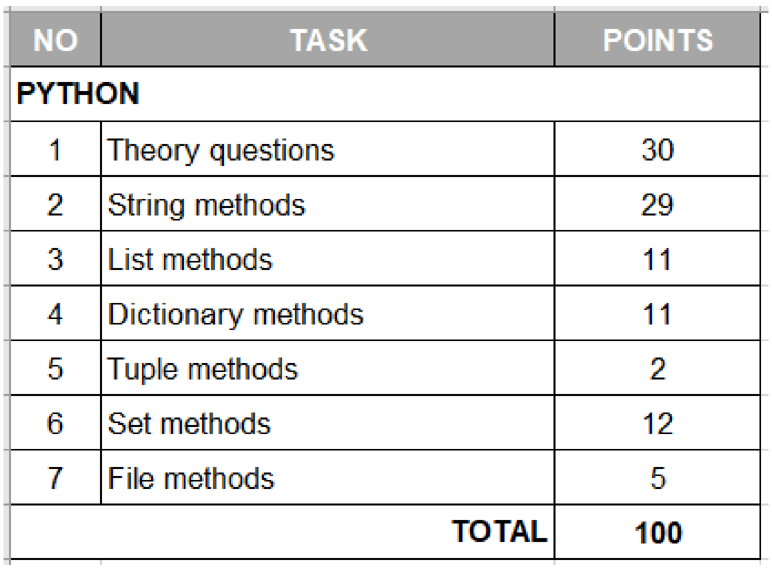
Deeqa Mohamed

THEORY QUESTIONS ASSIGNMENT

Python based theory



NO	TASK	POINTS
PYTHON		
1	Theory questions	30
2	String methods	29
3	List methods	11
4	Dictionary methods	11
5	Tuple methods	2
6	Set methods	12
7	File methods	5
	TOTAL	100

1. Python theory questions	**30 points**

1. What is Python and what are its main features?
    a. Python is an interpreted, object oriented programming language. The benefits of using python include: web and game development, data analysing,AI and machine learning, and data visualisation. Python also upports extensive libraries and is easy to use.

2. Discuss the difference between Python 2 and Python 3
    a. The main difference between Python 2 and Python 3 is that Python 2 uses an older syntax for the print function. It treated print as a statement rather than a function. Python 3 is a revised and renewed version of Python 2.

3. What is PEP 8?
    a. PEP stands for Python Enhancement Proposal. PEP 8 is a guideline documentation on how to write python code and the best python practices.

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

   a. Program in computer science refers to a set of specified orders and instructions that a computer is able to execute. Those orders are translated into source codes.

5. In computing / computer science what is a process?
   a. Process is when a program is being executed. Usually one process is associated with one program.

6. In computing / computer science what is cache?
   a. Cache is a chip-based feature of the computer that has temporary memory. It's useful when you want to access some information quickly as it is faster than accessing information from the main hard drive.

7. In computing / computer science what is a thread and what do we mean by multithreading?
   a. A thread is within a program and has a single sequential flow of control. Multi-thread is when more than one thread can run in the program and perform different tasks.

8. In computing / computer science what is concurrency and parallelism and what are the differences?
   a. Concurrency is when more than one task can start, run, and complete whilst overlapping in time. Parallelism is when two or more tasks run but only at the same exact time.

9. What is GIL in Python and how does it work?
   a. GIL stands for global interpreter lock and is used to sync the execution of threads to allow only one thread to be executed at a time.

10. What do these software development principles mean: DRY, KISS, BDUF
    a. DRY: Don't repeat yourself
    b. KISS: Keep It Simple, Stupid
    c. BDUF: Big Design Up Front

11. What is a Garbage Collector in Python and how does it work?
    a. Garbage collector is used in python to free up memory space. It destroys unused objects and re-used that memory slot for new objects.

