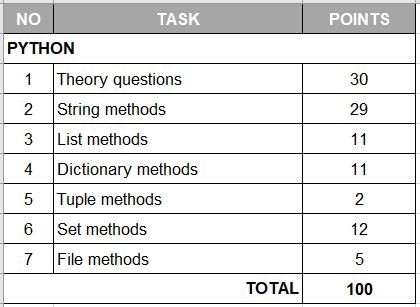
THEORY QUESTIONS ASSIGNMENT

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 high-level programming language with a large range of features. It can be used for application, ­­general, ­­­­­­ web, scripting, artificial intelligence and scientific computing. It is imperative, object oriented, functional, procedural, generic, reflective, event driven and has an aspect-oriented paradigm. It uses a ‘de facto’ standard via Python Enhancement Proposals (PEPs). Python is praised for its ease in code readability but it relies heavily on indentation.*

1. Discuss the difference between Python 2 and Python 3

*- Python 2 is older, comparatively more difficult to understand and considered more outdated than python 3.*

*- In 2020 python 2 was discontinued. However, it is still used to configure management in DevOps.*

*- Python 3 is current, simpler and easier to understand.*

*- Python 3 uses Unicode as it default storing of strings. On the other hand, Python 2 needs to define Unicode string value with “u.” in its stores.*

- *Python 3’s value of variables remains constant. However, inside a ‘for loop’, Python 2’s value of the global variable changes.*

- *Python 3’s exceptions should be enclosed in parenthesis while Python 2 exceptions should be enclosed in notations.*

*- Python 3’s ‘rules of ordering comparisons’ are simplified whereas Python 2 ‘rules of ordering comparison’ are complex.*

*- Python 3 offers Range() function to perform iterations whereas, In Python 2, the xrange() is used for iterations.*

1. What is PEP 8?

*PEP is a design document that gives information to the Python community. It describes new features for Python or its processes or environment. PEP 8 is a document that provides guidelines and best practices on how to write Python code. It helps to improve the readability and consistency of python code.*

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

*A program is a specific set of ordered operations for a computer to perform. Programming is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs.*

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

*A process is an operating system (OS) construct that is a running program. It contains the program code and its activity. The operating system (OS) manages all of the software and hardware on the computer. It performs basic tasks such as file, memory and process management, handling input and output, and controlling peripheral devices such as disk drives and printers. When several computer programs are running at the same time, they all need to access the computer’s central processing unit (CPU), memory and storage. The OS coordinates this and makes sure each program gets what it needs.*

1. In computing / computer science what is cache?

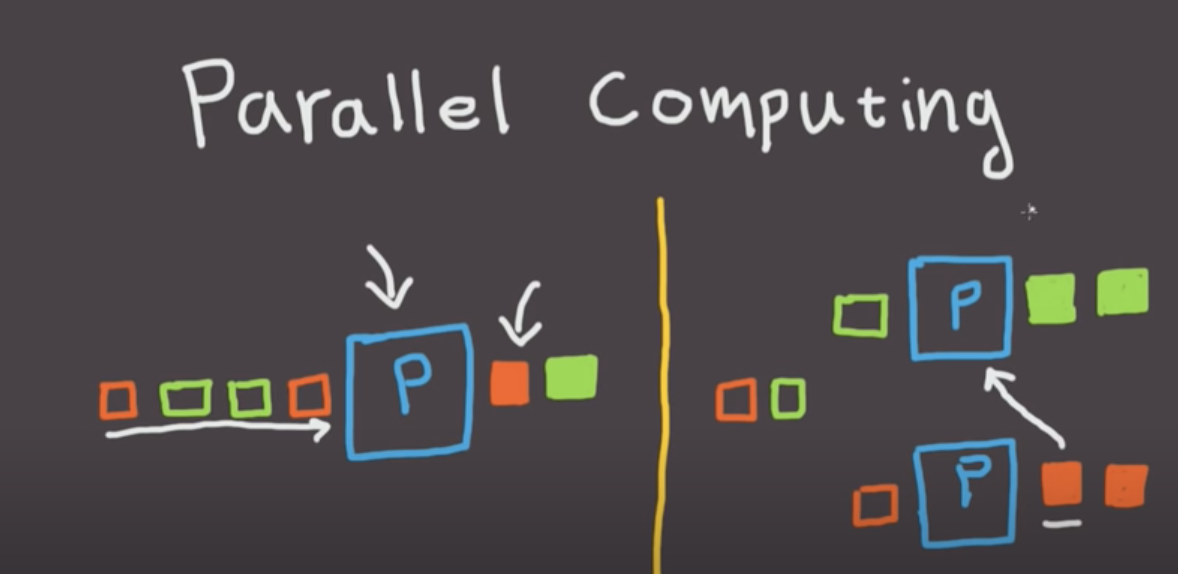
*A cache is memory used to speed memory access, that stores data that is anticipated to be used next or was just used and may be used again. For example: cache may exist on slow physically rotating discs to save information of data that is near to the data just accessed or to hold data without writing it to disc until it is likely not to change again.*

