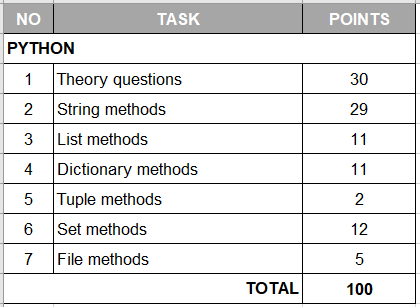
**THEORY QUESTIONS ASSIGNMENT**

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 free open source programming language. Some of its main features include that it is easy and readable, dynamically-typed and object-oriented approach, popular and widely used software programming language.*
2. **Discuss the difference between Python 2 and Python 3.** *Python 2 is an older open source version while Python 3 is most recent. After 2020, Python 2 will be discontinued as Python 2 syntax is comparatively difficult to understand and needs a Unicode string value “u” to store strings. Python 2 was used to implement technical details of PEP (Python Enhancement Proposal). Python 2 uses ASCII while Python 3 uses UTF-8 for it’s core source file encoding.*
3. **What is PEP 8?** *Python Enhancement Proposal 8 is a style guide to structure Python code in terms of code lay-out, strings, whitespaces, trailing commas, comments, naming conventions and recommendations. It can be found here:* <https://www.python.org/dev/peps/pep-0008/>
4. **In computing / computer science what is a program?** *A computer program is a collection of instructions written by a programmer and that can be executed with one or few clicks of a button to perform a specific task.*
5. **In computing / computer science what is a process?** *A process is a computer program in execution. One program has one process associated with it. It contains a program code and its activity.*
6. **In computing / computer science what is cache?** *A cache is a small amount of memory (a type of high-speed random access memory RAM) which is built into a Computer Processing Unit (CPU). It is used to temporarily hold instructions and data that the CPU is likely to reuse.*
7. **In computing / computer science what is a thread and what do we mean by multithreading?** *A single thread in computing means one command is processed at a time. Multi-threading is a widespread programming and execution model that allows multiple threads to exist within one process.*
8. **In computing / computer science what is concurrency and parallelism and what are the differences?** *Concurrency is when multiple computations are run at the same time. It is used for reducing the response time of the system by using a single processing unit, but increases the amount of work finished at a time. Debugging is very hard in concurrency. Parallelism is when the multiple computations are divided into smaller tasks and processed/run simultaneously. It is used to increase the throughput and speed of the computer system by using multiple processors. Debugging is hard but meh simpler than concurrency.*
9. **What is GIL in Python and how does it work?** *GIL stands for Global Interpreter Lock. It allows only one thread to hold the control of the Python interpreter meaning only one thread can be executed at a time.It protects Python objects preventing multiple threads from executing Python byte codes at once and this ensures thread safety.*
10. **What do these software development principles mean: DRY, KISS, BDUF** *DRY stands for “Don’t repeat yourself” - means to reduce repetition in programming; KISS means “Keep it simple, stupid” - means to keep the solutions to the programming problems simple and straight forward; BDUF means “Big Design Up Front” - this is when program designs are perfected before implementing them with the waterfall model of software development.*
11. **What is a Garbage Collector in Python and how does it work?** *Garbage collector is also known as an automatic memory management where programmers no longer need to manage memory themselves. Since programming language uses objects in its programs to perform operations, and objects include simple variables like strings, integers or booleans and complex data structures like lists, hashes or classes. These values of the objects are stored in memory for quick access. Deallocation of these objects is done automatically using garbage collector.   
    How it works: In Python, there are many implementations: CPython, PyPy, Jython (Java-based) or IronPython (C#-based). In the CPython, the following command needs to be run in the terminal:****>>>python -c 'import platform; print(platform.python\_implementation())'***

Or, you can have these lines for both Linux and Windows terminals.

