1. Python is a dynamic, high-level, open-source, interpreted programming language that is free to use. Both object-oriented and procedural-oriented programming are supported. Because Python is a dynamically typed language, we don't need to declare the type of a variable.
2. The print statement has been superseded with the print () function in Python 3. Python 2 is no longer supported, which means it will no longer receive updates or bug patches, including for security vulnerabilities. A similar policy is followed by many frameworks and other add-on projects.
3. PEP 8, often known as PEP8 or PEP-8, is a document that outlines best practises and recommendations for writing Python code. PEP 8 is primarily concerned with improving the readability and uniformity of Python code.
4. A computer programme is a set of instructions that a computer can use to carry out a certain task. A computer programme is usually developed in a programming language by a programmer.
5. A process is an instance of a computer programme that is being run by one or more threads in computing. It includes the program's code as well as its activity. A process can be made up of numerous threads of execution that execute instructions at the same time, depending on the operating system
6. Cache is a small piece of memory that is part of the CPU and is physically closer to it than RAM. It's used to store instructions and data that the CPU is likely to reuse in the future.
7. The shortest series of programmed instructions that can be controlled independently by a scheduler, which is usually part of the operating system, is called a thread of execution. Multithreading refers to a central processing unit's ability to provide several threads of operation at the same time, as enabled by the operating system.
8. Concurrency is the process of running and controlling many computations at the same time. Parallelism is the technique of performing many calculations at once. Concurrency boosts the amount of work that can be done in a given amount of time. While boosting the throughput and processing performance of the system
9. GIL is a mutex that permits only one thread to control the Python interpreter at any given time. This means that at any given time, only one thread can be in a state of execution.
10. DRY stands for "Don't Repeat Yourself," a software development principle focused at eliminating information repetition.

According to the KISS principle, most systems perform better when they are maintained simple rather than sophisticated; consequently, design should prioritise simplicity and minimise unneeded complexity.

BDUF

1. Python automatically deletes unnecessary objects (built-in types or class instances) to clear up memory. Garbage Collection is the mechanism by which Python regularly frees and reclaims memory blocks that are no longer in use.
2. private heap comprising all Python objects and data structures is used for memory management in Python.

2) Capitalize ()

The capitalise() function in Python returns a copy of the original string with the first character converted to a capital (uppercase) letter and all remaining characters in the string converted to lowercase letters.

txt = "hello, and welcome."

x = txt.capitalize()

print (x)

Casefold () The function is an aggressive lower() technique for caseless matching that converts strings to case folded strings. In a string, the casefold() technique removes any case distinctions. It's used for case-insensitive matching.

string =" RABIAT"

print(" lowercase string: ",string.casefold())

Center() The Python String centre() method is a built-in method for creating and returning a new string that has been padded with the provided character. When this method is used on a string, the original string does not change.

txt = "RabiaT"

x = txt.center(20)

print(x)

count()

The count() function in the string class returns the number of times a substring appears in the string. The count() method finds the substring in the given string and returns the number of times it appears.

fruits = ['sarah', 'anah', 'rabia']  
  
x = fruits.count("sarah")

endswith()

The method endswith() returns a value. If a string ends with the supplied suffix, it returns true; otherwise, it returns false. False

text = "Python is easy to learn."

result = text.endswith('to learn')

print(result)

result = text.endswith('to learn.')

print(result)

result = text.endswith('Python is easy to learn.')

print(result)

find() If the substring is found in the given string, the find() method returns the lowest index of the substring. If it cannot be located, it returns -1.

txt = "Hello, welcome to my world."  
  
x = txt.find("welcome")  
  
print(x)

format()

This built-in string class method allows you to perform complicated variable substitutions and value formatting.

txt = "For only {price:.2f} dollars!"

pr-int(txt.format(price = 49))

index() is a Python built-in function that searches for a specified element from the beginning of a list and returns the lowest index at which it appears.

name = ['r', 'a', 'b', 'i', 'a', 't']

index = vowels.index('r')

print('The index of e:', index)

index of the first 'i' is returned

index = vowels.index('i')

print('The index of i:', index)

isalnum() Python function that determines whether or not all characters in a string are alphanumeric. It determines if a string is made up entirely of letters, numbers, or both. isalnum() returns True if all characters are alphanumeric; otherwise, it returns False.

txt = “Rabia12"  
  
x = txt.isalnum()  
  
print(x)

isalpha()

is a built-in mechanism for dealing with strings. If all characters in the string are alphabets, the isalpha() method returns "True," otherwise "False." This function checks if the given only contains alphabet characters.

txt = "Company10"

x = txt.isalpha()

print(x)

isdigit()

sdigit() is a built-in string handling technique in Python. If all characters in the string are digits, the isdigit() method returns true; otherwise, it returns false

txt = "123456"

x = txt.isdigit()

print(x)

islower() is a built-in string handling technique in Python. If all characters in the string are lowercase, the islower() method returns True; otherwise, it returns "False."