12. How is memory managed in Python?
    a. There are two parts to memory: heap memory and stack memory. Heap memory stores all value objects and stack memory stores references and methods. Stack memory works by using continuous blocks of memory. When a function is running on python, the size of memory gets allocated on the stack. The compiler will know how much memory to allocate to the stack memory. Heap memory works by having the programmer allocate and de-allocate memory space. It's a pile of memory space available to programmers to use at their discretion.

13. What is a Python module?

   a. A python module is a file with '.py' extension. It contains python code, definition, and statements. It's used to larger programs into smaller ones.

14. What is docstring in Python?
    a. Docstrings are used in a python file alongside a function, method, class, or module. It's used to help programmers understand the code and what it does. Python ignores docstrings when executing a program. However, docstrings can be printed by using the __doc__.

15. What is pickling and unpickling in Python? Example usage.
    a. Pickling is a module used for flattening python object structures. This is the process of converting an object in memory to a byte stream that can be either stored on a disk or sent over a network. Unpickling is the process of converting a byte stream back to a python object.

16. What are the tools that help to find bugs or perform static analysis?
    a. Pychecker and Pylint perform static analysis to find bugs in python.

17. How are arguments passed in Python by value or by reference? Give an example.
    a. All arguments in python are passed by reference. An example: of that would be:

```
student={'Archana':28,'krishna':25,'Ramesh':32,'vineeth':25
}
def test(student):
   new={'alok':30,'Nevadan':28}
   student.update(new)
   print("Inside the function",student)
   return
test(student)
print("outside the function:",student)
```

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

    a. Dictionary comprehension is transforming one dictionary to another and filters or alters the content.
       i.

```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
   double_dict1 = {k:v*2 for (k,v) in dict1.items()}
   print(double_dict1)
```

    b. List comprehension transforms by altering or filtering a list and outputting a new list.
       i.

```
values = [2, 4, 6, 8, 10]
   doubled_values = [x*2 for x in values]
   print(doubled_values)
```

19. What is namespace in Python?
    a. Namespace refers to a collection of defined symbolic names. When python executes a program, it creates namespaces and deletes them when they are no longer required. Examples of namespaces are:
       i. Built-in
       ii. Local
       iii. Global

20. What is pass in Python?
    a. Pass is a null statement and is used as a placeholder for future code.

21. What is unit test in Python?
    a. Unit testing is when sections of a program undergo testing to see if it's a feasible code.

22. In Python what is slicing?
    a. Slicing is used in sequences and returns a sliced version of the sequence.

23. What is a negative index in Python?
    a. A negative index allows you to access the list from the end. So -1 would return the last value of the array.

24. How can the ternary operators be used in python? Give an example.
    a. Is a conditional expression used in python to check if a statement is true or false.
        i.
```
age = 48
discount = True if age >= 65 else False
print(discount)
```

25. What does this mean: *args, **kwargs? And why would we use it?
    a. *args are non-keyword arguments and *kwargs are keyword arguments. They are used when we are not sure of the number of arguments to pass in the function.

26. How are range and xrange different from one another?
    a. Xrange() is used in python 2 and range() is used in python 3. They are both similar in functionality but the difference is that range() returns a python list and xrange() returns an xrange object.

27. What is Flask and what can we use it for?
    a. Is a web framework that allows you to build web applications. Flask provides you with libraries, tools, and technologies.

28. What are clustered and non-clustered index in a relational database?
    a. Clustered index performs sorting only in one table. A clustered index is an index which defines the physical order in which table records are stored in a database. For non-clustered index, the data and index are stored separately. We can have many non-clustered indexes in one place.

29. What is a 'deadlock' a relational database?
    a. Is when two or more transactions are not able to progress because they rely on one another to release a lock.

Deeqa Mohamed

30. What is a 'livelock' a relational database?

   a. Livelock is similar to deadlock but instead of being in a waiting process state, it keeps having the request for the lock constantly denied because many shared locks keep overlapping.

