**THEORY QUESTIONS ASSIGNMENT**

Python-based theory

1. What is Python and what are its main features?

Answer- Python is a dynamic, high-level, free open source, and interpreted programming language. It supports both object and procedural-oriented languages. We can run any code without declaring the type of variable as it is a dynamically typed language.

Features of python are:

* Easy to code
* Free and open source
* Object-oriented language
* GUI programming support
* High-level programming language
* Extensible feature
* Portable language
* Integrated language
* Interpreted language
* Large standard library
* Dynamically typed language
* Frontend and backend development

1. Discuss the difference between Python 2 and Python 3

Answer- Python 3 is more in-demand and includes a typing system. Python 2 is outdated and uses an older syntax for the print function. While Python 2 is still in use for configuration management in DevOps, Python 3 is the current standard.

1. What is PEP 8?

Answer- PEP stands for Python Enhancement Proposal. PEP 8, sometimes spelled PEP8 or PEP-8, is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

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

Answer- In computing, a program is a specific set of ordered operations for a computer to perform, the program contains a one-at-a-time sequence of instructions that the computer follows. Typically, the program is put into a storage area accessible to the computer.

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

Answer- A process is a program that is running on your computer i.e. A process is an instance of a program running in a computer. It is close in meaning to task, a term used in some operating systems.

1. In computing/computer science what is cache?

Answer- Caches are used to store temporary files, using hardware and software components. An example of a hardware cache is a CPU cache. Cache primarily refers to a thing that is hidden or stored somewhere, or to the place where it is hidden.

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

Answer- A thread is an independent unit of execution created within the context of a process (or application that is being executed). When multiple threads are executing in a process at the same time, we get the term “multithreading.” Think of it as the application's version of multitasking.

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

Answer- Concurrency is when two or more tasks can start, run, and complete in overlapping time periods. It doesn't necessarily mean they'll ever both be running at the same instant. For example, multitasking on a single-core machine. Concurrency is a condition that exists when at least two threads are making progress. A more generalized form of parallelism that can include time-slicing as a form of virtual parallelism.

Parallelism is when tasks literally run at the same time, e.g., on a multicore processor. Parallelism: A condition that arises when at least two threads are executing simultaneously.

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

Answer- The Python Global Interpreter Lock or GIL, in simple words, is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter. This means that only one thread can be in a state of execution at any point in time. The impact of the GIL isn’t visible to developers who execute single-threaded programs, but it can be a performance bottleneck in CPU-bound and multi-threaded code.

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

Answer:

DRY- DRY, which stands for ‘don’t repeat yourself,’ is a principle of software development that aims at reducing the repetition of patterns and code duplication in favor of abstractions and avoiding redundancy.

KISS- Keep it simple, stupid (KISS) is a design principle which states that designs and/or systems should be as simple as possible. Wherever possible, complexity should be avoided in a system as simplicity guarantees the greatest levels of user acceptance and interaction.

BDUF- Big Design Up Front (BDUF) is a software development approach in which the program's design is to be completed and perfected before that program's implementation is started. It is often associated with the waterfall model of software development.

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

Answer- The garbage collector is keeping track of all objects in memory. A new object starts its life in the first generation of the garbage collector. If Python executes a garbage collection process on a generation and an object survives, it moves up into a second, older generation.

1. How is memory managed in Python?

Answer- Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, pre-allocation or caching.

1. What is a Python module?

Answer- In Python, Modules are simply files with the “.py” extension containing Python code that can be imported inside another Python Program. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.

1. What is docstring in Python?

Answer- A Python docstring is a string used to document a Python module, class, function or method, so programmers can understand what it does without having to read the details of the implementation. Also, it is a common practice to generate online (HTML) documentation automatically from docstrings.

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

Answer- The pickle module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1 or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

# Pickling in Python

import pickle

# Python object

my\_list = [11, 'Python', b'Love Python']

# Pickling

with open("data.pickle","wb") as file\_handle:

pickle.dump(my\_list, file\_handle, pickle.HIGHEST\_PROTOCOL)

print("Pickling completed!")

output- Pickling completed!

# Unpickling in Python

import pickle

# Pickling

with open("data.pickle","rb") as file\_handle:

retrieved\_data = pickle.load(file\_handle)

print(retrieved\_data)

output- [11, 'Python', b'Love Python']

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

Answer:

Pychecker and Pylint are the static analysis tools that help to find bugs in python.