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

*A thread (thread of execution) is a way for a program to split itself into two or more simultaneously running tasks*. *A thread can run in parallel with other threads in the process (multi-threading). It is a way to have parts of a process run simultaneously.*

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

*Parallel Computing - The field of Computer Science that deals with algorithms, techniques and hardware that enables simultaneous execution of many streams of instructions. For example, this means instead of one processor being used, two processors will be executing tasks at the same time. This is effective because parallel computing will be doing a lot of tasks at once so that the program can be sped up. Sometimes, the tasks are not dependent on each other. The problem with this form of computing is more bugs can be introduced, because you’d need to know which instructions need to be divided and what instructors need to be dedicated to each of these processors. You can also have latencies and different problems. But in recent years many computers are running on different core processors (quad core, dual core). This explains the number of processors running at the same time.*

**

*Concurrency Computing – is when multiple sequences of operations are run in overlapping periods of time. Unlike parallel computing, this type of programs is more difficult to write. This is because they deal with constructs like threads and locks and avoid certain issues (race conditions and deadlocks).*

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

*GIL is a python global interpreter lock. This lock allows an individual thread to hold control of the python interpreter. Therefore, only one thread can be executed at any time.*

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

*D.R.Y = Don’t Repeat Yourself*

*K.I.S.S = Systems and design should always be kept simple and not made to be complicated*

*B.D.U.F = Spend more time on designing (completing) a program before implementing it.*

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

*GC releases memory when the object is no longer in use. Can be seen as a form of recycling. This system destroys the unused object and reuses its memory space for new objects. Python has an automated GC.*

1. How is memory managed in Python?

*The Python memory manager manages chunks of memory called “Blocks”.*

*When a collection of blocks are the same size, these are called a “pool”. Pools are created on Arenas. Arenas are chunks of 256kB memory allocated on heap=64 pools. If the objects get destroyed, the memory manager fills this space with a new object of the same size.*

1. What is a Python module?

*Modules are files with the ‘. py’ extension. They contain Python code that can be imported inside another Python Program.*

1. What is docstring in Python?

Python docstrings*are the string literals that appear right after the definition of a function, method, class, or module. For example,*

def add\_numbers(a, b):

'''Takes two parameters

Returns sum.

'''

return a + b

print(add\_numbers(5,10))

*The comment inside the function is a docstring.*

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

*Pickle is a python built-in module that is used for serialising and deserialising python object structures. We can preserve the objects in files and later on we can read it by using python pickle. The process to convert any python object (list, dictionaries) into byte streams like zero and one is called pickling. We can convert the byte stream that is generating through pickling back into the python object by a process called unpickling.*

import pickle

#pickling

carlist = ['Toyota', 'BMW', 'Audi', 'Suzuki', 'Honda']

''' 1. open a file to store the pickle of the car's list

2. use "with open" inside it open a function & write a file name where your pickle will be stored

3.

