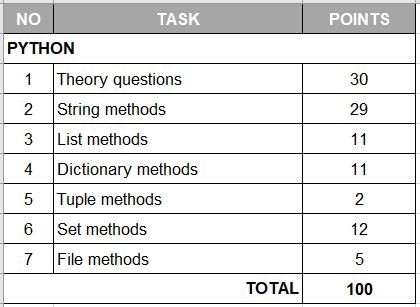
THEORY QUESTIONS ASSIGNMENT

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 an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

* Easy to code
* Free and Open Source
* Object-Oriented Language
* GUI Programming Support
* High-Level Language
* Extensible feature
* Python is Portable language
* Python is Integrated language

1. Discuss the difference between Python 2 and Python 3

* [Division operator](https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/#Division%20operator)
* [print function](https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/#print%20function)
* [Unicode](https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/#Unicode)
* [xrange](https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/#xrange)
* [Error Handling](https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/#Error%20Handling)

1. What is PEP 8?

PEP stands for Python Enhancement Proposal. It is a document that provides guidelines and best practices on how to write Python code. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

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

A collection of instructions that can be executed by a computer to perform a specific task.

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

A process is an instance of a program running in a computer.

1. In computing / computer science what is cache?

Cache is a small amount of memory which is a part of the CPU - closer to the CPU than RAM. It is used to temporarily hold instructions and data that the CPU is likely to reuse.

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

In computer programming, single threading is the processing of one command at a time. Multithreading is a widespread programming and execution model that allows multiple threads to exist within the context of one process. These threads share the process's resources but can execute independently.

1. 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. While parallelism is the task of running multiple computations simultaneously.

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

A global interpreter lock (GIL) is a mechanism to apply a global lock on an interpreter. It is used in computer-language interpreters to synchronize and manage the execution of threads so that only one native thread (scheduled by the operating system) can execute at a time. In a scenario where you have multiple threads, what can happen is that both the thread might try to acquire the memory at the same time, and as a result they would overwrite the data in the memory.

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

* Don’t Repeat Yourself
* Keep It Simple Stupid
* Big Design Up Front

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

Python deletes unwanted objects (built-in types or class instances) automatically to free the memory space. The process by which Python periodically frees and reclaims blocks of memory that no longer are in use is called Garbage Collection.

1. How is memory managed in Python?

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager.

1. What is a Python module?

A module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organised.

1. What is docstring in Python?

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods. It is specified in source code that is used, like a comment, to document a specific segment of code. Unlike conventional source code comments, the docstring should describe what the function does, not how.

* The doc string line should begin with a capital letter and end with a period.
* The first line should be a short description.
* If there are more lines in the documentation string, the second line should be blank, visually separating the summary from the rest of the description.
* The following lines should be one or more paragraphs describing the object’s calling conventions, its side effects, etc.

Declaring Docstrings: The docstrings are declared using ”’triple single quotes”’ or “””triple double quotes””” just below the class, method or function declaration. All functions should have a docstring.

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

“Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. It is the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

1. 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. Pychecker is an opensource tool for static analysis that detects the bugs from source code and warns about the style and complexity of the bug.

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

Python’s argument passing model is neither “Pass by Value” nor “Pass by Reference” but it is “Pass by Object Reference”.

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

Comprehensions provide an elegant method of creating a dictionary or list from an iterable or transforming one dictionary or list into another.

dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

# Put all keys of `dict1` in a list and returns the list

dict1.keys()

dict\_keys(['c', 'd', 'a', 'b'])

# Put all values saved in `dict1` in a list and returns the list

dict1.values()

dict\_values([3, 4, 1, 2])

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Example:

Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name.

Without list comprehension you will have to write a for statement with a conditional test inside:

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x for x in fruits if "a" in x]  
  
print(newlist)

1. What is namespace in Python?

A namespace is a collection of currently defined symbolic names along with information about the object that each name references. You can think of a namespace as a dictionary in which the keys are the object names and the values are the objects themselves

1. What is pass in Python?

Pass is a placeholder or a null statement. The interpreter does not ignore a pass statement, but nothing happens and the statement results into no operation. The pass statement is useful when you don't write the implementation of a function but you want to implement it in the future.

1. What is unit test in Python?

Unit Testing is the first level of software testing where the smallest testable parts of a software are tested. This is used to validate that each unit of the software performs as designed. The unittest test framework is python's xUnit style framework.

