THEORY QUESTIONS ASSIGNMENT

Python based theory

**30 points**

**1. Python theory questions**

1. Python is an object-oriented multi-paradigm programming language.

Main features: 1. Easy to understand and code. 2. High-level language and portable 3. Free and open source

1. In python3, default storing of string is Unicode, whereas in python2 Unicode should be specified with -u

Python3 offers iterable Range() function, while python2 has xrange() function.

In python3, exceptions are written in parenthesis whereas in python2, exceptions are written in notations. The syntax in python3 is much easier than python2.

1. PEP8 is a python style guide, a tool used to check python code against some style guides in PEP8.
2. Program is a set of ordered instructions performed to solve a task
3. Process is an instance of a program executed by one or more threads.
4. Cache: A hardware and software component that has data storing technique where data is stored so that future request for that data can be done faster.
5. Thread: Smallest set of instructions designed executed and managed independent of parent process by scheduler.

Multi-threading: Two or more threads can be executed independently using same process resources.

1. Concurrency: Making progress on more than one task but not necessarily simultaneously.

Parallelism: Splitting task to smaller subtasks and executing on multiple CPU at exact same time.

Difference b/w concurrency and parallelism:

Concurrency support progress on multiple tasks at same time while Parallelism supports execution of multiple tasks at same time.

1. GIL : Global interpreter lock allows only one thread to hold control of python interpreter at a time so that only one thread can be executed at once. This prevents race conditions and prevents thread safety

Working: GIL uses concept called ticks and checks every thread for 100 ticks.

1. DRY: Don’t Repeat Yourself ---A software principle aimed to eliminate repeated software patterns to avoid redundancy.

KISS: Keep It Simple, Stupid---Notifies that all software, application designs should be as simple as possible.

BDUF: Big Design Up Front—States that program's design is to be completed before that program's implementation started

1. Garbage Collector: Automatic memory management—When an object has zero references to it, then memory allocated from the object will be deallocated and object will be deleted.
2. Memory management: Memory management in python involves private heap containing python object and memory manager allocates heap space to objects.
3. Module: Python object containing code that defines functions and classes. Modules helps to reduce complexity of code when imported and the elements inside the module can be called and simply used.
4. Docstring: Documentation string that provides way for associating python documentation to modules, functions and classes.
5. Pickling: A process in which a python object is converted to byte stream

Unpickling: Binary file or bytes like object is converted to python object.

Pickled file is bit hard to read and improves security.

Example usage:

**Pickling:**

import pickledef pickle\_data(): data = { 'name': 'Sravya', 'profession': 'Student', 'country': 'UK' } filename = 'PersonalInfo' outfile = open(filename, 'wb') pickle.dump(data, outfile)

outfile.close()

pickle\_data()

**Unpickling**

infile = open('PersonalInfo', 'rb')

new\_dict = pickle.load(infile)

infile.close()

1. Pylint-It perform error detection and refactoring by detecting duplicated code, Mypy--checking code and find common bugs, Pyflakes- has limited set of errors and faster than Pylint.
2. Immutable objects like integers, strings etc are passed by value and mutable objects are passed by reference.

Ex: Call by value call by reference

y=10 y=10

def fun(x): def fun(x):

return print(x) return print(x)

fun(y) fun(id(y))

1. Dictionary comprehension is method of transforming one dictionary to other

Ex: dict={key:value ( for (key,value) in range dict.items())

List comprehension is method of transforming one list to other

Ex: list1= [ expression for value in range list if condition==true]

1. Python implements scope for variables to get resolved. Every module, function, method can create namespace for identifying variable and when variable is no longer used, namespace is dropped.
2. Pass: Null operation when executes. It is used when a statement is required syntactically but you do not want any command or code to execute.
3. Unit test: A framework in python when testcases are collected and executed in a single file all together.
4. Slicing: Breaking an object and accessing parts of sequence. Slicing in python finally return a sequence of selected

Function: Slice()

1. Negative indexing accesses the items from end of list counting from backwards.
2. Ternary operators are conditional expressions having operators that evaluate something based on a condition being true or false

Ex: a, b = 10, 20

min = a > b and a or b

print(min)

1. \*args: \*args in function definitions in python is used to pass a variable number of non-keyworded arguments to a function

\*\*Kwargs: \*\*kwargs in function definitions in python is used to pass a keyworded, variable-length argument list.