'''

with open ('carlist.pkl','wb') as carpickle:

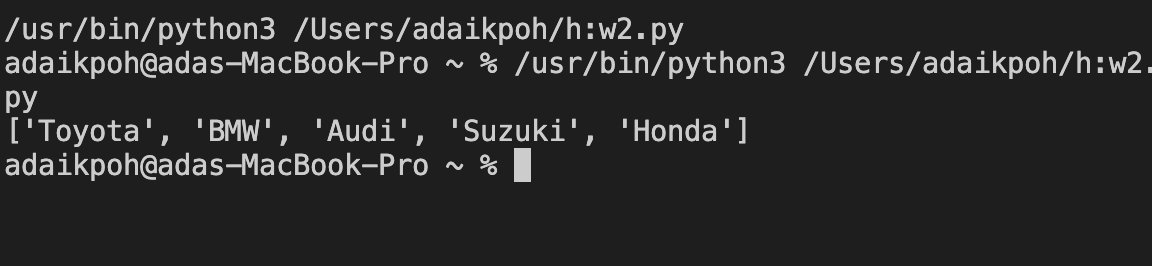
'''dump is a function that creates pickle. It takes 2 arguments. the 1st is the object you want to pickle'''

pickle.dump(carlist,carpickle)

with open ('carlist.pkl','rb') as carpickle:

mycar = pickle.load(carpickle)

print(mycar)



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.*

*Pylint is highly configurable and it acts like special programs to control warnings and errors, it is an extensive configuration file Pylint is also an opensource tool for static code analysis it looks for programming errors and is used for coding standard. it checks the length of each programming line. it checks the variable names according to the project style. it can also be used as a standalone program, it also integrates with python IDEs such as Pycharm, Spyder, Eclipse, and Jupyter*

1. 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.*

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

*A dictionary in Python is a collection of items accessed by a specific key rather than by index.*

*Dictionary comprehension is a method for transforming one dictionary into another dictionary. During this transformation, items within the original dictionary can be conditionally included in the new dictionary and each item can be transformed as needed.*

*A good list comprehension can make your code more expressive and thus, easier to read.*

*Dictionary comprehension is a powerful concept and can be used to substitute for loops and lambda functions. However, not all for loop can be written as a dictionary comprehension but all dictionary comprehension can be written with a for loop. (\*after the theory q’s are completed go back to data camp and read more on this\*) https://www.datacamp.com/community/tutorials/python-dictionary-comprehension*

1. What is namespace in Python?

*A namespace is a system that has a unique name for each and every object in Python. An object might be a variable or a method. Python itself maintains a namespace in the form of a Python dictionary. For instance, a dictionary in which the keys are the object names and the values are the objects themselves.*

1. What is pass in Python?

*In Python, pass is 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 a 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 and ascertains the quality of your code.*

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. ... Slicing enables writing clean, concise, and readable code.*

1. What is a negative index in Python?

*For negative index, -n is the first index, -(n-1) second, last negative index will be – 1.*

*A negative index accesses an element from the end of the list counting backwards.*

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

*The ternary operator is a**way of writing conditional statements**in Python. As the name ternary suggests, this Python operator consists of three operands. The ternary operator can be thought of as a simplified, one-line version of the if-else statement to test a condition.*

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

*\*\*Kwargs in function definitions in python is used to pass a keyworded, variable-length argument list. We use the name kwargs with the double star. The reason is because the double star allows us to pass through keyword arguments (and any number of them).*

*Both Python \*\*args and \*\*kwargs let you pass a variable number of arguments into a function. \*args arguments have no keywords whereas \*\*kwargs arguments each are associated with a keyword.*