Pychecker is an open-source 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 a special program to control warnings and errors, it is an extensive configuration file Pylint is also an open-source tool for static code analysis it looks for programming errors and is used for code standards 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

Pychecker can be simply installed by using pip package pip install Pychecker if suppose if you use python 3.6 version use upgrade pip install Pychecker --upgrade Pylint can be simply installed by using pip package

pip install Pylint

if suppose you use the Python 3.6 version use the upgrade

pip install Pylint –upgrade

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

Answer- Python uses a mechanism, which is known as "Call-by-Object", sometimes also called "Call by Object Reference" or "Call by Sharing". If you pass immutable arguments like integers, strings, or tuples to a function, the passing acts like Call-by-value. It's different if we pass mutable arguments. 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.

Example

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)

Output

Inside the function {'Archana': 28, 'krishna': 25, 'Ramesh': 32, 'vineeth': 25, 'alok': 30, 'Nevadan': 28}

outside the function: {'Archana': 28, 'krishna': 25, 'Ramesh': 32, 'vineeth': 25, 'alok': 30, 'Nevadan': 28}

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

Answer-

Dictionaries:

Dictionaries are similar to lists in that they store an array of objects, but these objects are accessed using keys and the object we are accessing is called the value. In Python, these are represented using {} (curly braces).

The following is an example of a Python dictionary:

{"name" : "Manasa", "surname" : "Sanga", "age" : 24}

List Comprehensions:

A list comprehension processes data to return afterward values.

Syntax of a List Comprehension

[some\_function(x) for x in original\_list if certain\_condition]

The first thing to notice is that because this is a list comprehension, it is surrounded by [](square brackets). Python recognises a list and then, by looking at the syntax inside the square brackets, is able to determine that this is a list comprehension.

1. What is namespace in Python?

Answer- A namespace is a collection of currently defined symbolic names along with information about the object that each name references. You can think of a namespace as a dictionary in which the keys are the object names and the values are the objects themselves. Each key-value pair maps a name to its corresponding object.

20.What is pass in Python?

Answer- The pass statement is used as a placeholder for future code. When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed. Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

1. What is unit test in Python?

Answer- A unit test is a test that checks a single component of code, usually modularized as a function, and ensures that it performs as expected. Unit tests are an important part of regression testing to ensure that the code still functions as expected after making changes to the code and helps ensure code stability. After making changes to our code, we can run the unit tests we have created previously to ensure that the existing functionality in other parts of the codebase has not been impacted by our changes.

1. In Python what is slicing?

Answer- 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. Slices can also be applied on third-party objects like NumPy arrays, as well as Pandas series and data frames. Slicing enables writing clean, concise, and readable code.

1. What is a negative index in Python?

Answer- Python supports “indexing from the end”, that is, negative indexing. This means the last value of a sequence has an index of -1, the second last -2, and so on.

You can use negative indexing as your advantage when you want to pick values from the end (right side) of an iterable.

- For instance, let’s get the last three names from a list of names:

names = ["x", "y", "z", "a", "b", "c"]

last = names [-1]

second\_last = names [-2]

third\_last = names [-3]

print(last, second\_last, third\_last)

Output:

c b a

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

Answer- 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.

Syntax:

The three operands in a ternary operator include:

- condition: A Boolean expression that evaluates to either true or false.

- true\_val: A value to be assigned if the expression is evaluated to true.

- false\_val: A value to be assigned if the expression is evaluated to be false.

Example-

# USING TERNARY OPERATOR

to\_check = 6

message = "Even" if to\_check%2 == 0 else "Odd"

print(message)

# USING USUAL IF-ELSE

message = ""

if(to\_check%2 == 0):

message = "Even"

else:

message = "Odd"

print(message)

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

Answer- In Python, we can pass a variable number of arguments to a function using special symbols. There are two special symbols:

1.) \*args (Non-Keyword Arguments)

2.) \*\*kwargs (Keyword Arguments)

Note- We use the “wildcard” or “\*” notation like this – \*args OR \*\*kwargs – as our function’s argument when we have doubts about the number of arguments we should pass in a function.

1.) \*args- The special syntax \*args in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list. The syntax is to use the symbol \* to take in a variable number of arguments; by convention, it is often used with the word args.