Use: We can pass any number of arguments during function call and while writing function we don’t need to decide number of arguments or even type of arguments in case of \*args.

26. xrange() method returns results in an object while range() method results all the numbers in one go

27. Flask is a web framework that provides libraries to build lightweight web applications in python.

28. Clustered index gives idea on how data is sorted in database. A table can have only one clustered index while non-clustered index stores data at once location and indices at another location. Non-clustered index has pointers to point to data location. A table can have more than one non-clustered index.

29. Dead-lock: It is a condition when two or database tasks are waiting for each other to use mutually sharable resources and none of the task is willing to give the resources.

30. Live-lock: A **Livelock** is a situation when two or more processes keep on changing their status, according to changes in other which further prevents them from completing the tasks.

**29 points**

**2. Python string methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | method returns a string where the first character is upper case. | txt = "hello, and welcome"  x = txt.capitalize()  print (x)  o/p: Hello, and welcome. |
| **casefold()** | method returns a string where all the characters are lower case | txt = "HEllo, And WelcomE"  x = txt.casefold()  print (x)  o/p: hello, and welcome |
| **center()** | method will center align the string, using a specified character | txt = "banana"  x = txt.center(10,' i')  print(x)  o/p:iibananii |
| **count()** | method returns the number of elements with the specified value. | fruits = ['a','b','c','a']  x = print(fruits.count('a'))  o/p:2 |
| **endswith()** | method returns True if the string ends with the specified value, otherwise False | txt = "Hello, welcome"  x = txt.endswith("e")  print(x)  o/p: true |
| **find()** | method finds the first occurrence of the specified value else returns -1 if not found | txt = "Hello, welcome"  x = txt.find("w")  print(x)  o/p:7 |
| **format()** | method formats the specified value(s) and insert them inside the string's placeholder denoted by {} | txt = "Hello, welcome"  print("{} my friend".format(txt))  o/p: Hello, welcome my friend |
| **index()** | method returns the position at the first occurrence of the specified value  used for tuples, lists | fruits = ['a', 'b', 'c']  x = print(fruits.index('c'))  o/p: 2 |
| **isalnum()** | method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9). | txt = "Pinky12"  x = txt.isalnum()  print(x)  o/p: true |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | method returns True if all the characters are alphabetic (a-z),  else return false | txt = "Pinky1"  x = txt.isalpha()  print(x)  o/p:false |
| **isdigit()** | method returns True if all the characters are digits, otherwise False. | txt = "P1123"  x = txt.isdigit()  print(x)  o/p: false |
| **islower()** | method returns True if all the characters are in lower case, otherwise False.  Return true for lowercase,numerics combination | txt = "hey pinkY"  x = txt.islower()  print(x)  o/p: false |
| **isnumeric()** | method returns True if all the characters are numeric (0-9), otherwise False.  Decimals, negative integers result false | txt = "1.5"  x = txt.isnumeric()  print(x)  o/p: false |
| **isspace()** | method returns True if **all** the characters in a string are just blankspaces, otherwise False | txt = " k "  x = txt.isspace()  print(x)  o/p: false |
| **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, This Is Pinky!!!"  x = txt.istitle()  print(x)  o/p:true |
| **isupper()** | method returns True if all the characters are in upper case | txt = "HELLo!!!"  x = txt.isupper()  print(x)  o/p: false |
| **join()** | method takes all items in list, tuple or any object of items and joins them into one string. These strings are joined with mentioned joining character. | myTuple = ["Sravya", "Betina", "Nice"]  x = "@".join(myTuple)  print(x)  o/p:Sravya@Betina@Nice |
| **lower()** | method returns a string where all characters are lower case while symbols and numbers can be ignored | txt = "Hello de@r PinKY"  x = txt.lower()  print(x)  o/p: hello de@r pinky |
| **lstrip()** | method removes any left leading characters. | txt = ",,,,,ssaaww.....ball....wwaass,,"  x = txt.lstrip(",.asw")  print(x)  o/p: ball....wwaass,, |
| **replace()** | method replaces a specified phrase with another specified phrase | txt = "I like bananas"  x = txt.replace("like", "hate")  print(x)  o/p: I hate bananas |
| **rsplit()** | method splits a string into a list, starting from the right based on given split character | txt = "I am Sravya"  x = txt.rsplit(" ")  print(x)  o/p: ['I', 'am', 'Sravya'] |
| **rstrip()** | method removes any right trailing characters. | txt = ",,,,,ssaaww.....ball....wwaass,,"  x = txt.rstrip(",.asw")  print(x)  o/p: ,,,,,ssaaww.....ball |
| **split()** | method splits a string into a list basing on character specified. | txt = "I am Sravya"  x = txt.split("a")  print(x)  o/p: ['I ', 'm Sr', 'vy', ''] |
| **splitlines()** | method splits a string into a list and the splitting is done at line breaks | txt = "Thank \nyou for\n the music\nWelcome"  x = txt.splitlines()  print(x)  o/p: ['Thank ', 'you for', ' the music', 'Welcome'] |
| **startswith()** | method returns True if the string starts with the specified value | txt = "Hello, welcome Sravya"  x = txt.startswith("Sravya")  print(x)  o/p: False |
| **strip()** | method removes any leading and trailing characters | txt = "Hey,Sravya,Hey"  x = txt.strip(",Hey")  print(x)  o/p:Sravya |
| **swapcase()** | method returns a string where all the upper case letters are lower case and vice versa | txt = "Hey,Sravya,Hey"  x = txt.swapcase()  print(x)  o/p: hEY,sRAVYA,hEY |