txt = "hello world!"

x = txt.islower()

print(x)

isnumeric() method checks whether all the characters in a string are numbers. If each character is a number, isnumeric() returns the value True . Otherwise, the method returns the value False .

txt = "123456"

x = txt.isnumeric()

print(x)

isspace() is a built-in method in Python that is used to manipulate strings. If all characters in a string are whitespace characters, the isspace() method returns "True," otherwise "False." This function is used to see if the argument is completely free of whitespace characters.

s = ' \t'

print(s.isspace())

s = ' a '

print(s.isspace())

s = ''

print(s.isspace())

istitle() procedure produces a result True if all words in a text begin with an upper case letter and the rest of the word is composed of lower case letters; otherwise, False. Symbols and numerals are not taken into account.

txt = "Hello, And Welcome To My World!"

x = txt.istitle()

print(x)

isupper() is a built-in string handling technique in Python. If all characters in the string are uppercase, this method returns True; otherwise, it returns "False."

txt = "RABIA T!"

x = txt.isupper()

print(x)

join() By merging all the components of an iterable, separated by a string separator, the string method returns a string. It concatenates each member in an iterable (such as a list, string, or tuple) using a string separator (the string on which the join() method is called) and returns the result.

myTuple = ("luke", "alexr", "phil")

x = "#".join(myTuple)

print(x)

lower() is a built-in Python function for dealing with strings. The lower() function takes no arguments and converts each capital letter to lowercase to produce lowercased strings from the given string

txt = "Hello my FRIENDS"

x = txt.lower()

print(x)

istrip()

istrip() is a Python built-in method that returns a copy of the string with both the leading and following characters removed (based on the string argument passed

txt = "     mango    "  
  
x = txt.strip()  
  
print("of all fruits", x, "is my favorite")

replace() is a Python built-in function that returns a copy of the string with all occurrences of a substring replaced with another substring. New substring to replace the old substring.

txt = "I like bananas"

x = txt.replace("mango", "apples")

print(x)

rsplit()

Starting from the right, the rsplit() method separates a string into a list. This method will provide the same result as the split() method if "max" is not supplied.

txt = "anah, alex, aman"

x = txt.rsplit(", ")

print(x)

rstrip()

The rstrip() method returns a string copy with the trailing characters deleted (based on the string argument passed). It removes trailing spaces if no argument is supplied.

txt = "     banana     "  
  
x = txt.rstrip()  
  
print("of all fruits", x, "is my favorite")

split()

The split() method creates a list from a string. The separator can be specified; the default separator is any whitespace.

txt = "I am tired"

x = txt.split()

print(x)

splitlines()

txt = "I hate bananas \nI love mangoes"

x = txt.splitlines()

print(x)

startswith()

The string method determines whether a string begins with a specific substring. The startswith() method returns True if the string begins with a specified substring; otherwise, it returns False.

txt = "Hello, my name is Rabia."

x = txt.startswith("Hello")

print(x)

strip()

Strip() removes any spaces or specified characters from the beginning and end of a string in Python. strip() creates a new string that is devoid of the characters you specified to be removed.

txt = "     mango    "  
  
x = txt.strip()  
  
print("of all fruits", x, "is my favorite")

swapcase()

The string swapcase() method returns a string that has all uppercase characters converted to lowercase and vice versa

txt = "Hello My Name Is RABIYAH"

x = txt.swapcase()

print(x)

title()

The title() function in Python is a Python String Method that returns a new string after converting the initial character in each word to Uppercase and the following characters to Lowercase.

txt = "Anah is needy"

x = txt.title()

print(x)

Upper()

This method changes all lowercase letters in a string to uppercase and returns the result. If all of the characters in a string are uppercase, the Python function isupper() returns true; otherwise, it returns false.

txt = "Rabia is the best"

x = txt.upper()

print(x)

3) Python list methods: describe each method and provide an example

Append()

function adds a single item to an existing list. It does not return a new list of items, but it does add the item to the end of the existing list. The size of the list grows by one after calling the add method on it.

fruits = ['mangoe', 'beetroot', 'orange']

fruits.append("orange")

clear()

The clear() method removes all items from the dictionary. Parameters: The clear() method doesn't take any parameters

fruits = ['apple', 'banana', 'cherry', 'orange']

fruits.clear()

copy()

Python's function returns a duplicate of the Set. The = operator can be used to copy a set to another set; however, copying a set using the = operator implies that when the new set is changed, the copied set is also changed; if you don't want this behaviour, use the copy() method instead of the = operator.

