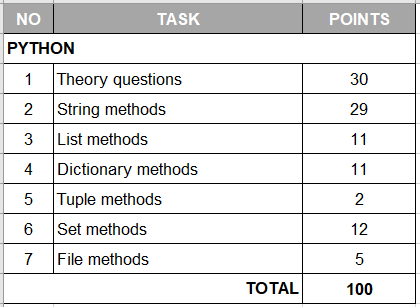
**THEORY QUESTIONS ASSIGNMENT**

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 an object-oriented and high-level coding or programming language.

1. Discuss the difference between Python 2 and Python 3
2. What is PEP 8?
3. **In computing / computer science what is a program?**

Programmes are a set of instructions that inform the computer how to behave/what actions to carry out.

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

A process is again a set of instructions that will lead up to a desired outcome.

1. **In computing / computer science what is cache?**

Cache is a bit of computer memory which is used to temporarily store programmes/data or processes that are used often.

1. In computing / computer science what is a thread and what do we mean by multithreading?
2. In computing / computer science what is concurrency and parallelism and what are the differences?
3. **What is GIL in Python and how does it work?**

GIL is an acronym for Global Interpreter Lock.

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

DRY means Don’t Repeat Yourself and essentially means software should be as efficient as possible by avoiding repetition as much as possible and writing logical, short (as much as possible) parts of code.

KISS is Keep It Simple Stupid, meaning to avoid overcomplicating software or code by writing simple, efficient and short methods of functions.

BDUF is Big Design Up Front. This is an approach to development, often linked to the waterfall way of working which requires programme design to be carried out and completed before implementing and working on the code itself to ensure it works.

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

The Garbage Collector

1. How is memory managed in Python?
2. **What is a Python module?**

A module is code that someone else has written and can be imported for use in python programmes.

1. **What is docstring in Python?**

A docstring is a string that is written right after the definition of a function which can be used to explain or summarise aspects of the given function. They are similar to a comment but are perhaps more useful as they can later be accessed by using \_\_doc\_\_ whereas comments are ignored by the interpreter.

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

Pickling is used to convert an object in python into a byte stream (essentially a sequence) which can make data transfer and storage easier. Unpickling is the process of converting the byte stream back into an object.

To pickle:

friends = ['Rosa', 'Alice', 'Anna', 'Mhari']

with open('our\_new\_file\_for\_writing.txt', 'wb') as fh:

pickle.dump(friends)

To unpickle:

pickle\_reverse = open('our\_new\_file\_for\_writing.txt', 'rb')

list\_a = pickle.load(pickle\_reverse)

print(list\_a)

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

Libraries e.g. Pandas for analysis

1. How are arguments passed in Python by value or by reference? Give an example.
2. What are Dictionary and List comprehensions in Python? Provide examples.
3. **What is namespace in Python?**

A namespace is where all the names of all objects, variables etc used can be found in one collection. Essentially, the namespace is a kind of dictionary.

1. **What is pass in Python?**

This method is used when a statement is required to ensure Python does not throw a syntax error, but where the user has no desire for any code to execute. Another reason it can be useful is in development, when unsure of what specific code is required in that space. This will still let the programme run to check other parts of it.

1. **What is unit test in Python?**

Unit testing takes individual small portions of an overall programme and tests them to see if they work on an isolated basis. This is an efficient way of testing all eventualities of a programme without having to manually do this.

1. **In Python what is slicing?**

Slicing is a method of extracting portions ( or ‘slices’ ) of a string or list. It is used by defining a start index and an end index, with the option to also define an argument for slice stepping which determines the increment at which we are slicing. e.g. print(b[3:5:2]) will return every second value between character position 3 and 5.

1. **What is a negative index in Python?**

A negative index in python allows you to locate a value in a string/list/dictionary/set but counts from the end of the object.

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

**A ternary operator is similar to if-else statements in that it is evaluating whether a condition has been met before carrying out a method, but in this case, the ternary operator is written on one line.**

1. What does this mean: \*args, \*\*kwargs? And why would we use it?
2. How are range and xrange different from one another?
3. What is Flask and what can we use it for?
4. What are clustered and non-clustered index in a relational database?
5. **What is a ‘deadlock’ a relational database?**

Databases are said to be in deadlock when there are 2 or more different tasks waiting on each other to progress and thus, no progress can be made. This means that none of the waiting tasks can be finished.