|  |  |  |
| --- | --- | --- |
| **title()** | method returns a string where the first character in every word is upper case | txt = "Welcome,dear sravya"  x = txt.title()  print(x)  Welcome,Dear Sravya |
| **upper()** | method returns a string where all characters are in upper case | txt = "Welcome,dear sravya"  x = txt.upper()  print(x)  o/p:  WELCOME, DEAR SRAVYA |

**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) | method appends an element to the end of the list. | fruits = ['apple', 'banana', 'cherry']  fruits.append("orange")  print(fruits)  o/p: ['apple', 'banana', 'cherry', 'orange'] |
| [**clear()**](https://www.w3schools.com/python/ref_list_clear.asp) | method removes all the elements from a list.  So, this results empty list | fruits = ['apple', 'banana', 'cherry']  fruits.clear()  print(fruits)  o/p: [] |
| [**copy()**](https://www.w3schools.com/python/ref_list_copy.asp) | method returns a copy of the mentioned list | fruits = ['apple', 'banana', 'cherry']  fruit1=fruits.copy()  print(fruit1)  o/p: ['apple', 'banana', 'cherry'] |
| [**count()**](https://www.w3schools.com/python/ref_list_count.asp) | method returns the number of elements with the specified value | fruits = ['apple', 'banana', 'cherry', 'cherry']  print(fruits.count('cherry'))  o/p:2 |
| [**extend()**](https://www.w3schools.com/python/ref_list_extend.asp) | method adds the specified list elements (or any iterable) to the end of the current list | fruits1 = ['apple', 'banana']  fruits2=['cherry', 'grapes']  fruits1.extend(fruits2)  print(fruits1)  o/p: ['apple', 'banana', 'cherry', 'grapes'] |
| [**index()**](https://www.w3schools.com/python/ref_list_index.asp) | method returns the position at the first occurrence of the specified element. | fruits1 = ['apple', 'banana']  print(fruits1.index('banana'))  o/p: 1 |
| [**insert()**](https://www.w3schools.com/python/ref_list_insert.asp) | method inserts the specified value at the specified position | fruits1 = ['apple', 'banana']  fruits1.insert(0, "orange")  print(fruits1)  o/p: ['orange', 'apple', 'banana'] |
| [**pop()**](https://www.w3schools.com/python/ref_list_pop.asp) | method removes the element at the specified position. | fruits1 = ['apple', 'banana']  fruits1.pop(0)  print(fruits1)  o/p: ['banana'] |
| [**remove()**](https://www.w3schools.com/python/ref_list_remove.asp) | method removes the first occurrence of the element with the specified value | fruits1 = ['apple', 'banana', 'apple']  fruits1.remove('apple')  print(fruits1)  o/p: ['banana', 'apple'] |
| [**reverse()**](https://www.w3schools.com/python/ref_list_reverse.asp) | method reverses the sorting order of the elements in the list. | fruits1 = ['apple', 'banana', 'grapes']  fruits1.reverse()  print(fruits1)  o/p: ['grapes', 'banana', 'apple'] |
| [**sort()**](https://www.w3schools.com/python/ref_list_sort.asp) | method sorts the list by default in ascending order | fruits1 = ['apple', 'grapes','banana', 'apricot']  fruits1.sort()  print(fruits1)  o/p: ['apple', 'apricot', 'banana', 'grapes'] |