*Kwargs is a dictionary of keyword arguments. The (\*\*) allows us to pass any number of keyword arguments. A keyword argument is basically a dictionary. An example of a keyword argument is fun(foo=2,bar=7) . \**

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

*The range() and xrange() are two functions that are used to iterate several times in a ‘*[*for*](https://www.geeksforgeeks.org/loops-and-loop-control-statements-continue-break-and-pass-in-python/)*’ loops in Python.*

*Python 3 has no xrange. However the range function behaves like xrange in Python 2.*

***range()****– This returns a range object (a type of iterable).*

***xrange()****– This function returns the****generator object****that can be used to display numbers only by looping.*

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

*Flask is a web framework written in python used for easy and fast web application development, and for configuring backend applications with the frontend in an easy way. It gives complete control to developers on how to access data. Flask is based on Werkzeug's(WSGI) toolkit and Jinja templating engine.*

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

*A ‘clustered index’ organises the data in a table. It defines the order in which data in a table is stored. Each table can only have one clustered index because you cannot store data in a database table in more than one order (unless you create a copy of that table). This makes it easy and fast for the SQL server to find the data it needs as the data is already sorted. It is best to use clustered indexes for range, group by max, min and count type queries. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.*

*A ‘non-clustered index’ (or regular b-tree index) is an index where the order of the rows does not match the physical order of the actual data. In a non-clustered index, the leaf pages of the index do not contain any actual data, but instead contain pointers to the actual data. A non-clustered index helps you to create a logical order for data rows and uses pointers for physical data files. Allows you to store data pages in the leaf nodes of the index. This indexing method never stores data pages in the leaf nodes of the index.*

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

* *A deadlock is a situation that occurs in Operating Systems when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software. A real-world example would be traffic, which is going only in one direction.*

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

* *A ‘****Livelock’****is a situation where a request for an exclusive lock is denied repeatedly, as many overlapping shared locks keep on interfering each other. The processes keep on changing their status, which further prevents them from completing the task. This further prevents them from completing the task. Livelock is a unique case of resource starvation. Starvation is a situation where all the low priority processes got blocked, and the high priority processes proceed.*
* *An example of Livelock would be two people who meet face-to-face in a corridor, and both of them move aside to let the other pass.*