Count()

Python has a built-in function called count(). It will provide you the count of a particular element in a list or string. When counting a string, start counting at the beginning and work your way to the end.

fruits = ['bob', 'alex', 'dylan']  
  
x =names.count("bob")

extend()

is a built-in function that appends the supplied list elements to the current list at the end. The items of seq are appended to the list by the list extend() function.

languages = ['French', 'English']

languages1 = ['Spanish', 'Portuguese']

languages.extend(languages1)

print('Languages List:', languages)

insert()

insert() is a Python built-in function that adds a specified element at a specified index in a list. element - the list element to be placed.

names= ['anah', 'emily', 'emma]

names.insert(1, "Georgia")

pop()

is a Python built-in function that removes and returns the list's final value or the provided index value.

fruits = ['apple', 'banana', 'cherry']

fruits.pop(1)

remove()

a built-in function in the Python programming language for removing a specific object from a list. It doesn't give you anything.

colours = ['blue', 'red', 'green’]

colours.remove("blue")

remove()

is a built-in technique in the Python programming language for reversing list objects in place.

banks = ['Lloyds', 'barclays', 'halifax']

.reverse()

Sort()

method that updates the list in the middle of it. A built-in function called sorted() creates a new sorted list from an iterable.

films = ['titanic', 'red', 'street']  
  
films.sort()

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

The method counts the number of times an element appears in the tuple. It returns the occurrence of the element put in as a parameter during the call. It necessitated a parameter that could be counted. If an item in the tuple is missing, it simply returns 0

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

x = thistuple.count(5)

print(x)

The index() function in Python finds and returns the position of a specified element in a tuple. It returns the element's first occurrence in the tuple. The index begins at 0 and ends at n-1, where n is the length of the tuple.

thistuple = (2,4,6,8,10,12,14,16,18,20,)

x = thistuple.index(8)

print(x)

5) Python dictionary methods: describe each method and provide an example

clear() method clears the dictionary of all elements. There are no parameters passed to the clear() function.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

Subjects.clear ()

print(subjects)

copy()

The list is shallow copy using the copy() technique. The difficulty with this method of copying lists is that if you change new list, old list is also changed.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.copy()

print(subjects)

fromkeys()

Dictionary in Python fromKeys() is a built-in function that generates a new dictionary from a series of elements and a user-supplied value. The dictionary with the supplied keys and values is returned by the fromKeys method.

x = ('key1', 'key2', 'key3')

y = 0

thisdict = dict.fromkeys(x, y)

print(thisdict)

get()

The following is a standard approach for accessing a value for a key in Python dictionaries. To avoid such scenarios, the get() function is utilised. If the supplied key exists in the dictionary, this method returns its value.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.get(subject)

print(subjects)

items() method returns a list of all dictionary keys and their values.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.items()

print(subjects)

keys() in the python directory function provides a view object that shows a list of all the keys in the dictionary in order of insertion

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.keys()

print(subjects)

pop()

a Python built-in function that removes and returns the list's last value or the provided index value. The value at index is deleted and popped out. If no index is specified, the last element is selected and eliminated.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.pop ()

print(subjects)

popitem()

The popitem() method removes the item from the dictionary that was last inserted. The popitem() method in versions prior to 3.7 removes a random item.

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.popitem()

print(subjects)

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

subjects = {

"biology": "genetics",

"maths": "trigonometry",

"year": 2001

}

X = subjects.setdefault(“maths”, “Pythagoras”

print(subjects)

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

The add() method adds an element to the set. If the element already exists, the add() method does not add the element.

names = {"rabia", "sarah", "anah"}

names.add("farah")

print(names)

clear()

This procedure clears the dictionary of all items. There are no parameters passed to the clear() function

names = {"rabia", "sarah", "anah"}

names.clear("farah")

print(names)

copy()

The list is copied shallowly using the copy() technique. The difficulty with this method of copying lists is that if you change new list, old list is also changed

names = {"anah", "rabia", "claire"}  
x = fruits.copy()  
print(x)

copy()

difference()

In Python, the difference between two sets is equal to the difference in the number of elements in the two sets. The difference() function returns a set that represents the difference between two sets.

x = {"royal", "blue", "corny"}

y = {"gallantry", "apple", "grey"}

z = x.difference(y)

print(z)

intersection()

The intersection() method returns a set that contains the similarity between two or more sets. The returned set contains only items that exist in both sets, or in all sets if the comparison is done with more than two sets

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)

issubset()

The issubset() method returns True if all elements of a set A are present in another set B which is passed as an argument and returns false if all elements not present.

x = {"a", "b", "c"}

y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)

print(z)

issuperset()

The issuperset() method returns True if a set has every elements of another set (passed as an argument). If not, it returns False.

x = {"f", "e", "d", "c", "b", "a"}

y = {"a", "b", "c"}

z = x.issuperset(y)

print(z)

symmetric\_difference()

The symmetric difference of two sets is returned by this technique. The set of elements that are in either A or B but not in their intersection is the symmetric difference of two sets A and B.

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.symmetric\_difference(y)

print(z)

union()

union() method returns a set that includes all items from both the original and provided sets (s).

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.union(y)

print(z)

7) Python file methods: describe each method and provide an example