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

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

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | **This will capitalise the first letter of the first word in the string** | str.capitalize('cats')  ‘Cats’ |
| **casefold()** | **This method will return a string with all lower case characters** | **str.casefold('CATS')**  **'cats'** |
| **center()** | **This will centre the string with x number of filler characters surrounding it (in my example this will be 30 spaces wrapping around our string).** | **text = "cats"**  **x = text.center(30)**  **print(x)**  **cats** |
| **count()** | **This method will count the number of time the specified value (or word) appears in the defined string.** | string = 'I went on a walk at the weekend. I love to walk with my friends'  x = string.count('walk')  print(x)  2 |
| **endswith()** | **This returns true or false depending on whether the condition is met. In this case, the condition is that the string ends in the specified value (e.g. d) and so returns True.** | string = 'My cat is called Treacle and she is 13 years old'  x = string.endswith('d')  print(x)  True |
| **find()** | **This method will locate the first instance of the specified value in the string. You can also add parameters to search within a specific portion of the string. If the value isn’t present in the string, the output will be -1.** | **string = 'My cat is called Treacle and she is 13 years old'**  **x = string.find('d')**  **print(x)**  **15** |
| **format()** | **This method will format and insert values into a string using curly brackets {} which act as a sort of placeholder for where we require the information to be placed.** | cats = 3  cans\_per\_day = 2  total\_cans = cats \* cans\_per\_day  output = '{} cats will need {} cans of cat food.’.format(cats, total\_cans)  print(output)  \*\*\*\*  3 cats will need 6 cans of cat food. |
| **index()** | **Very similar to the find() method as it will locate the first instance of the specified value in the string. The difference between the 2 is that while find() will return -1 if there is no presence of the value, the index() method simply returns an error.** | **string = 'My cat is called Treacle and she is 13 years old'**  **x = string.index('d')**  **print(x)**  **15** |
| **isalnum()** | **This method outputs True if all characters in the defined string are ‘alphanumeric’—essentially a letter or a number. If there is a different character (e.g ! ?) then it will return False.** | string2 = 'Treacle47'  print(string2.isalnum())  \*\*\*  True |
| **isalpha()** | **Similar to above, except it is only returned as True if all characters are letters only.** | string2 = 'Treacle'  print(string2.isalpha())  \*\*\*  True |
| **isdigit()** | **Again, very similar to the above methods. In this case, Python will return True if all characters are numeric. Any letters or special characters will return False.** | string2 = '12345'  print(string2.isdigit())  \*\*\*  True |
| **islower()** | **This method will return True if all characters in the string are lower case. Any upper or capitalised characters will cause a False output.** | string2 = 'treacle'  print(string2.islower())  \*\*\*  True |
| **isnumeric()** | **This method returns True if all characters in the string are numeric. Important to note that values with decimal points will cause a False output as the ‘.’ Is not a numeric value.** | string2 = '127683'  print(string2.isnumeric())  \*\*\*  True |
| **isspace()** | **This method will give a True output if all characters in the string are empty spaces. If there is any other characters present, it will return False.** | string2 = ' '  print(string2.isspace())  \*\*\*  True |
| **istitle()** | **This checks whether the string is written using title formatting (i.e. the first letter of all words is capitalised and the rest of the characters are lower) and will return True if this is the case. If not, Python will output False.** | **string2 = 'Harry Potter And The Order Of The Phoenix'**  **print(string2.istitle())**  **True** |
| **isupper()** | **This method returns True if all characters in the string are uppercase. Any exception to this will produce a False output.** | **string2 = 'BELFAST'**  **print(string2.isupper())**  **True** |
| **join()** | **This method will join a set of strings together into one string, separated by a character that is first defined (in the example, this is //).** | names = ('Rosa', 'Alice', 'Anna', 'Mhari')  print('//'.join(names))  \*\*\*  Rosa//Alice//Anna//Mhari |
| **lower()** | **Will change all characters in the string to lower case.** | **Cat’.lower**  **cat** |
| **lstrip()** | **This method allows us to remove any leading characters and print the string without these present. You can define which characters you want to remove from the string in the argument.** | string = "?????.....??????,,,,sjkI am meeting my friend"  print(string.lstrip('?.,sjk'))  \*\*\*  I am meeting my friend |
| **replace()** | **This method allows us to replace a value in the string with a new character/word. The first defined value in the replace() function will be the section of the existing string you are wanting to replace, while the second value will be the new string you are wanting to insert.** | names = ('Rosa, Alice, Anna, Mhari')  print(names.replace('Anna', 'Erin'))  \*\*\*  Rosa, Alice, Erin, Mhari |
| **rsplit()** |  |  |
| **rstrip()** |  |  |
| **split()** |  |  |
| **splitlines()** |  |  |
| **startswith()** | **This method will return True if the string begins with the specified value. If this is not the case, we will receive a False output. Note: it is case sensitive.** | string = 'My cat is called Treacle and she is 13 years old'  print(string.startswith('My'))  \*\*\*  True |
| **strip()** | **This method removes all characters (both leading and following) that you specify within your parameters and allows you to leave the desired string.** | string3 = '..... klk apple, banana, pear, grapes ///????'  print(string3.strip(' .kl/?'))  \*\*\*  apple, banana, pear, grapes |
| **swapcase()** | **This method allows us to switch the formatting on an entered string i.e uppercase will be converted to lower case and vice versa.** | string4 = 'Harry potter aNd tHE Chamber oF secrets'  print(string4.swapcase())  \*\*\* hARRY POTTER AnD The cHAMBER Of SECRETS |
| **title()** | **This will change all characters in the string to a title case** | **Harry potter and the chamber of secrets’.title**  **Harry Potter and the Chamber of Secrets** |
| **upper()** | **Will change all characters in the string to upper case** | **dog’.upper**  **DOG** |