2. Python string methods: describe each method and provide an example	**29 points**

METHOD	DESCRIPTION	EXAMPLE

capitalize()	capitalize() returns a copy of the original string and changes the first character to uppercase and keeps the remaining characters as lowercase.	`var = 'hello,world'`
		`x = var.capitalize()`
		`print(x)`

casefold()	casefold() returns a string with all lowercases	`var = 'Hello, World'`
		`x = var.casefold()`
		`print(x)`
		
		hello, world

center()	center() centre aligns the string by the number of characters stated in the brackets	`var = 'Hello, World'`
		`x = var.center(50)`
		`print(x)`

count()	Count() will count the number of elements specified in the bracket.	`var = ['Sweets', 'Chocolate', 'Cake']`
		`x = var.count('Sweets')`
		`print(x)`

endswith()	endswith() returns true or false if the last value in the string is the same as the specified value in the brackets	`var = 'Python is fun'`
		`x = var`
		`print(var.endswith('fun'))`

find()	find() finds the index of the first occurrence of the substring. If it returns -1, then the specified value is not found	`var = 'Python is fun'`
		`x = var.find('fun')`
		`print(x)`

format()	**format() allows you to do variable and data substitution in the string variable. A placeholder,{}, is required for this to work.**	`var = 'My name is {}'.format('Deeqa')`
		`print(var)`

index()	**index() is the same as find() in the sense that it returns the first occurrence of the substring. However, if the specified value is -1, index() raises an error.**	`var = 'Python is fun'`
		`x = var.index('Python')`
		`print(x)`

isalnum()	**isalnum() will return true if all the characters in the variable are alphanumeric. Returns false if not.**	`var = 'Pythonisfun123'`
		`x = var.isalnum()`
		`print(x)`

isalpha()	**Will return true if all the characters in the variable are alphabets. Returns false if not.**	`var = 'Pythonisfun'`
		`x = var.isalpha()`
		`print(x)`

isdigit()	**Will return true if all the characters in the variable are digits, including exponentials. Returns false if not.**	`var = '123'`
		`x = var.isdigit()`
		`print(x)`

islower()	**Will return true if all the characters in the variable are lowercase. Returns false if not.**	`var = 'hello my name is deeqa'`
		`x = var.islower()`
		`print(x)`

isnumeric()	**Wil return true if all the characters in the variable are numerical. Returns false if not.**	`var = '12345'`
		`x = var.isnumeric()`
		`print(x)`

isspace()	**Checks if all the characters in the variable are whitespaces. Returns true if the variable is all whitespaces. Otherwise it returns false.**	`var = '    '`
		`x = var.isspace()`
		`print(x)`

istitle()	**Checks if all the first letters of the words in the string are uppercase and the rest of the words are lowercase. Returns true if it satisfies the check and returns false if not.**	`var = 'Hello World'`
		`x = var.istitle()`
		`print(x)`

isupper()	**Checks if all the characters in the variable are uppercase. Will return true if all characters are upper and false if it does not satisfy the requirements.**	`var = 'HELLO WORLD'`
		`x = var.isupper()`
		`print(x)`

join()	**Joins all items in a tupple or dictionary and by using a string as a separator.**	`myTuple = ("Sam", "Ellie", "Hannah")`
		`x = "#".join(myTuple)`
		`print(x)`

lower()	**Will return a string in a variable as all lower cases.**	`var = 'HELLO WORLD'`
		`x = var.lower()`
		`print(x)`

lstrip()	**It removes any prior whitespaces left of the string.**	`var = '      hello'`
		`x = var.lstrip()`
		`print(x,'World')`

replace()	**Will replace a world in a string with another by specifying it in the brackets of replace.**	`var = 'Hello my name is Ellie'`
		`x =`
		`var.replace('Ellie', 'Deeqa')`
		`print(x)`

rsplit()	**Splits a string into a list and the separator is specified in the brackets**	`var = "Sweets, Chocolate, Cake"`
		`x = var.rsplit(", ")`
		`print(x)`

rstrip()	**Removes all whitespaces right of the string.**	`var = 'hello      '`
		`x = var.rstrip()`
		`print(x,'World')`