**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) | method returns the number of times a specified value appears in the tuple | fruits1 = ('apple', 'grapes','banana', 'apricot')  print(fruits1.count('apple'))  o/p:1 |
| [**index()**](https://www.w3schools.com/python/ref_tuple_index.asp) | method finds the first occurrence of the specified value and raise exception when not found | fruits1 = ('apple', 'grapes','banana')  print(fruits1.index('apricot'))  o/p: Traceback (most recent call last):  File "main.py", line 2, in <module>  print(fruits1.index('apricot'))  ValueError: tuple.index(x): x not in tuple |

**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) | method removes all the elements from a dictionary | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  bio.clear()  print(bio)  o/p: {} |
| [**copy()**](https://www.w3schools.com/python/ref_dictionary_copy.asp) | method returns a copy of the mentioned dictionary | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  bio1=bio.copy()  print(bio)  print(bio1)  o/p: {'age': 23, 'name': 'Sravya', 'weight': 59}  {'age': 23, 'name': 'Sravya', 'weight': 59} |
| [**fromkeys()**](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | method returns a dictionary with the specified keys and the specified value | x = ('key1', 'key2', 'key3')  y = (0,1)  thisdict = dict.fromkeys(x, y)  print(thisdict)  o/p: {'key1': (0, 1), 'key3': (0, 1), 'key2': (0, 1)} |
| [**get()**](https://www.w3schools.com/python/ref_dictionary_get.asp) | method returns the value of the element with the given key | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  value=bio.get('name')  print(value)  o/p: Sravya |
| [**items()**](https://www.w3schools.com/python/ref_dictionary_items.asp) | method returns an object with each key-value pairs as tuples in a list | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  value=bio.items()  print(value)  o/p: dict\_items([('name', 'Sravya'), ('weight', 59), ('age', 23)]) |
| [**keys()**](https://www.w3schools.com/python/ref_dictionary_keys.asp) | method returns an object containing keys of dictionary as a list | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  value=bio.keys()  print(value)  o/p: dict\_keys(['name', 'weight', 'age']) |
| [**pop()**](https://www.w3schools.com/python/ref_dictionary_pop.asp) | method removes the specified item from the dictionary. | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  bio.pop('name')  print(bio)  o/p: {'age': 23, 'weight': 59} |
| [**popitem()**](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | method removes the item that was last inserted into the dictionary | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  bio.popitem()  print(bio)  o/p: {'name': 'Sravya', 'age': 23} |

|  |  |  |
| --- | --- | --- |
| [**setdefault()**](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | method returns the value of the item with the specified key and if doesn’t exist, it sets key with specified value. | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  print(bio.setdefault("age"))  bio.setdefault("colour",'brown')  print(bio)  o/p: {'weight': 59, 'name': 'Sravya', 'colour': 'brown', 'age': 23} |
| [**update()**](https://www.w3schools.com/python/ref_dictionary_update.asp) | method inserts the specified items to the dictionary, items can be a dictionary, or an iterable object with key value pairs | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  bio.update({"color":"white"})  print(bio)  o/p: {'weight': 59, 'color': 'white', 'name': 'Sravya', 'age': 23} |
| [**values()**](https://www.w3schools.com/python/ref_dictionary_values.asp) | method returns an object having values of dictionary as a list. | bio = {  "name": "Sravya",  "age": 23,  "weight": 59  }  print(bio.values())  o/p: dict\_values([23, 'Sravya', 59]) |