**29 points**

**2. Python string methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | Change the first letter of a string to upper case | string="you’re a badass"  print(string.capitalize())  You’re a badass |
| **casefold()** | The casefold() method is an aggressive lower() method which converts strings to case folded strings for caseless matching. | Example 1: the German lowercase letter **ß** is equivalent to **ss**. However, since **ß** is already lowercase, the lower() method does nothing to it. But, casefold() converts it to **ss**.  firstString = "der Fluß"  secondString = "der Fluss"  # ß is equivalent to ss  if firstString.casefold() == secondString.casefold():  print('The strings are equal.')  else:  print('The strings are not equal.')  The strings are equal.  Example 2: string = "Just keep smiling"  # print lowercase string  print("Lowercase string:", string.casefold())  Lowercase string: just keep smiling |
| **center()** | The center() method will center align the string, using a specified character (space is default) as the fill character. | string.center(length, character)  txt = "boobies"  x = txt.center(20, "O")  print(x)  OOOOOOboobiesOOOOOOO |
| **count()** | The count() method returns the number of elements with the specified value. | fruits = ['apple', 'banana', 'cherry']  x = fruits.count("cherry")  print(x)  1 |
| **endswith()** | The endswith() method returns True if a string ends with the specified suffix. If not, it returns False. | message = 'Marriage is great but can be hard'  # check if the message ends with fun  print(message.endswith('soft'))  # Output: False  False |
| **find()** | The find() method finds the first occurrence of the specified value.  The find() method returns -1 if the value is not found. The find() method is almost the same as the [index()](https://www.w3schools.com/python/ref_string_index.asp) method, the only difference is that the index() method raises an exception if the value is not found. | txt = "Look for the rainbow in every dark cloud."  x = txt.find("e")  print(x)  11 |
| **format()** | Python String format() is a function used to replace, substitute, or convert the string with placeholders with valid values in the final string. It is a built-in function of the Python string class, which returns the formatted string as an output. The placeholders inside the string are defined in curly brackets. | “Welcome to CFG {}”.format(‘value here’) |
| **index()** | The index() method returns the index of the specified element in the list. | animals = ['cat', 'dog', 'rabbit', 'horse']  # get the index of 'dog'  index = animals.index('dog')  print(index)  # Output: 1  1 |
| **isalnum()** | The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).  Example of characters that are not alphanumeric: (space)!#%&? etc. | txt = "Company 12"  x = txt.isalnum()  print(x)  False |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | The isalpha() method returns True if all the characters are alphabet letters (a-z).  Example of characters that are not alphabet letters: (space)!#%&? etc. | txt = "Company10"  x = txt.isalpha()  print(x)  False |
| **isdigit()** | The isdigit() method returns True if all the characters are digits, otherwise False.  Exponents, like ², are also considered to be a digit. | a = "\u0030" #unicode for 0  b = "\u00B2" #unicode for ²  print(a.isdigit())  print(b.isdigit())  True  True |
| **islower()** | The islower() method returns True if all the characters are in lower case, otherwise False.  Numbers, symbols and spaces are not checked, only alphabet characters. | a = "How to get away murder!"  b = "Scandal"  c = "Grey's Anatomy"  print(a.islower())  print(b.islower())  print(c.islower())  False  False  False |
| **isnumeric()** | The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False.  Exponents, like ² and ¾ are also considered to be numeric values.  "-1" and "1.5" are NOT considered numeric values, because all the characters in the string must be numeric, and the “–” and the “.” are not. | a = "\u0030" #unicode for 0 b = "\u00B2" #unicode for &sup2; c = "10km2" d = "-1" e = "1.5"  print(a.isnumeric()) print(b.isnumeric()) print(c.isnumeric()) print(d.isnumeric()) print(e.isnumeric())  True  True  False  False  False |