1. In Python what is slicing?

Slicing in Python is a feature that enables accessing parts of sequences like strings, tuples, and lists. You can also use them to modify or delete the items of mutable sequences such as lists.

1. What is a negative index in Python?

Python programming language supports negative indexing of arrays, something which is not available in arrays in most other programming languages. This means that the index value of -1 gives the last element, and -2 gives the second last element of an array. The negative indexing starts from where the array ends.

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

The Python ternary operator lets you evaluate whether a condition is true or false. The ternary operator takes up one line of code, which means it is shorter and more concise than a full if…else statement. Conditional statements, such as if statements, allow you to control the flow of your program.

age = 48

discount = True **if** age >= 65 **else** False

**print**(discount)

1. What does this mean: \*args, \*\*kwargs? And why would we use it?

* \*args and \*\*kwargs are special keyword which allows function to take variable length argument.
* \*args passes variable number of non-keyworded arguments list and on which operation of the list can be performed.
* \*\*kwargs passes variable number of keyword arguments dictionary to function on which operation of a dictionary can be performed.
* \*args and \*\*kwargs make the function flexible.

1. How are range and xrange different from one another?

* As range() returns the list, all the operations that can be applied on the list can be used on it. On the other hand, as xrange() returns the xrange.
* Because of the fact that xrange() evaluates only the generator object containing only the values that are required by lazy evaluation, therefore is faster in implementation than range().
* If you want to write code that will run on both Python 2 and Python 3, use range() as the xrange function is deprecated in Python 3
* range() is faster if iterating over the same sequence multiple times.
* xrange() has to reconstruct the integer object every time, but range() will have real integer objects. (It will always perform worse in terms of memory however)

1. What is Flask and what can we use it for?

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application.

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

With a clustered index the rows are stored physically on the disk in the same order as the index. Therefore, there can be only one clustered index. With a non clustered index there is a second list that has pointers to the physical rows.

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

In a database, a deadlock is a situation in which two or more transactions are waiting for one another to give up locks. All activity stops and remains at a standstill unless the DBMS detects the deadlock and aborts one of the transactions.

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

A Live lock is one, where a request for exclusive lock is denied continuously because a series of overlapping shared locks keeps on interfering each other and to adapt from each other they keep on changing the status which further prevents them to complete the task.