**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) | method adds an element to the set if element doesn’t exist, if exist doesn’t add the element | fruits = {"apple", "banana", "cherry"}  fruits.add("apricot")  print(fruits)  o/p: {'banana', 'apple', 'apricot', 'cherry'} |
| [**clear()**](https://www.w3schools.com/python/ref_set_clear.asp) | method removes all elements in a set | fruits = {"apple", "banana", "cherry"}  fruits.clear()  print(fruits)  o/p: set() |
| [**copy()**](https://www.w3schools.com/python/ref_set_copy.asp) | method copies the set | fruits = {"apple", "banana", "cherry"}  fruits1=fruits.copy()  print(fruits1)  print(fruits)  o/p: {'apple', 'banana', 'cherry'}  {'apple', 'banana', 'cherry'} |
| [**difference()**](https://www.w3schools.com/python/ref_set_difference.asp) | method returns a set that contains the difference between two sets. | fruits = {"apple", "banana", "cherry"}  fruits2={"apple", "apricot"}  fruits3=fruits-fruits2  print(fruits3)  o/p: {'cherry', 'banana'} |
| [**intersection()**](https://www.w3schools.com/python/ref_set_intersection.asp) | method returns a set that contains the similarity between two or more sets | fruits={"banana","apple"}  fruits1 = {"apple", "banana", "cherry"}  fruits2={"apple", "apricot"}  fruits3=fruits.intersection(fruits2)  print(fruits3)  o/p: {'apple'} |
| [**issubset()**](https://www.w3schools.com/python/ref_set_issubset.asp) | method returns True if all items in the set exists in the specified set. | fruits={"banana","apple"}  fruits1 = {"apple", "banana", "cherry"}  print(fruits.issubset(fruits1))  o/p:true |
| [**issuperset()**](https://www.w3schools.com/python/ref_set_issuperset.asp) | method returns True if all items in the specified set thought as subset exists in other set defined as superset. | fruits={"banana","apple"}  fruits1 = {"apple", "banana", "cherry"}  print(fruits1.issuperset(fruits))  o/p:true |
| [**pop()**](https://www.w3schools.com/python/ref_set_pop.asp) | method removes a random item from the set | fruits={"banana","apple"}  fruits.pop()  print(fruits)  o/p: {'apple'} |
| [**remove()**](https://www.w3schools.com/python/ref_set_remove.asp) | method removes the specified element from the set | fruits={"banana","apple"}  fruits.remove("apple")  print(fruits)  o/p: {'banana'} |
| [**symmetric\_differ**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp)[**ence()**](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | method returns a set that contains all items from two sets, but not the items that are common in both sets. | fruits={"banana","apple"}  fruits1={"apple", "apricot"}  print(fruits.symmetric\_difference(fruits1))  o/p: {'banana', 'apricot'} |
| [**union()**](https://www.w3schools.com/python/ref_set_union.asp) | method returns a set that contains all items from the original set, and all items from the specified set | fruits={"banana","apple"}  fruits1={"apple", "apricot"}  print(fruits.union(fruits1))  o/p: {'banana', 'apple', 'apricot'} |

|  |  |  |
| --- | --- | --- |
| [**update()**](https://www.w3schools.com/python/ref_set_update.asp) | method updates the current set, by adding items from another set. If item is already present in both sets, only once that item is shown in the set. | fruits={"banana","apple"}  fruits1={"apple", "apricot"}  fruits.update(fruits1)  print(fruits)  o/p: {'apple', 'apricot', 'banana'} |

**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) | method returns the data or specified bytes from the file. Default is -1 which means the whole file | f = open("demofile.txt", "a")  f.write("See you soon!")  f.close()  f = open("demofile.txt", "r")  print(f.read())  o/p: See you soon! |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | method returns single line or specified byte count from the file | f = open("demofile.txt", "r")  print(f.readline(5))  o/p: See y |
| [**readlines()**](https://www.w3schools.com/python/ref_file_readlines.asp) | method returns a list containing each line in the file as a list item | f = open("demofile.txt", "r")  print(f.readlines(30))  o/p: ['See you soon!’,’See you soon!’] |
| [**write()**](https://www.w3schools.com/python/ref_file_write.asp) | method writes a specified text to the file (‘a’ for appending at the end, ‘w’ for emptying the file and writing the data | f = open("demofile4.txt", "a")  f.write("See you soon!")  f.close()  file=open("demofile4.txt","r")  print(file.read())  o/p: See you soon! |
| [**writelines()**](https://www.w3schools.com/python/ref_file_writelines.asp) | method writes the items of a list to the file | f = open("demofile7.txt", "a")  f.writelines(["Hey Sravya\n", "Comeon chill"])  f.close()  f = open("demofile7.txt", "r")  print(f.read())  o/p: Hey Sravya  Comeon chill |