>>> import platform

>>> print(platform.python\_imlplementation())

CPython

*There are two aspects to memory management and garbage collection in CPython: Reference counting and Generational garbage collection.  
Reference counting: a Python object’s reference count is incremented whenever the object is referenced, and it’s decremented when an object is dereferenced. If an object’s reference count is 0, the memory for the object is deallocated.   
The “sys” module from the Python standard library can be used to check reference counts for a particular object.  
Generational garbage collection: There are 2 key concepts: generation and threshold. The Python garbage collector has three generations in total, and an object moves into an older generation whenever it survives a garbage collection process on its current generation.*

1. **How is memory managed in Python?** *Python memory manager manages chunks of memory called “Blocks” . It has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching. There is a private heap where the management is performed by the interpreter itself and the user has no control over it, even if they regularly manipulate object pointers to memory blocks inside that heap.*
2. **What is a Python module?** *A Python module is similar to a code library. It is a file that contains a set of functions, classes and/or variables. It can also include runnable code.*
3. **What is docstring in Python?** *This is a string used to document a Python module, class, function or method. The purpose of this is s that programmers can understand what it does without having to read the details of the implementation. It is a common practice to generate online (html) documentation automatically from docstrings.*
4. **What is pickling and unpickling in Python? Example usage.** *Pickling allows to serialise Python object structures into a byte stream, while unpicking is the where the byte stream is de-serialised and converted into an object hierarchy. Here is an example:  
     
   Pickling:  
   import pickle  
     
   def pickle\_data():  
    data = {‘name’: ‘Karishma’, ‘profession’: ‘Data Analyst’, ‘country’: ‘UK’ }  
    filename = ‘PersonalDetails’  
    outfile = open(filename, ‘wb’)   
   #’wb’ = written in the form of byte objects  
    pickle.dump(data, outfile)  
    outfile.close()  
     
   pickle\_data()  
     
   Unpickling:  
   import pickle  
     
   def unpickling\_data():  
    file = open(filename, ‘rb’)  
   # ‘rb’ = reading in binary mode.  
    new\_data = pickle.load(file)  
    file.close()  
    return new\_data  
     
   print(unpickling\_data())  
     
   The output of unpicking function is:  
   {‘name’: ‘Karishma’, ‘profession’: ‘Data Analyst’, ‘country’: ‘UK’}*
5. **What are the tools that help to find bugs or perform static analysis?** *Pychecker and Pylint perform static analysis that help find bugs in python. Pychecker is an open-source tool that detects bugs from source code and warns about the style and complexity of the bug.*

**How are arguments passed in Python by value or by reference? Give an example.***Arguments are mentioned in parenthesis after a function name.   
Example of by reference:  
def main():  
 arg = 4  
 square(arg)  
 print(arg)  
  
def square(n):  
 n \*= n  
  
main()  
>> 4**By reference: Given a value x which occupies memory space a: Pass x directly in its original memory space a, call this y. Any changes made to y will be reflected in x.  
  
Example of by value:  
def main():  
 n = 9001  
 print(f"Initial address of n: {id(n)}")  
 increment(n)  
 print(f" Final address of n: {id(n)}")*

*def increment(x):  
 print(f"Initial address of x: {id(x)}")  
 x += 1  
 print(f" Final address of x: {id(x)}")*

*main()*

*>> Initial address of n: 140562586057840*

*>> Initial address of x: 140562586057840*

*>> Final address of x: 140562586057968*

*>> Final address of n: 140562586057840  
  
By value: Given a value x which occupies memory space a: Copy x into a new memory space b, call this y. Now any changes made to y will not be reflected in x.*

1. **What are Dictionary and List comprehensions in Python? Provide examples.** *List Comprehension is a handy and faster way to create lists in Python in just a single line of code. Similarly with dictionary comprehension, one can easily create dictionaries.  
   Example:  
   # create list of fruits*