**29 points**

**2. Python string methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | Returns a copy of the original string and converts the first character of the string to a capital (uppercase) letter while making all other characters in the string lowercase letters. | text = "hello, welcome to my world."  greeting = text.capitalize()  print(greeting)   * Hello, welcome to my world. |
| **casefold()** | Similar to the lower() method, but the casefold() method is stronger​, more aggressive, meaning that it will convert more characters into lower case | text = "Hello, Welcome To My World."  greeting = text.casefold()  print(greeting)   * hello, welcome to my world. |
| **center()** | Returns a new centered string of the specified length, which is padded with the specified character. The default character is space.  string.center(length, fillchar) | string = "geeks for geeks"    new\_string = string.center(24, '\*')  print(new\_string)   * \*\*\*\*geeks for geeks\*\*\*\*\* |
| **count()** | Returns the number of times a specified value appears in the string.  string.count(value, start, end) | string = "geeks for geeks"  letterEs = points.count(‘e’)  print(letterEs)   * 4 |
| **endswith()** | Returns True if the string ends with the specified value, otherwise False.  string.endswith(value, start, end​) | text = "Hello, welcome to my world."  greeting = text.endswith(‘.’)  print(greeting)   * True |
| **find()** | The find() method finds the first occurrence of the specified value. Returns -1 if the value is not found. | txt = "Hello, welcome to my world."  x = txt.find("e")  print(x)   * 1 |
| **format()** | The format() method formats the specified value(s) and insert them inside the string's placeholder.  The placeholder is defined using curly brackets: {} | txt = "For only {price:.2f} dollars!"  print(txt.format(price = 49))   * For only 49.00 dollars! |
| **index()** | The index() method finds the first occurrence of the specified value. | txt = "Hello, welcome to my world."  x = txt.find("l")  print(x)   * 2 |
| **isalnum()** | The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9). | txt = "Company12"  x = txt.isalnum()  print(x)   * True |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | The isalpha() method returns True if all the characters are alphabet letters (a-z). | txt = "CompanyX"  x = txt.isalpha()  print(x)   * True |
| **isdigit()** | The isdigit() method returns True if all the characters are digits, otherwise False. | txt = "50800"  x = txt.isdigit()  print(x)   * True |
| **islower()** | The islower() method returns True if all the characters are in lower case, otherwise False. | txt = "hello world!"  x = txt.islower()  print(x)   * True |
| **isnumeric()** | The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False. | txt = "565543"  x = txt.isnumeric()  print(x)   * True |
| **isspace()** | The isspace() method returns True if all the characters in a string are whitespaces, otherwise False. | txt = " "  x = txt.isspace()  print(x)   * True |
| **istitle()** | The istitle() method returns True if all words in a text start with a upper case letter, AND the rest of the word are lower case letters, otherwise False. | txt = "Hello, And Welcome To My World!"  x = txt.istitle()  print(x)   * True |
| **isupper()** | The isupper() method returns True if all the characters are in upper case, otherwise False. | txt = "THIS IS NOW!"  x = txt.isupper()  print(x)   * True |
| **join()** | The join() method takes all items in an iterable and joins them into one string. A string must be specified as the separator. | myTuple = ("John", "Peter", "Vicky")  x = "#".join(myTuple)  print(x)   * John#Peter#Vicky |
| **lower()** | The lower() method returns a string where all characters are lower case. Symbols and Numbers are ignored. | txt = "Hello my FRIENDS"  x = txt.lower()  print(x)   * hello my friends |
| **lstrip()** | The lstrip() method removes any leading characters (space is the default leading character to remove) | txt = "     banana     "  x = txt.lstrip()  print("of all fruits", x, "is my favorite")  of all fruits banana     is my favorite |
| **replace()** | The replace() method replaces a specified phrase with another specified phrase. | txt = "I like bananas"  x = txt.replace("bananas", "apples")  print(x)   * "I like apples" |
| **rsplit()** | The rsplit() method splits a string into a list, starting from the right.  If no "max" is specified, this method will return the same as the [split()](https://www.w3schools.com/python/ref_string_split.asp) method.  *string*.rsplit(*separator, maxsplit*) | txt = "apple, banana, cherry"  x = txt.rsplit(", ")  print(x)   * ['apple', 'banana', 'cherry'] |
| **rstrip()** | The rstrip() method removes any trailing characters (characters at the end a string), space is the default trailing character to remove. | txt = " banana "  x = txt.rstrip()  print("of all fruits", x, "is my favorite")   * of all fruits     banana is my favorite |
| **split()** | The split() method splits a string into a list.  You can specify the separator, default separator is any whitespace. | txt = "welcome to the jungle"  x = txt.split()  print(x)   * ['welcome', 'to', 'the', 'jungle'] |
| **splitlines()** | The splitlines() method splits a 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()** | The startswith() method returns True if the string starts with the specified value, otherwise False. | txt = "Hello, welcome to my world."  x = txt.startswith("Hello")  print(x)   * True |
| **strip()** | The strip() method removes any leading (spaces at the beginning) and trailing (spaces at the end) characters (space is the default leading character to remove) | txt = " banana "  x = txt.strip()  print("of all fruits", x, "is my favorite")   * of all fruits banana is my favorite |
| **swapcase()** | The swapcase() method returns a string where all the upper case letters are lower case and vice versa. | txt = "Hello My Name Is KARA"  x = txt.swapcase()  print(x)   * hELLO mY nAME iS kara |

|  |  |  |
| --- | --- | --- |
| **title()** | The title() method returns a string where the first character in every word is upper case. Like a header, or a title.  If the word contains a number or a symbol, the first letter after that will be converted to upper case. | txt = "Welcome to my world"  x = txt.title()  print(x)   * Welcome To My World |
| **upper()** | The upper() method returns a string where all characters are in upper case.   Symbols and Numbers are ignored. | txt = "Hello my friends"  x = txt.upper()  print(x)   * HELLO MY FRIENDS |