|  |  |
| --- | --- |
| 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)** | **This method allows the user to add another element to the end of an existing list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.append('Erin')**  **print(friends)**  **['Rosa', 'Alice', 'Anna', 'Mhari', 'Erin']** |
| **[clear()](https://www.w3schools.com/python/ref_list_clear.asp)** | **This method will delete (or clear) all values or characters from within a list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  friends.clear()  #Using print to show empty list  print(friends)  \*\*\*  [] |
| **[copy()](https://www.w3schools.com/python/ref_list_copy.asp)** | **This method copies a list and will return it.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  print(friends.copy()  \*\*\*  ['Rosa', 'Alice', 'Anna', ‘Mhari'] |
| **[count()](https://www.w3schools.com/python/ref_list_count.asp)** | **This method will return the number of times a defined value appears in a list.** | **shopping\_list = ['Eggs', 'Avocado', 'Chocolate', 'Tomatoes', 'Cat food', 'Eggs']**  print(shopping\_list.count(‘Eggs’))  \*\*\*  2 |
| **[extend()](https://www.w3schools.com/python/ref_list_extend.asp)** | **This allows us to add one list onto the end of another list, essentially combining the two.** | friends = ['Rosa', 'Alice', 'Anna', 'Mhari']  numbers= ['1', '4', '13']  friends.extend(numbers)  print(friends)  \*\*\*  ['Rosa', 'Alice', 'Anna', 'Mhari', '1', '4', '13'] |
| **[index()](https://www.w3schools.com/python/ref_list_index.asp)** | **This method will locate the first instance of the specified value in the list. Importantly, it starts counting from 0.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  print(friends.index('Mhari'))  **\*\*\***  **3** |
| **[insert()](https://www.w3schools.com/python/ref_list_insert.asp)** | **This allows us to insert a specified value into a specified place in a list. So in our example, we are inserting the string ‘Erin’, as the 4th element in our friends list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.insert(3, 'Erin')**  **print(friends)**  **['Rosa', 'Alice', 'Anna', 'Erin', 'Mhari']** |
| **[pop()](https://www.w3schools.com/python/ref_list_pop.asp)** | **The pop() method will remove the value which is positioned at the defined location (or index) in the list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.pop(1)**  **print(friends)**  **['Rosa', 'Anna', 'Mhari']** |
| **[remove()](https://www.w3schools.com/python/ref_list_remove.asp)** | **This method allows us to remove a specified item from our list. We must enter the value we are wanting to remove rather than just the position in the list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.remove('Anna')**  **print(friends)**  **['Rosa', 'Alice', 'Mhari']** |
| **[reverse()](https://www.w3schools.com/python/ref_list_reverse.asp)** | **This method reverses the order of a list.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.reverse()**  **print(friends)**  **['Mhari', 'Anna', 'Alice', 'Rosa']** |
| **[sort()](https://www.w3schools.com/python/ref_list_sort.asp)** | **This method will sort the list in alphabetical order. To reverse this alphabetically, we must add reverse=True within our sort() method.** | **friends = ['Rosa', 'Alice', 'Anna', 'Mhari']**  **friends.sort()**  **print(friends)**  **['Alice', 'Anna', 'Mhari', 'Rosa']** |

|  |  |
| --- | --- |
| 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)** | **This method will tell us the number of occurrences of a specific value within a tuple.** | **tuple\_a = (4, 12, 16, 17, 98, 2, 4, 2, 4)**  **print(tuple\_a.count(4))**  **\*\*\***  **3** |
| **[index()](https://www.w3schools.com/python/ref_tuple_index.asp)** | **This will locate the first instance of the specified value in the tuple, and if it is not present, will return an error.** | **tuple\_a = (4, 12, 16, 17, 98, 2, 4, 2, 4)**  **print(tuple\_a.index(98))**  **\*\*\***  **4** |