*fruits = ['apple', 'mango', 'banana','cherry']*

*# dict comprehension to create dict with fruit name as keys*

*{f:len(f) for f in fruits}*

*>> {'cherry': 6, 'mango': 5, 'apple': 5, 'banana': 6}*

1. **What is namespace in Python?** *It is a collection of names and the details of the objects references by the names. There are 4 types of namespaces: built-in, global, local and enclosing namespaces.*
2. **What is pass in Python?** *Pass means to provide an argument to a function. It is a null statement; when executed, nothing happens, but the statement avoids getting an error when empty code is not allowed (i.e. in loops, functions, class definitions, if-statements).*
3. **What is unit test in Python?** *Software testing method by which individual units of source code are put under various tests to determine whether they are fit for use. IT determines the quality of the code written.*
4. **In Python what is slicing?** *It is a feature that enables accessing parts of a string, tuple and/or lists. It can also be used to modify or delete the items of mutable sequences such as lists.*
5. **What is a negative index in Python?** *Python arrays are list items are accessed using positive and negative numbers known as index. A positive index begins from 0 (as first index). A negative index accesses elements from the end of the list counting backwards: -1 is the last element, -2 is the second last element, and so on.*
6. **How can the ternary operators be used in python? Give an example.** *In an if-else statement, this is how the syntax reads:  
   if condition:  
    value\_if\_true  
   else:  
    value\_if\_false  
   In Ternary operators can be used to made the code more concise. The entire code is written in one line as:  
   value\_if\_true if condition else value\_if\_false  
   Example:  
   age = input('Enter your age:')  
   ticket\_price = 20 if int(age) >= 18 else 5  
   print(f"The ticket price is {ticket\_price}”)*
7. **What does this mean: \*args, \*\*kwargs? And why would we use it?** *In Python, special symbols can be used to pass a variable number of arguments to a function. These special symbols are \*args and \*\*kwargs. \*args is Non-Keyword Arguments and \*\*kwargs is Keyword Arguments. These special symbols are used when we are unsure about the number of arguments to pass in the functions.  
   Example of \*args:**def adder(\*num):*

*sum = 0*

*for n in num:*

*sum = sum + n*

*print("Sum:",sum)*

*adder(3,5)*

*adder(4,5,6,7)*

*adder(1,2,3,5,6)  
  
>> Sum: 8  
>> Sum: 22  
>> Sum: 17  
  
Example of \*\*kwargs:  
def intro(\*\*data):*

*print("\nData type of argument:",type(data))*

*for key, value in data.items():*

*print("{} is {}".format(key,value))*

*intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)*

*intro(Firstname="John", Lastname="Wood", Email="johnwood@nomail.com", Country="Wakanda", Age=25, Phone=9876543210)  
  
>>  
Data type of argument: <class 'dict'>*

*Firstname is Sita*

*Lastname is Sharma*

*Age is 22*

*Phone is 1234567890*

*Data type of argument: <class 'dict'>*

*Firstname is John*

*Lastname is Wood*

*Email is johnwood@nomail.com*

*Country is Wakanda*

*Age is 25*

*Phone is 9876543210*

1. **How are range and xrange different from one another?** *range() and xrange() are two functions that can be used to iterate a certain number of times in for loops. The only difference is that range() returns a Python list object while xrange() returns an orange object. Also, xrange() is only used in Python 2, range() is used in Python 3 and it behaves like xrange().*
2. **What is Flask and what can we use it for?** *Flask is a web framework providing tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.*
3. **What are clustered and non-clustered index in a relational database?** *A clustered index orders the data phsyically on the disk to match the index. It is useful if you access the data in the table very often via just one column, e.g. via the primary key. A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk.*
4. **What is a ‘deadlock’ a relational database?** *A deadlock is a situation that occurs when processes block each other with resource acquisition and makes no further progress. A real-word example is traffic: which is going in only 1 direction. A bridge is a resource. When deadlock happens, it cannot be resolved easily if 1 car backs up. Several cars need to be backed-up if a deadlock situation occurs.*
5. **What is a ‘livelock’ a relational database?** *Livelock is a deadlock-like situation in which processes block each other with a repeated state change yet make no progress. In SQL Server Live Lock occurs when read transactions are applied on table which prevents write transaction to wait indefinitely. This is different then deadlock as in deadlock both the processes wait on each other. A real-life example of lovelock is when 2 people meet face-to-face in a corridor and each moves aside to let the other pass, but they end up moving from side to side without making any progress because they always move the same way at the same time and never cross each other.*

|  |  |
| --- | --- |
| 1. **Python string methods:**   **describe each method and provide an example** | **29 points** |