**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 appends (adds) an element to the end of the list. | fruits = ["apple", "banana", "cherry"]  fruits.append("orange")  print(fruits)   * ['apple', 'banana', 'cherry', 'orange'] |
| [**clear()**](https://www.w3schools.com/python/ref_list_clear.asp) | The clear() method removes all the elements from a list. | fruits = ["apple", "banana", "cherry"]  fruits.clear()  print(fruits)   * [] |
| [**copy()**](https://www.w3schools.com/python/ref_list_copy.asp) | The copy() method returns a copy of the specified list. | fruits = ["apple", "banana", "cherry"]  x = fruits.copy()  print(x)   * ['apple', 'banana', 'cherry'] |
| [**count()**](https://www.w3schools.com/python/ref_list_count.asp) | The count() method returns the number of elements with the specified value. | points = [1, 4, 2, 9, 7, 8, 9, 3, 1]  winners = points.count(9)  print(winners)   * 2 |
| [**extend()**](https://www.w3schools.com/python/ref_list_extend.asp) | The extend() method adds the specified list elements (or any iterable) to the end of the current 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) | The index() method returns the position at the first occurrence of the specified value. | fruits = ['apple', 'banana', 'cherry']  x = fruits.index("cherry")  print(x)   * 2 |
| [**insert()**](https://www.w3schools.com/python/ref_list_insert.asp) | The insert() method inserts the specified value at the specified position. | fruits = ['apple', 'banana', 'cherry']  fruits.insert(1, "orange")  print(fruits)  ['apple', 'orange', 'banana', 'cherry'] |
| [**pop()**](https://www.w3schools.com/python/ref_list_pop.asp) | The pop() method removes the element at the specified position. | fruits = ['apple', 'banana', 'cherry']  fruits.pop(1)  print(fruits)   * ['apple', 'cherry'] |
| [**remove()**](https://www.w3schools.com/python/ref_list_remove.asp) | The remove() method removes the first occurrence of the element with the specified value. | fruits = ['apple', 'banana', 'cherry']  fruits.remove("banana")  print(fruits)   * ['apple', 'cherry'] |
| [**reverse()**](https://www.w3schools.com/python/ref_list_reverse.asp) | The reverse() method reverses the sorting order of the elements. | fruits = ['apple', 'banana', 'cherry']  fruits.reverse()  print(fruits)   * ['cherry', 'banana', 'apple'] |
| [**sort()**](https://www.w3schools.com/python/ref_list_sort.asp) | The sort() method sorts the list ascending by default. | cars = ['Ford', 'BMW', 'Volvo']  cars.sort()  print(cars)  ['BMW', 'Ford', 'Volvo'] |

**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. | 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) | The index() method finds the first occurrence of the specified value. It raises an exception if the value is not found. | thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)  x = thistuple.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. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  car.clear()  print(car)   * {} |
| [**copy()**](https://www.w3schools.com/python/ref_dictionary_copy.asp) | The copy() method returns a copy of the specified 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) | The fromkeys() method returns a dictionary with the 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) | The get() method returns the 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) | The items() method 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) | The keys() method returns a view object. The view object contains the keys of the dictionary, as a list. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.keys()  print(x)  dict\_keys(['brand', 'model', 'year']) |
| [**pop()**](https://www.w3schools.com/python/ref_dictionary_pop.asp) | The pop() method removes the specified item 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) | The popitem() method removes the item that was last inserted into the dictionary. The removed item is the return value of the popitem() method, as a tuple. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  car.popitem()  print(car)  {'brand': 'Ford', 'model': 'Mustang'} |

|  |  |  |
| --- | --- | --- |
| [**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, insert the key, with the specified value. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.setdefault("model", "Bronco")  print(x)   * Mustang |
| [**update()**](https://www.w3schools.com/python/ref_dictionary_update.asp) | The update() method inserts the specified items to the dictionary.  The specified items can be a dictionary, or an iterable object with key value pairs. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  car.update({"color": "White"})  print(car)   * {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'} |
| [**values()**](https://www.w3schools.com/python/ref_dictionary_values.asp) | The values() method returns a view object. The view object contains the values of the dictionary, as a list. The view object will reflect any changes done to the dictionary. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.values()  print(x)  dict\_values(['Ford', 'Mustang', 1964]) |