|  |  |
| --- | --- |
| 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)** | **This method removes all items from a dictionary** | **fruits = [**  **{"name": "apple", "colour": "red", "price": 0.15},**  **{"name": "banana", "colour": "yellow", "price": 0.20},**  **{"name": "pear", "colour": "green", "price": 0.40},**  **{'name': 'grapes', 'colour': "purple", 'price': 1.00}**  **]**  **fruits.clear()**  **print(fruits)**  **\*\*\***  **[]** |
| **[copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp)** | **This method will make an exact copy of the dictionary in question.** | **fruits = [**  **{"name": "apple", "colour": "red", "price": 0.15},**  **{"name": "banana", "colour": "yellow", "price": 0.20},**  **{"name": "pear", "colour": "green", "price": 0.40},**  **{'name': 'grapes', 'colour': "purple", 'price': 1.00}**  **]**  **fruits.copy()**  **print(fruits)**  **[{'name': 'apple', 'colour': 'red', 'price': 0.15}, {'name': 'banana', 'colour': 'yellow', 'price': 0.2}, {'name': 'pear', 'colour': 'green', 'price': 0.4}, {'name': 'grapes', 'colour': 'purple', 'price': 1.0}]** |
| **[fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp)** | **This creates a new dictionary with the given elements as the keys of the new dictionary** | **dict1 = {'a': 2, 'b': 5, 'c': 9}**  **print(dict1.fromkeys(‘def'))**  **\*\*\***  **{'d': None, 'e': None, 'f': None}** |
| **[get()](https://www.w3schools.com/python/ref_dictionary_get.asp)** | **This method returns the value from the specified key in the dictionary.** | **apple = {"colour": "red", "price": 0.15}**  **print(apple.get(‘colour'))**  **\*\*\***  **red** |
| **[items()](https://www.w3schools.com/python/ref_dictionary_items.asp)** | **This allows us to see the tuple pairs from within the dictionary in a list format.** | **apple = {"colour": "red", "price": 0.15}**  **print(apple.items())**  **\*\*\***  **dict\_items([('colour', 'red'), ('price', 0.15)])** |
| **[keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp)** | **This method prints the names of the dictionary keys** | **apple = {"colour": "red", "price": 0.15}**  **print(apple.keys())**  **\*\*\***  **dict\_keys(['colour', 'price'])** |
| **[pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp)** | **This method allows the user to remove a specified item from the dictionary by calling the key** | **apple = {"colour": "red", "price": 0.15}**  **apple.pop('price')**  **print(apple)**  **\*\*\***  **{'colour': 'red'}** |
| **[popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp)** | **This method removes the last item in a dictionary and will allow the user to print it as a tuple.** | **apple = {"colour": "red", "price": 0.15, 'sort': 'granny smith'}**  **a1 = apple.popitem()**  **print(a1)**  **\*\*\***  **('sort', 'granny smith')** |
| **[setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp)** | **This method can be used to set a default value for a specific key in a dictionary. If no value is entered, then the default will be entered.** | **apple = {"colour": "red", "price": 0.15}**  **apple.setdefault('sort', 'red gala')**  **print(apple)**  **\*\*\***  **{'colour': 'red', 'price': 0.15, 'sort': 'red gala'}** |
| **[update()](https://www.w3schools.com/python/ref_dictionary_update.asp)** | **This method allows the user to update the dictionary with the elements from another dictionary, inserting the values into the end of the dictionary.** | **exam\_results = {'English': 'A', 'Maths': 'A', 'Biology': 'A', 'German': 'B'}**  **new\_result = {'Modern Studies': 'A'}**  **exam\_results.update(new\_result)**  **print(exam\_results)**  **\*\*\***  **{'English': 'A', 'Maths': 'A', 'Biology': 'A', 'German': 'B', 'Modern Studies': 'A'}** |
| **[values()](https://www.w3schools.com/python/ref_dictionary_values.asp)** | **This will return a list of all the values within the dictionary.** | **exam\_results = {'English': 'A', 'Maths': 'A', 'Biology': 'A', 'German': 'B'}**  **print(exam\_results.values())**  **\*\*\***  **dict\_values(['A', 'A', 'A', 'B'])** |