Usage of \*args

# Python program to illustrate

# \*args for variable number of arguments

def myFun(\*argv):

for arg in argv:

print (arg)

myFun('Hello', 'C', 'F', 'G')

Output:

Hello

C

F

G

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

Usage of \*\*Kwargs

# Python program to illustrate

# \*kwargs for variable number of keyword arguments

def myFun(\*\*kwargs):

for key, value in kwargs.items():

print ("%s == %s" %(key, value))

# Driver code

myFun(first ='CODE', mid ='FIRST', last='GIRLS')

Output:

last == GIRLS

mid == FIRST

first == CODE

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

Answer- The range() and xrange() are two functions that could be used to iterate a certain number of times in for loops in Python. In Python 3, there is no xrange, but the range function behaves like xrange in Python 2. If you want to write code that will run on both Python 2 and Python 3, you should use range().

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. The only particular range is displayed on demand and hence called “lazy evaluation “.

|  |  |
| --- | --- |
| range() | Xrange() |
| Execution speed is slower | Execution speed is faster. |
| Returns a list of integers. | Returns a generator object. |
| Takes more memory as it keeps the entire list of elements in memory. | Takes less memory as it keeps only one element at a time in memory. |
| All arithmetic operations can be performed as it returns a list. | Such operations cannot be performed on xrange(). |
|  |  |

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

Answer- Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file.

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

Answer-

Clustered index is created only when both the following conditions satisfy:

The data or file, that you are moving into secondary memory should be in sequential or sorted order.

There should be a key value, meaning it cannot have repeated values.

Whenever you apply clustered indexing in a table, it will perform sorting in that table only. You can create only one clustered index in a table like primary key. A clustered index is as same as a dictionary where the data is arranged by alphabetical order. In a clustered index, the index contains a pointer to block but not direct data.

Non-Clustered Index is similar to the index of a book. The index of a book consists of a chapter name and page number, if you want to read any topic or chapter then you can directly go to that page by using the index of that book. No need to go through each and every page of a book. The data is stored in one place, and the index is stored in another place. Since, the data and non-clustered index is stored separately, then you can have multiple non-clustered index in a table.

In non-clustered index, index contains the pointer to data.

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

Answer- In a database, a deadlock is a situation in which two or more transactions are waiting for one another to give up locks.

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

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

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

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| capitalize() | The capitalize() method returns a string where the first character is upper case, and the rest is lower case. | txt = "python is FUN!"  x = txt.capitalize()  print (x)  output- Python is fun! |
| casefold() | The casefold() method returns a string where all the characters are lower case. | text = "pYtHon"  convert all characters to lowercase  lowercased\_string = text.casefold()  print(lowercased\_string)  Output: python |
| center() | The center() method will center align the string, using a specified character (space is the default) as the fill character. | txt = "banana"  x = txt.center(20, "O")  print(x)  output- OOOOOOObananaOOOOOOO |
| count() | The count() method returns the number of times a specified value appears in the string. | txt = "I love apples, apple is my favorite fruit"  x = txt.count("apple", 10, 24)  print(x)  output- 1 |
| endswith() | The endswith() method returns True if a string ends with the specified suffix. If not, it returns False. | message = 'Python is fun'  # check if the message ends with fun  print(message.endswith('fun'))  # Output: True |
| find() | The find() method returns the index of first occurrence of the substring (if found). If not found, it returns -1. | message = 'Python is a fun programming language'  # check the index of 'fun'  print(message.find('fun'))  # Output: 12 |
| format() | The format() method formats the specified value(s) and insert them inside the string's placeholder. The placeholder is defined using curly brackets: {}. | #named indexes:  txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)  #numbered indexes:  txt2 = "My name is {0}, I'm {1}".format("John",36)  #empty placeholders:  txt3 = "My name is {}, I'm {}".format("John",36)  print(txt1)  print(txt2)  print(txt3)  output-  My name is John, I'm 36  My name is John, I'm 36  My name is John, I'm 36 |
| index() | The index() method finds the first occurrence of the specified value. The index() method raises an exception if the value is not found. The index() method is almost the same as the find() method, the only difference is that the find() method returns -1 if the value is not found. | txt = "Hello, welcome to my world."  x = txt.index("e")  print(x)  output- 1 |
| isalnum() | The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letters (a-z) and numbers (0-9). | txt = "Company12"  x = txt.isalnum()  print(x)  output- True |