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | **returns a string with first character is in capitals (upper case) and the rest is in lower case.** | **txt = “hello, my name is karishma.” x = txt.capitalize() print(x) >> Hello, My Name Is Karishma.** |
| **casefold()** | **similar to lower(), it will return a string with all characters in lower case. It is known to be stronger and more aggressive than lower()** | **txt = "Hello, How Are You?” x = txt.casefold() print(x) >> hello, how are you?** |
| **center()** | **prints the word or text, by taking up the space of the characters mentioned.** | **txt = “banana" x = txt.center(20, “O”) print(x) >> OOOOOOObananaOOOOOOO** |
| **count()** | **counts the number of items in the field/column/string** | **txt = “Hello World” string\_count = txt.count(‘o’) print(string\_count) >> 2** |
| **endswith()** | **returns True if the string ends with the mentioned value. If the mentioned value is not True, then it returns False.** | **txt = “Hi!” x = txt.endswith(“!”) print(x) >> True** |
| **find()** | **finds the first occurrence of the mentioned value. returns a number of the position the mentioned value is. if the value is not found, it returns a -1.** | **txt = "hello, welcome to my house.” x = txt.find(“e”) print(x) >> 1** |
| **format()** | **this string formats the mentioned value(s).** | **txt = "For only {price:.2f} dollars!" print(txt.format(price = 49)) >> For only 49.00 dollars!** |
| **index()** | **returns the position of the first occurrence of the mentioned value. Python count starts from 0.** | **fruits = ['apple', 'banana', 'cherry'] x = fruits.index("cherry") print(x) >> 2** |
| **isalnum()** | **checks if all the characters are alphanumeric in the string. Gives a boolean True or False answer** | **txt = "Company12" x = txt.isalnum() print(x) >> True** |
| **isalpha()** | **returns True or False. checks for if all characters are alphabet letters a-z.** | **txt = "CompanyY" x = txt.isalpha() print(x) >> True** |
| **isdigit()** | **returns True or False. checks for if all characters are numbers** | **txt = "50800" x = txt.isdigit() print(x) >> True** |
| **islower()** | **returns True or False. checks for if all characters are in lowercase.** | **txt = "CompanyY" x = txt.islower() print(x) >> False** |
| **isnumeric()** | **returns True or False. checks for if all characters are numeric (0-9).** | **txt = "50800" x = txt.isnumeric() print(x) >> True** |
| **isspace()** | **returns True or False. checks for if all characters are whitespace characters.** | **txt = " " x = txt.isspace() print(x) >> True** |
| **istitle()** | **returns True or False. checks for if all the words in a text start with a upper case letter and rest of the word is in lower case letters.** | **txt = “Hello World” x = txt.istitle() print(x) >> True** |
| **isupper()** | **returns True or False. checks for if all characters are in uppercase.** | **txt = “Hello World” x = txt.isupper() print(x) >> True** |
| **join()** | **joins all items in an iterable with a specified string. the specified string acts as the separator.** | **myTuple = ("John", "Peter", "Vicky") x = "#".join(myTuple) print(x) >> John#Peter#Vicky** |
| **lower()** | **returns a string with all characters in lower case.** | **txt = "Hello, How Are You?” x = txt.lower() print(x) >> hello, how are you?** |
| **lstrip()** | **returns a copy of the string with any leading characters removed. the “l” stands for “left” so it strips the string from the left side of the argument.** | **txt = "     banana     " x = txt.lstrip() print("of all fruits", x, "is my favorite") >> of all fruits banana     is my favorite** |
| **replace()** | **returns a string after replacing the mentioned character with a new character.** | **txt = "I like bananas" x = txt.replace("bananas", "apples") print(x) >> I like apples** |
| **rsplit()** | **splits the string into a list. the separator that will split the string must be specified. If it is not specified then it will be the same as split() command (default as whitespace).** | **txt = "apple, banana, cherry" x = txt.rsplit(", ") print(x) >> [‘apple’,’banana’,’cherry’]** |
| **rstrip()** | **returns a copy of the string with any leading characters removed. the “r” stands for “right” so it strips the string from the right side of the argument.** | **txt = "     banana     " x = txt.lstrip() print("of all fruits", x, "is my favorite") >> of all fruits     banana is my favorite** |
| **split()** | **splits the string into a list. if a separator is specified then it will use this, otherwise it will split by whitespace.** | **txt = "welcome to the jungle" x = txt.split() print(x) >> [‘welcome’,’to’,’the’,’jungle’]** |
| **splitlines()** | **splits the string into a list. the splitting is done at line breaks.** | **txt = "Thank you for the music\nWelcome to the jungle" x = txt.splitlines() print(x) >> [‘Thank you for the music’,’Welcome to the jungle’]** |
| **startswith()** | **returns True if the string starts with the mentioned value. If the mentioned value is not True, then it returns False.** | **txt = “Hi, how are you!” x = txt.startswith(“Hi”) print(x) >> True** |
| **strip()** | **removes any spaces or mentioned characters at the start or end of a string.** | **txt = "     banana     " x = txt.lstrip() print("of all fruits", x, "is my favorite") >> of all fruits banana is my favorite** |
| **swapcase()** | **converts a case of the string characters from uppercase to lowercase and vice versa.** | **txt = "Hello My Name Is PETER" x = txt.swapcase() print(x) >> hELLO mY nAME iS peter** |
| **title()** | **returns the string that starts with an upper case letter and rest of the word is in lower case letters.** | **txt = “Hello World, welcome” x = txt.title() print(x) >> Hello World Welcome** |
| **upper()** | **returns a string with all characters in upper case.** | **txt = "Hello, How Are You?” x = txt.upper() print(x) >> HELLO, HOW ARE YOU?** |