| **isspace()** | Python String isspace() is a built-in method used for string handling. The isspace() method returns “True” if all characters in the string are whitespace characters, Otherwise, It returns “False”. This function is used to check if the argument contains all whitespace characters such as: ' ' – Space   * ‘ ‘ – Space * ‘\t’ – Horizontal tab * ‘\n’ – Newline * ‘\v’ – Vertical tab * ‘\f’ – Feed * ‘\r’ – Carriage return | string = 'MyHeartGotAMindOfItsOwn'    print(string.isspace())  False |
| **istitle()** | The istitle() method checks whether each word's first character is upper case and the rest are in lower case or not. It returns True if a string is title cased; otherwise, it returns False. The symbols and numbers are ignored. | >>> greet='Hello World'  >>> greet.istitle()  True  >>> greet='Hello WORLD'  >>> greet.istitle()  False  >>> greet='hello world'  >>> greet.istitle()  False  >>> greet='HelloWorld'  >>> greet.istitle()  False  >>> s='Python Is A Programming Language'  >>> s.istitle()  True |
| **isupper()** | The isupper() method checks whether all the case-based characters (letters) of the string are uppercase. | str = "ROCKING AND VIBING....OK!!!"  print (str.isupper())  str = "THIS is string example....wow!!!"  print (str.isupper())  True  False |
| **join()** | The join() method takes all items in an iterable and joins them into one string.  A string must be specified as the separator. | myDict = {"name": "Jesus", "country": "Heaven"}  mySeparator = "TEST"  x = mySeparator.join(myDict)  print(x)  nameTESTcountry |
| **lower()** | lower() is a built-in Python method primarily used for string handling. The lower() method takes no arguments and returns the lowercased strings from the given string by converting each uppercase character to lowercase. If there are no uppercase characters in the given string, it returns the original string. | txt = "Hello my FRIENDS"  x = txt.lower()  print(x)  hello my friends |
| **lstrip()** | This method is used to delete all the leading characters mentioned in its argument. | str = "---software---"  print ("String after stripping all the '-' is : ", end = "")  print(str.lstrip('-'))  String after stripping all the '-' is : software--- |
| **replace()** | The replace() method replaces a specified phrase with another specified phrase. | txt = "Monkeys eat bananas"  x = txt.replace("bananas", "apples")  print(x) |
| **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. | txt = "apple, banana, cherry"  x = txt.rsplit(", ")  print(x)  ['apple', 'banana', 'cherry'] |
| **rstrip()** | This method is used to delete all the trailing characters mentioned in its argument | str = "---software---"  print ("String after stripping all the '-' is : ", end = "")  print(str.rstrip('-'))  String after stripping all the '-' is : ---software |
| **split()** | The split() method returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally limiting the number of splits to num. | str = "this is string example....wow!!!"  print (str.split( ))  print (str.split('i',1))  print (str.split('w'))  ['this', 'is', 'string', 'example....wow!!!']  ['th', 's is string example....wow!!!']  ['this is string example....', 'o', '!!!'] |
| **splitlines()** | The splitlines() method returns a list with all the lines in string, optionally including the line breaks (if num is supplied and is true). | str = "this is \nstring example....\nwow!!!"  print (str.splitlines( ))  ['this is ', 'string example....', 'wow!!!'] |
| **startswith()** | The startswith() method checks whether the string starts with str, optionally restricting the matching with the given indices start and end. | str = "this is string example....wow!!!"  print (str.startswith( 'this' ))  print (str.startswith( 'string', 8 ))  print (str.startswith( 'this', 2, 4 ))  True  True  False |
| **strip()** | This method is used to delete all the leading and trailing characters mentioned in its argument | str = "---software---"  print ("String after stripping all the '-' is : ", end = "")  print(str.strip('-'))  String after stripping all the '-' is : software |
| **swapcase()** | The swapcase() method returns a string where all the upper case letters are lower case and vice versa. | txt = "Hello My Name Is Ada"  x = txt.swapcase()  print(x)  hELLO mY nAME iS aDA |