split()	**Will split a string into a list.**	`var = "Sweets, Chocolate, Cake"`
		`x = var.split()`
		`print(x`

splitlines()	**Will split a string into a list and separate the string at the line breaks into single items.**	`var = "Python is so fun\nyou learn so much"`
		`x = var.splitlines()`
		`print(x)`

startswith()	Startswith() returns true or false if the first value in the string is the same as the specified value in the brackets.	`var = 'Python is fun'` `x = var` `print(var.startswith('Python'))`
strip()	Removes all whitespaces right and left of the string.	`var = '     hello      '` `x = var.strip()` `print(x,'World')`
swapcase()	Will return the string with swapped cases. For example, if the string had uppercases, it will change it to lowercase and vice versa.	`var = 'HeLLO woRld'` `x = var.swapcase()` `print(x)`
title()	Will return all the first characters of each word in the string as an uppercase.	`var = 'hello world'` `x = var.title()` `print(x)`
upper()	Will return all the characters in the variable as uppercase.	`var = 'hello world'` `x = var.upper()` `print(x)`

3. Python list methods: describe each method and provide an example	**11 points**

Method	Description	Example
append()	Adds elements to the end of the list.	`var = ['Sweets', 'Chocolate', 'Cake']` `x = var.append('apple')` `print(var)`
clear()	Will remove all the elements in a list.	`var = ['Sweets', 'Chocolate', 'Cake']` `x = var.clear()` `print(var)`

copy()	**Makes a copy of the original list and returns it.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x = var.copy()`
		`print(x)`

count()	**Will count the number of times the specified element is in the list.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x =`
		`var.count('Sweets`
		`')`
		`print(x)`

extend()	**Will add all the elements of a list, tupple, dictionary to the end of a list**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`var1 = ['1','2']`
		`var.extend(var1)`
		`print(var)`

index()	**It will return the position of the specified element**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x =`
		`var.index('Chocol`
		`ate')`
		`print(x)`

insert()	**Will add an element to a list given the specific position.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x =`
		`var.insert(2,'App`
		`le')`
		`print(var)`

pop()	**Deletes an element from the list at the given position.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x = var.pop(1)`
		`print(var)`

remove()	**Deletes the specified element passed as an argument in the list.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x =`
		`var.remove('Cake'`
		`)`
		`print(var)`

reverse()	**Changes the position of the elements in the list in reverse order.**	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x = var.reverse()`
		`print(var)`

sort()	Orders the list in alphabetical order or ascending order.	`var = ['Sweets',`
		`'Chocolate',`
		`'Cake']`
		`x = var.sort()`
		`print(var)`

4. Python tuple methods: describe each method and provide an example	**2 points**

Method	Description	Example

count()	Counts the number of times an element, passed as an argument is in the tuple.	`var =`
		`('3','6','7','1',`
		`'6')`
		`x =`
		`var.count('6')`
		`print(x)`

index()	Will return the index of the specified element in the tuple.	`var =`
		`('3','6','7','1',`
		`'6')`
		`x =`
		`var.index('7')`
		`print(x)`

5. Python dictionary methods: describe each method and provide an example	**11 points**

Method	Description	Example

clear()	Clears all the elements in the dictionary	`var = {0:`
		`'zero',1:`
		`'one',2: 'two'}`
		`x = var.clear()`
		`print(var)`

copy()	Makes a copy of the dictionary and returns it.	`var = {0:`
		`'zero',1:`
		`'one',2: 'two'}`