|  |  |  |
| --- | --- | --- |
| isalpha() | The isalpha() method returns True if all the characters are alphabet letters (a-z). | txt = "Company10"  x = txt.isalpha()  print(x)  output- False |
| isdigit() | The isdigit() method returns True if all the characters are digits, otherwise False. | a = "\u0030" #unicode for 0  b = "\u00B2" #unicode for ²  print(a.isdigit())  print(b.isdigit())  output-  True  True |
| islower() | The islower() method returns True if all the characters are in lower case, otherwise False. | a = "Hello world!"  b = "hello 123"  c = "mynameisPeter"  print(a.islower())  print(b.islower())  print(c.islower())  output- False  True  False |
| isnumeric() | The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False. | a = "\u0030" #unicode for 0  b = "\u00B2" #unicode for ²  c = "10km2"  d = "-1"  e = "1.5"  print(a.isnumeric())  print(b.isnumeric())  print(c.isnumeric())  print(d.isnumeric())  print(e.isnumeric())  output- True  True  False  False  False |
| isspace() | The isspace() method returns True if all the characters in a string are whitespaces, otherwise False. | txt = " s "  x = txt.isspace()  print(x)  output- False |
| istitle() | The istitle() method returns True if all words in a text start with an upper case letter, AND the rest of the words are lower case letters, otherwise False. | a = "HELLO, AND WELCOME TO MY WORLD"  b = "Hello"  c = "22 Names"  d = "This Is %'!?"  print(a.istitle())  print(b.istitle())  print(c.istitle())  print(d.istitle())  output- False  True  True  True |
| isupper() | The isupper() method returns True if all the characters are in upper case, otherwise False. | a = "Hello World!"  b = "hello 123"  c = "MY NAME IS PETER"  print(a.isupper())  print(b.isupper())  print(c.isupper())  output-  False  False  True |
| join() | The join() method takes all items in an iterable and joins them into one string. | myDict = {"name": "John", "country": "Norway"}  mySeparator = "TEST"  x = mySeparator.join(myDict)  print(x)  output- nameTESTcountry |
| lower() | The lower() method returns a string where all characters are lower case. | txt = "Hello my FRIENDS"  x = txt.lower()  print(x)  output- hello my friends |
| lstrip() | The lstrip() method returns a copy of the string with leading characters removed (based on the string argument passed). The lstrip() removes characters from the left based on the argument (a string specifying the set of characters to be removed). | random\_string = ' this is good '  txt = " banana "  x = txt.lstrip()  print("of all fruits", x, "is my favorite")  output- of all fruits banana is my favorite |
| replace() | The replace() method replaces a specified phrase with another specified phrase. | txt = "one one was a race horse, two two was one too."  x = txt.replace("one", "three")  print(x)  output- three three was a race horse, two two was three too." |
| 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() method. | txt = "apple, banana, cherry"  # setting the maxsplit parameter to 1, will return a list with 2 elements!  x = txt.rsplit(", ", 1)  print(x)  # note that the result has only 2 elements "apple, banana" is the first element, and "cherry" is the last.  output- ['apple, banana', 'cherry'] |
| rstrip() | The rstrip() method removes any trailing characters (characters at the end a string), space is the default trailing character to remove. | txt = "banana,,,,,ssqqqww....."  x = txt.rstrip(",.qsw")  print(x)  output- banana |
| split() | The split() method splits the string from the specified separator and returns a list object with string elements. | txt = "welcome to the jungle"  x = txt.split()  print(x)  output- ['welcome', 'to', 'the', 'jungle'] |
| splitlines() | The splitlines() method splits a string into a list. The splitting is done at line breaks. | txt = "Thank you for the music\nWelcome to the jungle"  x = txt.splitlines(True)  print(x)  output- ['Thank you for the music\n', 'Welcome to the jungle'] |
| startswith() | The startswith() method returns True if the string starts with the specified value, otherwise False. | txt = "Hello, welcome to my world."  x = txt.startswith("wel", 7, 20)  print(x)  output- True |
| strip() | Strip() method in string accepts only one parameter which is optional and has characters. If the specified character is available at the beginning and end of the string, it will be removed, and the rest of the string will be returned. | txt = " banana "  x = txt.strip()  print("of all fruits", x, "is my favorite")  output- of all fruits banana is my favorite |
| swapcase() | The string swapcase() method returns a copy of a string with all lowercase characters converted to uppercase and vice versa. | txt = "Hello My Name Is MANASA"  x = txt.swapcase()  print(x)  output- hELLO mY nAME iS manasa |
| title() | The title() method returns a string where the first character in every word is upper case. Like a header, or a title. If the word contains a number or a symbol, the first letter after that will be converted to upper case. | txt = "Welcome to my 2nd world"  x = txt.title()  print(x)  output- Welcome To My 2Nd World |
| upper() | The upper() method returns a string where all characters are in upper case. | message = 'python is fun'  # convert message to uppercase  print(message.upper())  # Output: PYTHON IS FUN |