|  |  |
| --- | --- |
| 1. **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)** | **adds a specific item to the existing/original list.** | **fruits = ["apple", "banana", "cherry"] fruits.append("orange") print(fruits) >> [‘apple’,’banana’,’cherry’,’orange’]** |
| **[clear()](https://www.w3schools.com/python/ref_list_clear.asp)** | **clears the list completely and returns nothing.** | **fruits = ["apple", "banana", "cherry"] fruits.clear() print(fruits) >> []** |
| **[copy()](https://www.w3schools.com/python/ref_list_copy.asp)** | **creates a copy of an existing list.** | **all = [‘apple’,’banana’,’cherry’] fruits = all.copy() print(fruits) >> [‘apple’,’banana’,’cherry’]** |
| **[count()](https://www.w3schools.com/python/ref_list_count.asp)** | **counts the number of items in the field/column/string** | **# an object appears in a list using count() method list1 = [1, 1, 1, 2, 3, 2, 1] # Counts the number of times 1 appears in list1 print(list1.count(1)) >> 4 list2 = ['a', 'a', 'a', 'b', 'b', 'a', 'c', 'b'] # Counts the number of times 'b' appears in list2 print(list2.count('b')) >> 3 list3 = ['Cat', 'Bat', 'Sat', 'Cat', 'cat', 'Mat'] # Counts the number of times 'Cat' appears in list3 print(list3.count('Cat')) >> 2** |
| **[extend()](https://www.w3schools.com/python/ref_list_extend.asp)** | **this built-in function adds more content to the end of an existing list.** | **fruits = ['apple', 'banana', 'cherry'] cars = ['Ford', 'BMW', 'Volvo'] fruits.extend(cars) print(fruits) >> ['apple', 'banana', 'cherry', 'Ford', 'BMW', ‘Volvo']** |
| **[index()](https://www.w3schools.com/python/ref_list_index.asp)** | **returns the position at which the item is found in a list or string.** | **attendees = [‘Alice’,’Bethan’,’Claudia’] attendees\_name = ‘Bethan' print(attendees.index(attendees\_name)) >> 1** |
| **[insert()](https://www.w3schools.com/python/ref_list_insert.asp)** | **this function inserts a mentioned value at the specific position mentioned.** | **fruits = ['apple', 'banana', 'cherry'] fruits.insert(1, "orange") print(fruits) >> ['apple', 'orange', 'banana', ‘cherry']** |
| **[pop()](https://www.w3schools.com/python/ref_list_pop.asp)** | **this removes the element in a list in the specified position. If it is specified to a new variable, then this lists the removed element in the new variable.** | **fruits = ['apple', 'banana', 'cherry'] x = fruits.pop(1) print('fruits: ', fruits) print(‘x: ', x) >> fruits: ['apple', 'cherry'] >> x: banana** |
| **[remove()](https://www.w3schools.com/python/ref_list_remove.asp)** | **this removes a given object from a list.** | **# the first occurrence of 1 is removed from the list list1 = [ 1, 2, 1, 1, 4, 5 ] list1.remove(1)print(list1) >> [2,1,1,4,5]**  **# removes 'a' from list2 list2 = [ 'a', 'b', 'c', 'd' ] list2.remove(‘a’) print(list2) >> [‘b’,’c’,’d’]** |
| **[reverse()](https://www.w3schools.com/python/ref_list_reverse.asp)** | **this reverses objects of the list in place.** | **# a list of numbers list1 = [1, 2, 3, 4, 1, 2, 6] list1.reverse() print(list1) >> [6,2,1,4,3,2,1]  # a list of characters list2 = ['a', 'b', 'c', 'd', 'a', 'a'] list2.reverse() print(list2) >> ['a', 'a', 'd', 'c', 'b', 'a']** |
| **[sort()](https://www.w3schools.com/python/ref_list_sort.asp)** | **sorts the elements in a list in either ascending or descending order.** | **cars = ['Ford', 'BMW', 'Volvo'] cars.sort() print(cars) >> [‘BMW’,’Ford’,’Volvo’]** |

