python/ref_set_remove.asp) | The remove() method removes the item that first match the argument passed and returns the list modified. | set\_a = {10, 3, 8, 4, 6}  set\_a.remove(3)  set\_a >> {4, 6, 8, 10} |
| [**symmetric\_differ**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp)[**ence()**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | The symmetric\_difference() method compares two sets and returns all the items present in fist sets, except the items in common. | set\_a = {10, 3, 8, 4, 6}  set\_b = {3, 4, 8}  set\_a = {10, 3, 8, 4, 6, 5, 34, 7}  set\_b = {3, 4, 8}  result = set\_a.symmetric\_difference(set\_b)  result >> {34, 5, 6, 7, 10} |
| [**union()**](https://www.w3schools.com/python/ref_set_union.asp) | The union()method compares two set and return a set with all the element of the first and the second set. | set\_a = {10, 3, 8, 4, 6}  set\_b = {3, 4, 8}  set\_c = set\_a.union(set\_b)  set\_c >> {3, 4, 6, 8, 10} |

|  |  |  |
| --- | --- | --- |
| [**update()**](https://www.w3schools.com/python/ref_set_update.asp) | The update() method compares two set and return the first set updated with the new element of the second set. | set\_a = {10, 3, 8, 4, 6}  set\_b = {'a', 'b', 'c'}  set\_a.update(set\_b)  set\_a >> {3, 4, 'c', 6, 8, 10, 'a', 'b'} |

**5 points**

**7. Python file methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| **[read()](https://www.w3schools.com/python/ref_file_read.asp)** | The read() method returns the text inside the textfile on which is called. | file = open("file\_example.txt", "r")  print(file.read()) |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | The readline() method returns one line from the file. | file = open("file\_example.txt", "r")  print(file.readline()) |
| [**readlines()**](https://www.w3schools.com/python/ref_file_readlines.asp) | The readline() method returns all lines from the file sotred inside a list. | file = open("file\_example.txt", "r")  print(file.readlines())  >> [‘line1\n’, ‘line2\n’, ‘line3\n’, ‘line4\n’] |
| [**write()**](https://www.w3schools.com/python/ref_file_write.asp) | The write() method writes a specified text to the file the position where is inserted in the file depends on the read mode. > write (start) or append (end). | file = open("file\_example.txt", "w")  file.write(‘Hello World!’)  file.close() |
| [**writelines()**](https://www.w3schools.com/python/ref_file_writelines.asp) | The writelines() method writes a more lines sotred in a list text to the file the position where is inserted in the file depends on the read mode. > write (start) or append (end). | file = open("file\_example.txt", "w")  file.writelines([‘line1\n’, ‘line2\n’, ‘line3\n’, ‘line4\n’])  file.close() |