**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 set add() method adds a given element to a set if the element is not present in the set. | # set of letters  GEEK = {'g', 'e', 'k'}    # adding 's'  GEEK.add('s')  print('Letters are:', GEEK)    # adding 's' again  GEEK.add('s')  print('Letters are:', GEEK)   * ('Letters are:', set(['k', 'e', 's', 'g'])) * ('Letters are:', set(['k', 'e', 's', 'g']) |
| [**clear()**](https://www.w3schools.com/python/ref_set_clear.asp) | The clear() method removes all elements in a set. | thisset = {"apple", "banana", "cherry"}  thisset.clear()  print(thisset)   * () |
| [**copy()**](https://www.w3schools.com/python/ref_set_copy.asp) | The copy() method copies the set. | fruits = {"apple", "banana", "cherry"}  x = fruits.copy()  print(x)  {'banana', 'apple', 'cherry'} |
| [**difference()**](https://www.w3schools.com/python/ref_set_difference.asp) | The difference() method returns a set that contains the difference between two sets. The returned set contains items that exist only in the first set, and not in both sets. | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.difference(y)  print(z)   * {'banana', 'cherry'} |
| [**intersection()**](https://www.w3schools.com/python/ref_set_intersection.asp) | The intersection() method returns a set that contains the similarity between two or more sets. The returned set contains only items that exist in both sets, or in all sets if the comparison is done with more than two 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) | The issubset() method returns True if all items in the set exists in the specified set, otherwise it returns 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) | The issuperset() method returns True if all items in the specified set exists in the original set, otherwise it returns False. | 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) | The pop() method removes a random item from the set.  This method returns the removed item. | fruits = {"apple", "banana", "cherry"}  fruits.pop()  print(fruits)   * {'cherry', 'apple'} |
| [**remove()**](https://www.w3schools.com/python/ref_set_remove.asp) | The remove() method removes the specified element from the set.  This method is different from the discard() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not. | fruits = {"apple", "banana", "cherry"}  fruits.remove("banana")  print(fruits)   * {'apple', 'cherry'} |
| [**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 returns a set that contains all items from both set, but not the items that are present in both sets.  Meaning: The returned set contains a mix of items that are not present in both sets. | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.symmetric\_difference(y)  print(z)   * {'microsoft', 'google', 'banana', 'cherry'} |
| [**union()**](https://www.w3schools.com/python/ref_set_union.asp) | The union() method returns a set that contains all items from the original set, and all items from the specified set(s).  You can specify as many sets you want, separated by commas.  It does not have to be a set, it can be any iterable object.  If an item is present in more than one set, the result will contain only one appearance of this item. | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.union(y)  print(z)   * {'apple', 'microsoft', 'google', 'cherry', 'banana'} |

|  |  |  |
| --- | --- | --- |
| [**update()**](https://www.w3schools.com/python/ref_set_update.asp) | **update()** function in set adds elements from a set (passed as an argument) to the set. | list1 = [1, 2, 3]  list2 = [5, 6, 7]  list3 = [10, 11, 12]    # Lists converted to sets  set1 = set(list2)  set2 = set(list1)  # Update method  set1.update(set2)  # Print the updated set  print(set1)  # List is passed as an parameter which  # gets automatically converted to a set  set1.update(list3)  print(set1)   * {1, 2, 3, 5, 6, 7} * {1, 2, 3, 5, 6, 7, 10, 11, 12} |

**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) | By default the read() method returns the whole text, but you can also specify how many characters you want to return | f = open("demofile.txt", "r") print(f.read(**5**)) |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | You can return one line by using the readline() method. By looping through the lines of the file, you can read the whole file, line by line. | f = open("demofile.txt", "r") print(f.readline()) |
| [**readlines()**](https://www.w3schools.com/python/ref_file_readlines.asp) | The readlines() method returns a list containing each line in the file as a list item. Use the hint parameter to limit the number of lines returned. If the total number of bytes returned exceeds the specified number, no more lines are returned. | f = open("demofile.txt", "r") print(f.readlines()) |
| [**write()**](https://www.w3schools.com/python/ref_file_write.asp) | The write() method writes a specified text to the file. Where the specified text will be inserted depends on the file mode and stream position.  "a":  The text will be inserted at the current file stream position, default at the end of the file.  "w": The file will be emptied before the text will be inserted at the current file stream position, default 0. | f = open("demofile2.txt", "a") f.write("See you soon!") f.close()  #open and read the file after the appending: f = open("demofile2.txt", "r") print(f.read()) |
| [**writelines()**](https://www.w3schools.com/python/ref_file_writelines.asp) | The writelines() method writes the items of a list to the file. Where the texts will be inserted depends on the file mode and stream position.  "a":  The texts will be inserted at the current file stream position, default at the end of the file.  "w": The file will be emptied before the texts will be inserted at the current file stream position, default 0. | f = open("demofile3.txt", "a") f.writelines(["\nSee you soon!", "\nOver and out."]) f.close()  #open and read the file after the appending: f = open("demofile3.txt", "r") print(f.read()) |