|  |  |  |
| --- | --- | --- |
| 3. | 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) | append() takes an object as an argument and adds it to the end of an existing list, right after its last element. | >>> numbers = [1, 2, 3]  >>> numbers.append(4)  >>> numbers  [1, 2, 3, 4] |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | The clear() method removes all the elements from a list. | fruits = ["apple", "banana", "cherry"]  fruits.clear()  print(fruits)  output- [] |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | The copy() method returns a shallow copy of the list. | # mixed list  prime\_numbers = [2, 3, 5]  # copying a list  numbers = prime\_numbers.copy()  print('Copied List:', numbers)  # Output: Copied List: [2, 3, 5] |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | The count() method returns the number of times the specified element appears in the list. | # create a list  numbers = [2, 3, 5, 2, 11, 2, 7]  # check the count of 2  count = numbers.count(2)  print('Count of 2:', count)  # Output: Count of 2: 3 |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | The extend() method adds all the elements of an iterable (list, tuple, string etc.) to the end of the list. | # create a list  prime\_numbers = [2, 3, 5]  # create another list  numbers = [1, 4]  # add all elements of prime\_numbers to numbers  numbers.extend(prime\_numbers)  print('List after extend():', numbers)  # Output: List after extend(): [1, 4, 2, 3, 5] |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | 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 |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | The insert() method inserts an element to the list at the specified index. | # create a list of vowels  vowel = ['a', 'e', 'i', 'u']  # 'o' is inserted at index 3 (4th position)  vowel.insert(3, 'o')  print('List:', vowel)  # Output: List: ['a', 'e', 'i', 'o', 'u'] |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | The pop() method removes the item at the given index from the list and returns the removed item. | # create a list of prime numbers  prime\_numbers = [2, 3, 5, 7]  # remove the element at index 2  removed\_element = prime\_numbers.pop(2)  print('Removed Element:', removed\_element)  print('Updated List:', prime\_numbers)  # Output:  # Removed Element: 5  # Updated List: [2, 3, 7] |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | The remove() method removes the first matching element (which is passed as an argument) from the list | # create a list  prime\_numbers = [2, 3, 5, 7, 9, 11]  # remove 9 from the list  prime\_numbers.remove(9)  # Updated prime\_numbers List  print('Updated List: ', prime\_numbers)  # Output: Updated List: [2, 3, 5, 7, 11] |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | The reverse() method reverses the elements of the list. | # create a list of prime numbers  prime\_numbers = [2, 3, 5, 7]  # reverse the order of list elements  prime\_numbers.reverse()  print('Reversed List:', prime\_numbers)  # Output: Reversed List: [7, 5, 3, 2] |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | The sort() method sorts the items of a list in ascending or descending order. | prime\_numbers = [11, 3, 7, 5, 2]  # sorting the list in ascending order  prime\_numbers.sort()  print(prime\_numbers)  # Output: [2, 3, 5, 7, 11] |

|  |  |  |
| --- | --- | --- |
| 4. | 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) | 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)  output- 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)  output- 3 |