|  |  |
| --- | --- |
| 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)** | **This method adds a single item into a set.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate.add('Salted Caramel')**  **print(chocolate)**  **\*\*\***  **{'Milk', 'White', 'Mint', 'Dark', 'Salted Caramel'}** |
| **[clear()](https://www.w3schools.com/python/ref_set_clear.asp)** | **This method will wipe all values/elements from the set.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate.clear()**  **print(chocolate)**  **\*\*\***  **set()** |
| **[copy()](https://www.w3schools.com/python/ref_set_copy.asp)** | **This method returns a duplicate copy of the set.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **choc2 = chocolate.copy()**  **print(choc2)**  **\*\*\***  **{'Dark', 'Mint', 'Milk', 'White'}** |
| **[difference()](https://www.w3schools.com/python/ref_set_difference.asp)** | **This method will compare two sets and return all values that are present in set A but not in set B.** | **set1 = {1, 44, 83, 19}**  **set2 = {7, 44, 36, 21}**  **set3 = set1.difference(set2)**  **print(set3)**  **\*\*\***  **{19, 1, 83}** |
| **[intersection()](https://www.w3schools.com/python/ref_set_intersection.asp)** | **SImilar to above, but this method will return all values present in both set A and set B.** | **set1 = {1, 44, 83, 19}**  **set2 = {7, 44, 36, 21}**  **set3 = set1.intersection(set2)**  **print(set3)**  **\*\*\***  **{44}** |
| **[issubset()](https://www.w3schools.com/python/ref_set_issubset.asp)** | **This method will return as True if all values in set A are also present in set B i.e set A is a subset of set B.** | **set1 = {1, 2, 3, 4}**  **set2 = {7, 6, 5, 4, 3, 2, 1}**  **set3 = set1.issubset(set2)**  **print(set3)**  **\*\*\***  **True** |
| **[issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp)** | **This method will return True if all items in set B are also present in set A. If not, Python will return False.** | **set1= {100, 90, 85, 70, 60, 50}**  **set2 = {50, 60, 70}**  **set3 = set1.issuperset(set2)**  **print(set3)**  **\*\*\***  **True** |
| **[pop()](https://www.w3schools.com/python/ref_set_pop.asp)** | **This method removes an element from the set at random.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate.pop()**  **print(chocolate)**  **\*\*\***  **{'Mint', 'Milk', 'White'}** |
| **[remove()](https://www.w3schools.com/python/ref_set_remove.asp)** | **This will remove an element from the set as specified.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate.remove('Mint')**  **print(chocolate)**  **\*\*\***  **{'Dark', 'Milk', 'White'}** |
| **[symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp)** | **This method returns a set of elements from both sets, excluding elements which appear in both, thus printing items which are unique to both lists.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **choc2 = {'Dark', 'Salted Caramel', 'White', 'Aero'}**  **choc3 = chocolate.symmetric\_difference(choc2)**  **print(choc3)**  **\*\***  **{'Milk', 'Aero', 'Mint', 'Salted Caramel'}** |
| **[union()](https://www.w3schools.com/python/ref_set_union.asp)** | **This method will return all items from both sets, essentially combining both sets into one.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate2 = {'Dark', 'Salted Caramel', 'White', 'Aero'}**  **chocolate3 = chocolate.union(chocolate2)**  **print(chocolate3)**  **\*\*\***  **{'Milk', 'White', 'Aero', 'Mint', 'Dark', 'Salted Caramel'}** |
| **[update()](https://www.w3schools.com/python/ref_set_update.asp)** | **This method will update set A by adding elements from set B. If elements are present in both sets, it will only be returned once in the updated set.** | **chocolate = {'Milk', 'Dark', 'White', 'Mint'}**  **chocolate2 = {'Dark', 'Salted Caramel', 'White', 'Aero'}**  **chocolate.update(chocolate2)**  **print(chocolate)**  **\*\*\***  **{'Milk', 'White', 'Aero', 'Mint', 'Dark', 'Salted Caramel'}** |

|  |  |
| --- | --- |
| 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)** | **This method will return the file content.** |  |
| **[readline()](https://www.w3schools.com/python/ref_file_readline.asp)** | **This method will return a line from the file, and it is possible to specify the size of the line to display. The default is one line only.** |  |
| **[readlines()](https://www.w3schools.com/python/ref_file_readlines.asp)** | **This will give an output of the file where each line of text is an item in a list.** |  |
| **[write()](https://www.w3schools.com/python/ref_file_write.asp)** | **This will allow you to write a string into a file, often adding this onto the end of the file. However, using only the ‘w’ method when dealing with a file will first wipe the file contents before allowing the user to insert the new string.** |  |
| **[writelines()](https://www.w3schools.com/python/ref_file_writelines.asp)** | **This method allows us to add (or write) a set of strings in the format of a list into a file. Again, this will either be at the end of an existing file or the target file will be wiped before inserting the new list into the file.** |  |