		`x = var.copy()` `print(x)`
fromkeys()	**Creates a new dictionary using the sequence of elements from the orginal dictionary. You can also call on new values.**	`var = {0: 'zero',1: 'one',2: 'two'}` `value = 'number'` `x = var.fromkeys(var, value)` `print(x)`
get()	**Will return the value of the specified key given in the argument.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.get(1)` `print(x)`
items()	**Will return a display of the dictionary with each key and value.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.items()` `print(x)`
keys()	**Will return a list of the key elements in the dicionary**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.keys()` `print(x)`
pop()	**Pop() will remove the specified element (key and value) from the dictionary and return it.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.pop(1)` `print(var)`
popitem()	**Removes the last item that was added to the dictionary.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.popitem()` `print(var)`
setdefault()	**Will return the value of the specified key.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.setdefault(0)` `print(x)`
update()	**Adds a specified item into the dictionary.**	`var = {0: 'zero',1: 'one',2: 'two'}` `x = var.update({3:'three'})` `print(var)`
values()	**Will return all the values of the dictionary without their respective keys.**	`var = {0: 'zero',1: 'one',2: 'two'}`

		`x = var.values()`
		`print(x)`

6. Python set methods: **describe each method and provide an example**	**12 points**

Method	Description	Example

add()	Will add the element passed in the argument onto the set.	`var = {"chocolate", "sweets", "cake"}`
		`x = var.add("orange")`
		`print(var)`

clear()	Will clear all the elements in the set.	`var = {"chocolate", "sweets", "cake"}`
		`x = var.clear()`
		`print(var)`

copy()	Copy() creates an identical copy of the set to it's assigned variable.	`var = {"chocolate", "sweets", "cake"}`
		`x = var.copy()`
		`print(x)`

difference()	Returns a new list that has the unique elements in both lists. An element that is not in both lists will not be returned	`var = {"chocolate", "sweets", "cake"}`
		`var1 ={"orange","pear","cake"}`
		`x = var.difference(var1)`
		`print(x)`

intersection()	Unlike difference(), intersection() will return the elements that are the same in both lists and ignores the unique elements.	`var = {"chocolate", "sweets", "cake"}`
		`var1 ={"orange","pear","cake"}`
		`x = var.intersection(var1)`
		`print(x)`

issubset()	Returns true if all items in the first set are present in the second set.	`var = {"chocolate", "sweets", "cake"}`
		`var1`

		={"chocolate", "sweets","orange","pear","cake"}
		x = var.issubset(var1)
		print(x)

issuperset()	**Returns true if all the items in the second set are also in the first.**	var1 ={"chocolate", "sweets","orange","pear","cake"}
		var = {"chocolate", "sweets", "cake"}
		x = var1.issuperset(var)
		print(x)

pop()	**Will randomly delete an item from the set.**	var = {"chocolate", "sweets", "cake"}
		x = var.pop()
		print(var)

remove()	**This will remove the specified item in the set that is passed as an argument.**	var = {"chocolate", "sweets", "cake"}
		x = var.remove('sweets')
		print(var)

symmetric_difference()	**This will return a new set the contains all the unique items in both sets. If there's an item present in both sets, it will not return it.**	var = {"chocolate", "sweets", "cake"}
		var1 ={"orange","pear",'cake'}
		x = var.symmetric_difference(var1)
		print(x)

union()	**This will return all the items in both sets including those that are present in both sets. However, if there's an item present in both sets, it will only return it once in the new set.**	var = {"chocolate", "sweets", "cake"}
		var1 ={"orange","pear",'cake'}
		x = var.union(var1)
		print(x)

update()	**Update() will update the first set by adding items from the new set to it.**	var = {"chocolate", "sweets", "cake"}
		var1 ={"orange","pear",'cake'}

		`x = var.update(var1) print(var)`

7. **Python file methods: describe each method and provide an example**	**5 points**

Method	Description	Example
read()	Will read and output the content of the file.	`file = open("poem.txt", "r") print(file.read())`
readline()	Will read and output the first line of the file.	`file = open("poem.txt", "r") print(file.readline())`
readlines()	Will read and output the each line of the file as a list.	`file = open("poem.txt", "r") print(file.readlines())`
write()	Write() allows us to write in a file. The desired additional text is written in the brackets.	`file = open("poem.txt", "a") file.write('However, I like reading poesm')`
writelines()	Writes additional text to the file. This is done by writing the content in list format.	`file = open("poem.txt", "a") file.writelines(['However, I like reading poesm','That is it for now!'])`