|  |  |
| --- | --- |
| 1. **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)** | **counts the number of times a given value appears in a tuple.** | **thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5) x = thistuple.count(5) print(x) >> 2** |
| **[index()](https://www.w3schools.com/python/ref_tuple_index.asp)** | **searches for the first occurrence of a given value and returns the position.** | **thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5) x = thistuple.index(8) print(x) >> 3** |

|  |  |
| --- | --- |
| 1. **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)** | **removes all elements in a dictionary.** | **car = { "brand": "Ford", "model": "Mustang", "year": 1964} car.clear() print(car) >> {}** |
| **[copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp)** | **a new dictionary is created with a copy of the references from original dictionary.** | **car = { "brand": "Ford", "model": "Mustang", "year": 1964} x = car.copy() print(x) >> {"brand": "Ford", "model": "Mustang", "year": 1964}** |
| **[fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp)** | **returns a dictionary with specified keys and the specified value.** | **x = ('key1', 'key2', 'key3') y = 0 thisdict = dict.fromkeys(x, y) print(thisdict) >> ['key1': 0, 'key2': 0, 'key3': 0]** |
| **[get()](https://www.w3schools.com/python/ref_dictionary_get.asp)** | **returns a value of the item with the specified key.** | **car = { "brand": "Ford", "model": “Mustang", "year": 1964} x = car.get("model") print(x) >> Mustang** |
| **[items()](https://www.w3schools.com/python/ref_dictionary_items.asp)** | **returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.** | **car = {"brand": "Ford", "model": "Mustang", "year": 1964} x = car.items() print(x) >> dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])** |
| **[keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp)** | **There are no parameters to be mentioned. this function returns a view object that displays all the keys in the dictionary.** | **Dictionary1 = {'A': 'Geeks', 'B': 'For', 'C': 'Geeks'} print(Dictionary1.keys()) >> dict\_keys([‘A’,’B’,’C’])** |
| **[pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp)** | **this removes a mentioned element from the dictionary** | **car = {"brand": "Ford","model": "Mustang","year": 1964} car.pop("model") print(car) >> {"brand": "Ford","year": 1964}** |
| **[popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp)** | **this removes the last key-value pair from the dictionary.** | **car = {"brand": "Ford","model": "Mustang","year": 1964} car.pop("model") print(car) >> {'brand': 'Ford', 'model': 'Mustang'}** |
| **[setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp)** | **returns the value of key if the key exists in dictionary. Otherwise, it inserts the key with a value to the dictionary.** | **Dictionary1 = { 'A': 'Geeks', 'B': 'For'} Third\_value = Dictionary1.setdefault('C') print("Dictionary:", Dictionary1) >> Dictionary: {'A': 'Geeks', 'B': 'For', 'C': None} print("Third\_value:", Third\_value) >> Third\_value: None** |
| **[update()](https://www.w3schools.com/python/ref_dictionary_update.asp)** | **this function doesn’t return any value. it modifies (updates) a value associated with a key in the dictionary.** | **{'A': 'Geeks', 'B': 'For', }Dictionary2 = {'B': 'Geeks'}print("Original Dictionary:") print(Dictionary1) >> >> Original Dictionary: {'A': 'Geeks', 'B': ‘For'}**  **Dictionary1.update(Dictionary2)print("Dictionary after updation:") print(Dictionary1) >> Dictionary after updation:{'A': 'Geeks', 'B': 'Geeks'}** |