|  |  |  |
| --- | --- | --- |
| 5. | 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) | The clear() method removes all the elements from a dictionary. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  car.clear()  print(car)  output- {} |
| [copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp) | Copy() method returns a copy (shallow copy) of the dictionary. | original\_marks = {'Physics':67, 'Maths':87}  copied\_marks = original\_marks.copy()  print('Original Marks:', original\_marks)  print('Copied Marks:', copied\_marks)  # Output: Original Marks: {'Physics': 67, 'Maths': 87}  Copied Marks: {'Physics': 67, 'Maths': 87} |
| [fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | The fromkeys() method creates a new dictionary from the given sequence of elements with a value provided by the user. | # vowels keys  keys = {'a', 'e', 'i', 'o', 'u' }  vowels = dict.fromkeys(keys)  print(vowels)  Output-  {'a': None, 'u': None, 'o': None, 'e': None, 'i': 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)  output- Mustang |
| [items()](https://www.w3schools.com/python/ref_dictionary_items.asp) | The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.items()  print(x)  output- dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)]) |
| [keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp) | The keys() method returns a view object. The view object contains the keys of the dictionary, as a list. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.keys()  print(x)  output- dict\_keys(['brand', 'model', 'year']) |
| [pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp) | The pop() method removes the specified item from the dictionary. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  car.pop("model")  print(car)  output- {'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. In versions before 3.7, the popitem() method removes a random item. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.popitem()  print(x)  output- ('year', 1964) |
| [setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | The setdefault() method returns the value of a key (if the key is in dictionary). If not, it inserts key with a value to the dictionary. | person = {'name': 'Phill', 'age': 22}  age = person.setdefault('age')  print('person = ',person)  print('Age = ',age)  Output  person = {'name': 'Phill', 'age': 22}  Age = 22 |
| [update()](https://www.w3schools.com/python/ref_dictionary_update.asp) | The update() method updates the dictionary with the elements from another dictionary object or from an iterable of key/value pairs. | marks = {'Physics':67, 'Maths':87}  internal\_marks = {'Practical':48}  marks.update(internal\_marks)  print(marks)  # Output: {'Physics': 67, 'Maths': 87, 'Practical': 48} |
| [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. | car = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  x = car.values()  print(x)  output- dict\_values(['Ford', 'Mustang', 1964]) |

|  |  |  |
| --- | --- | --- |
| 6. | 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) | Adds an element to the set | thisset = {"apple", "banana", "cherry"}  thisset.add("orange")  print(thisset)  output- {'banana', 'orange', 'cherry', 'apple'} |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set | thisset = {"apple", "banana", "cherry"}  thisset.clear()  print(thisset)  output- set() |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set | fruits = {"apple", "banana", "cherry"}  x = fruits.copy()  print(x)  output-  fruits = {"apple", "banana", "cherry"} |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.difference(y)  print(z)  output-  {'banana', 'cherry'} |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two or more sets | x = {"apple", "banana", "cherry"}  y = {"google", "microsoft", "apple"}  z = x.intersection(y)  print(z)  output- {'apple'} |
| [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)  output- 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)  output- True |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set | fruits = {"apple", "banana", "cherry"}  fruits.pop()  print(fruits)  output- {'banana', 'apple'} |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element | fruits = {"apple", "banana", "cherry"}  fruits.remove("banana")  print(fruits)  output- {'cherry', 'apple'} |
| [symmetric\_differ 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)  output- {'cherry', 'microsoft', 'google', 'banana'} |
| [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)  output- {'apple', 'banana', 'microsoft', 'cherry', 'google'} |
| [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)  output- {'apple', 'microsoft', 'banana', 'cherry', 'google'} |

|  |  |  |
| --- | --- | --- |
| 7. | 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) | Returns the file content | f = open("demofile.txt", "r")  print(f.read())  output-  C:\Users\My Name>python demo\_file\_read.py  Hello! Welcome to demofile.txt  This file is for testing purposes.  Good Luck! |
| [readline()](https://www.w3schools.com/python/ref_file_readline.asp) | Returns one line from the file | f = open("demofile.txt", "r")  print(f.readline())  output-  C:\Users\My Name>python demo\_file\_readline.py  Hello! Welcome to demofile.txt |
| [readlines()](https://www.w3schools.com/python/ref_file_readlines.asp) | Returns the list of lines from the file | f = open("demofile.txt", "r")  print(f.readlines())  output-  C:\Users\My Name>python demo\_file\_readlines.py  ['Hello! Welcome to demofile.txt\n', 'This file is for testing purposes.\n', 'Good Luck!'] |
| [write()](https://www.w3schools.com/python/ref_file_write.asp) | Writes the specified string to the file | f = open("demofile2.txt", "a")  f.write("See you soon!")  f.close()  #open and read the file after the appending:  f = open("demofile2.txt", "r")  print(f.read())  output- C:\Users\My Name>python demo\_file\_write.py  Hello! Welcome to demofile2.txt  This file is for testing purposes.  Good Luck!See you soon! |
| [writelines()](https://www.w3schools.com/python/ref_file_writelines.asp) | Writes a list of strings to the file | f = open("demofile3.txt", "a")  f.writelines(["See you soon!", "Over and out."])  f.close()  #open and read the file after the appending:  f = open("demofile3.txt", "r")  print(f.read())  output-  C:\Users\My Name>python demo\_file\_writelines.py  Hello! Welcome to demofile2.txt  This file is for testing purposes.  Good Luck!See you soon!Over and out. |