|  |  |  |
| --- | --- | --- |
| **title()** | The title() function in python is the Python String Method which is used to convert the first character in each word to Uppercase and remaining characters to Lowercase in the string and returns a new string. | txt = "Functional programming in java"  x = txt.title()  print(x)  Functional Programming In Java |
| **upper()** | The upper() method returns a string where all characters are in upper case.   Symbols and Numbers are ignored. | txt = "Cards against humanity"  x = txt.upper()  print(x)  CARDS AGAINST HUMANITY |

**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 in python adds a single item to the existing list. It doesn't return a new list of items but will modify the original list by adding the item to the end of the list. After executing the method append on the list the size of the list increases by one. | a = ["apple", "banana", "cherry"] b = ["Ford", "BMW", "Volvo"] a.append(b)  ['apple', 'banana', 'cherry', ['Ford', 'BMW', 'Volvo']] |
| [**clear()**](https://www.w3schools.com/python/ref_list_clear.asp) | The clear() method removes all the elements from a list. | fruits = ['apple', 'banana', 'cherry', 'orange']  fruits.clear()  print(fruits)  [] |
| [**copy()**](https://www.w3schools.com/python/ref_list_copy.asp) | The Python copy() method creates a copy of an existing list.  The copy() method is added to the end of a list object and so it does not accept any parameters.  Copy() returns a new list.  Python includes a built-in function to support creating a shallow copy of a list: copy(). | # A Set of names  names = {"Steve", "Rick", "Negan"}  # copying using the copy() method  names2 = names.copy()  # adding "Glenn" to the new set  names2.add("Glenn")  # removing "Negan" from the old set  names.remove("Negan")  # displaying both the sets  print("Old Set is:", names)  print("New Set is:", names2)  Old Set is: {'Steve', 'Rick'}  New Set is: {'Negan', 'Glenn', 'Steve', 'Rick'} |
| [**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]  x = points.count(9)  print(x)  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']  points = (1, 4, 5, 9)  fruits.extend(points)  print(fruits)  ['apple', 'banana', 'cherry', 1, 4, 5, 9] |
| [**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']  x = fruits.pop(1)  print(x)  banana |
| [**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.  You can also make a function to decide the sorting criteria(s). | 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.  The index() method 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')  thisdict = dict.fromkeys(x)  print(thisdict)  {'key1': None, 'key2': None, 'key3': None} |
| [**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.  The view object will reflect any changes done to the dictionary, see example below. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.items()  car["year"] = 2018  print(x)  dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2018)]) |
| [**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.  The view object will reflect any changes done to the dictionary, see example below. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.keys()  car["colour"] = "white"  print(x)  dict\_keys(['brand', 'model', 'year', 'colour']) |
| [**pop()**](https://www.w3schools.com/python/ref_dictionary_pop.asp) | The pop() method removes the specified item from the dictionary.  The value of the removed item is the return value of the pop() method, see example below. | 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. I  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, see example below | 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({"colour": "White"})  print(car)  {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'colour': '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()  car["year"] = 2018  print(x)  dict\_values(['Ford', 'Mustang', 2018]) |

**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) | Adds an element to the set | Tribes = {"Igbo", "Yoruba", "Hausa"}  Tribes.add("Ashanti")  print(Tribes)  {'Hausa', 'Igbo', 'Ashanti', 'Yoruba'} |
| [**clear()**](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set | Tribes = {"Igbo", "Yoruba", "Hausa"}  Tribes.clear()  print(Tribes)  set() |
| [**copy()**](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set | Greatest\_Beyonce\_Hits = {"Formation", "Hold up", "Me myself and i"}  x = Greatest\_Beyonce\_Hits.copy()  print(x)  {'Formation', 'Me myself and i', 'Hold up'} |
| [**difference()**](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets.  For example the adjacent example returns a set that contains the items that only exist in set x, and not in set y: | x = {"beach", "skiing", "hiking"}  y = {"spa days", "mountain climbing", "beach"}  z = x.difference(y)  print(z)  {'skiing', 'hiking'} |
| [**intersection()**](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two or more sets | x = {"beach", "skiing", "hiking"}  y = {"spa days", "mountain climbing", "beach"}  z = x.intersection(y)  print(z)  {'beach'} |
| [**issubset()**](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not | 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 whether this set contains another set or not | 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 an element from the set | emotions = {"happy", "sad", "excited"}  emotions.pop()  print(emotions)  {'sad', 'happy'} |
| [**remove()**](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specifies element.  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*. | emotions = {"happy", "sad", "excited"}  emotions.remove("sad")  print(emotions)  {'excited', 'happy'} |
| [**symmetric\_differ**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp)[**ence()**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.symmetric\_difference(y)  print(z)  {'banana', 'microsoft', 'google', 'cherry'} |
| [**union()**](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets | x = {"apple", "banana", "cherry"} y = {"google", "microsoft", "apple"}  z = x.union(y)  print(z)  {'banana', 'google', 'apple', 'cherry', 'microsoft'} |

|  |  |  |
| --- | --- | --- |
| [**update()**](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with another set, or any other iterable | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  x.update(y)  print(x)  {'banana', 'apple', 'cherry', 'google', 'microsoft'} |

**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 specified number of bytes from the file. Default is -1 which means the whole file. | f = open("demofile.txt", "r") print(f.read(33)) |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | The readline() method returns one line from the file.  You can also specified how many bytes from the line to return, by using the size parameter. | f = open("demofile.txt", "r") print(f.readline()) 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(33)) |
| [**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("\nSee 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()) |