| **[values()](https://www.w3schools.com/python/ref_dictionary_values.asp)** | **this returns a list of keys or values. Similar to items() which returns a list of (key, value) tuples, that is more efficient way.** | **# numerical values dictionary = {"raj": 2, "striver": 3, "vikram": 4}**  **print(dictionary.values())   >> dict\_values([2,3,4])**  **# string values dictionary = {"geeks": "5", "for": "3", "Geeks": "5"}**  **print(dictionary.values()) >> dict\_values(['5', '3', ‘5'])** |

|  |  |
| --- | --- |
| 1. **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)** | **adds an element to a given set. If the element already exists, then it won’t be added.** | **GEEK = {'g', 'e', 'k'} # adding 's' GEEK.add('s') print('Letters are:', GEEK)  >>  ('Letters are:', set(['k', 'e', 's', 'g']))**  **# adding 's' again GEEK.add('s') print('Letters are:', GEEK) >> ('Letters are:', set(['k', 'e', 's', ‘g']))** |
| **[clear()](https://www.w3schools.com/python/ref_set_clear.asp)** | **removes all elements from a set** | **thisset = {"apple", "banana", "cherry"}thisset.clear() print(thisset) >> set()** |
| **[copy()](https://www.w3schools.com/python/ref_set_copy.asp)** | **returns a copy of the set** | **set1 = {1,2,3} set2 = set1.copy() print(set2) >> {1,2,3}** |
| **[difference()](https://www.w3schools.com/python/ref_set_difference.asp)** | **returns the set difference of two sets.** | **A = {1, 2, 3, 4} B = {2, 3, 9}**  **#A - B = {1, 4} print(A.difference(B)) >> {1,4}  #B - A = {9} print(B.difference(A)) >> {9}** |
| **[intersection()](https://www.w3schools.com/python/ref_set_intersection.asp)** | **returns a set that contains the similarity between 2 or more sets.** | **x = {"apple", "banana", "cherry"} y = {"google", "microsoft", "apple"} z = x.intersection(y) print(z) >> {“apple"}** |
| **[issubset()](https://www.w3schools.com/python/ref_set_issubset.asp)** | **returns True if all items in the set exists in the specified set, otherwise it retuns False.** | **x = {"a", "b", "c"} y = {"f", "e", "d", "c", "b", "a"} z = x.issubset(y) print(z) >> True** |
| **[issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp)** | **returns True if a set has every elements of another set (passed as an argument). If not, it returns False. Set X is said to be the superset of set Y if all elements of Y are in X** | **x = {"f", "e", "d", "c", "b", "a"} y = {"a", "b", "c"} z = x.issuperset(y) print(z) >>True** |
| **[pop()](https://www.w3schools.com/python/ref_set_pop.asp)** | **removes or pops out the first element from the set.** | **fruits = {"apple", "banana", "cherry"}fruits.pop() print(fruits) >> {‘cherry’,’banana}** |
| **[remove()](https://www.w3schools.com/python/ref_set_remove.asp)** | **removes the mentioned element from the set.** | **fruits = {"apple", "banana", “cherry"}fruits.remove("banana) print(fruits) >> {‘apple’,’cherry’}** |
| **[symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp)** | **returns a set that contains all items from both set but not items that a present in both sets.** | **x = {"apple", "banana", "cherry"} y = {"google", "microsoft", "apple"} z = x.symmetric\_difference(y) print(z) >> {'cherry', 'google', 'microsoft', ‘banana'}** |
| **[union()](https://www.w3schools.com/python/ref_set_union.asp)** | **returns a set that contains all items from original set and all items from the mentioned set. Combining all elements in 2 different sets.** | **x = {"apple", "banana", "cherry"} y = {"google", "microsoft", "apple"} z = x.union(y) print(z) >> {'cherry', 'banana', 'microsoft', 'google', ‘apple'}** |
| **[update()](https://www.w3schools.com/python/ref_set_update.asp)** | **updates or modifies a current set with another set, list, string or dictionary** | **x = {"apple", "banana", "cherry"} y = {"google", "microsoft", "apple"}x.update(y) print(x) >> {'banana', 'cherry', 'apple', 'microsoft', ‘google'}** |

|  |  |
| --- | --- |
| 1. **Python file methods:**   **describe each method and provide an example** | **5 points** |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| **[read()](https://www.w3schools.com/python/ref_file_read.asp)** | **reads the data in a file in the form of a string. If a number is specified then it reads those number of bytes.** | **file = open(“dog\_breads.txt”, "r") print(file.read()) >> Pug Jack Russell Terrier English Springer Spaniel German Shepherd Staffordshire Bull Terrier Cavalier King Charles Spaniel Golden Retriever West Highland White Terrier Boxer Border Terrier** |
| **[readline()](https://www.w3schools.com/python/ref_file_readline.asp)** | **reads the first line in a file in the form of a string, until the break point (that is ‘\n’)** | **file = open("new\_file.txt", "r") print(file.readline()) >> Pug** |
| **[readlines()](https://www.w3schools.com/python/ref_file_readlines.asp)** | **reads all the lines in a file in the form of a list** | **file = open("new\_file.txt", "r") print(file.readlines()) >> ['Pug\n', 'Jack Russell Terrier\n', 'English Springer Spaniel\n', 'German Shepherd\n', 'Staffordshire Bull Terrier\n', 'Cavalier King Charles Spaniel\n', 'Golden Retriever\n', 'West Highland White Terrier\n', 'Boxer\n', 'Border Terrier\n’]** |
| **[write()](https://www.w3schools.com/python/ref_file_write.asp)** | **writes a specific text to the file.** | **f = open("demofile2.txt", "a") f.write("Now the file has more content!") f.close()**  **#open and read the file after the appending: f = open("demofile2.txt", "r") print(f.read())  >> Hello! Welcome to demofile2.txt**  **This file is for testing purposes.**  **Good Luck!Now the file has more content!** |
| **[writelines()](https://www.w3schools.com/python/ref_file_writelines.asp)** | **writes the items of a list to the file** | **f = open("demofile3.txt", "a") f.writelines(["See you soon!", "Over and out."]) f.close()  #open and read the file after the appending: f = open("demofile3.txt", "r") print(f.read())  >> Hello! Welcome to demofile2.txt**  **This file is for testing purposes.**  **Good Luck!See you soon!Over and out.** |