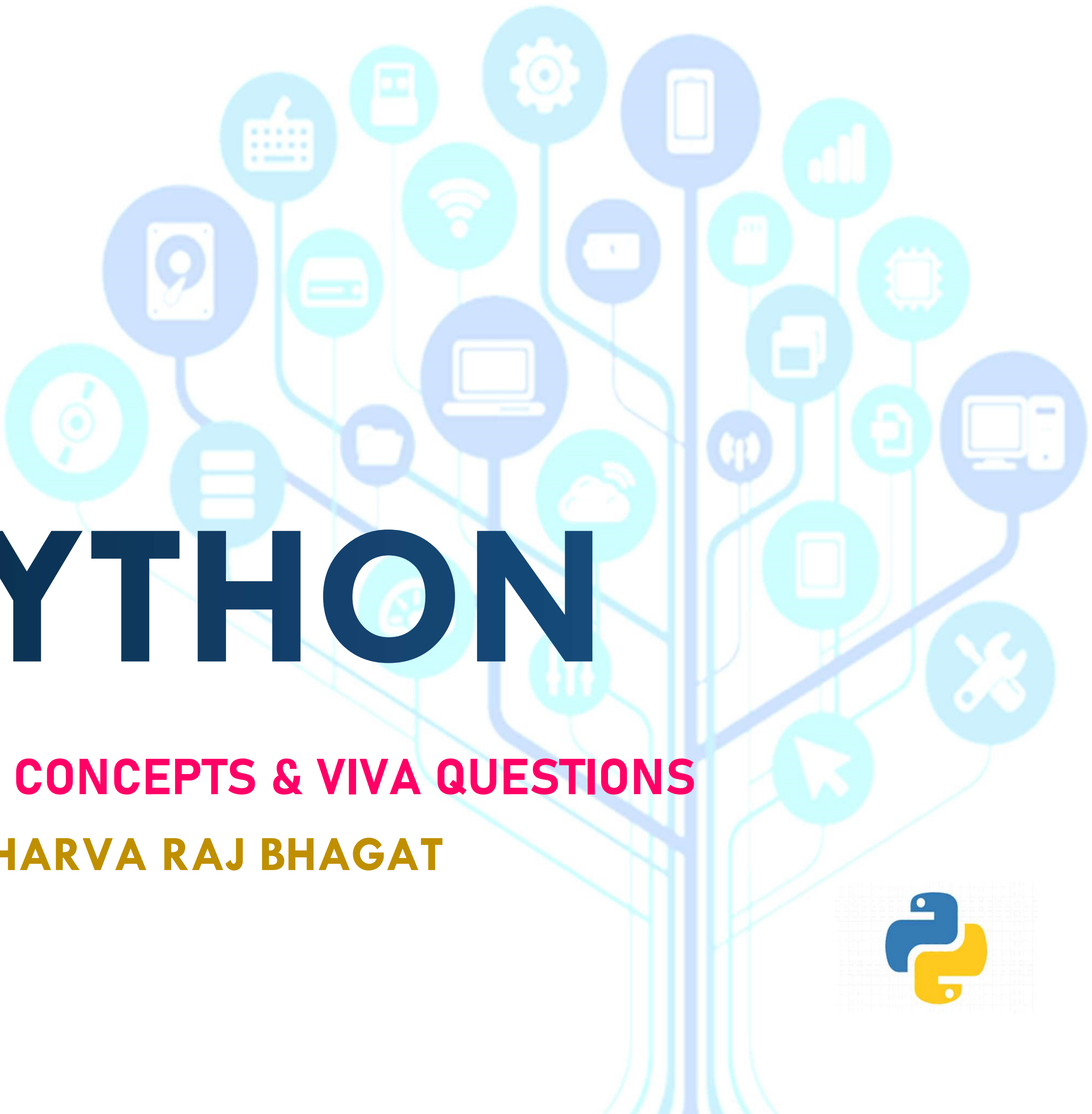

PYTHON

BASIC CONCEPTS & VIVA QUESTIONS

BY ATHARVA RAJ BHAGAT

Basics of Programming Languages

Programming Languages- The process of telling the computer what to do also known as coding.

Language Translator- A translator is a computer program that translates a program written in a given programming language into a functionally equivalent program in a different computer language, without losing the functional or logical structure of the original code.

Types of Language Translator

Compiler- A compiler is a computer program that transforms human readable complete code of computer program into the machine-readable code that a CPU can execute.

How compiler works: Source code -> Compiler-> Machine code-> output

Interpreter- An interpreter is a computer program that reads the source code of another computer program and executes that program. Because it is interpreted line by line, it is a much slower way of running a program than one that has been compiled but is easier for learners because the program can be stopped, modified and rerun without time-consuming compiles.

How interpreter works: Source code -> Interpreter -> output

Assembler- Assembler converts code written in assembly language into machine language. It works same like interpreter and compiler. The assembler program takes each program statement in the code and generates a corresponding bit stream or pattern (a series of 0's and 1's of a given length).

Python language is interpreted. But that is half correct, the python program is first compiled and then interpreted. The compilation part is hidden from the programmer.

Testing & Debugging

Testing- The tasks performed to determine the existence of defects

Debugging- The tasks performed to detect the exact location of defects. Defects are also called bugs or errors

Types of Error

Syntax Errors- They are caused by the code that somehow violates the rules of the language, easy to detect and fix errors.

Semantic Errors- Occur when a statement executes and has an effect not intended by the programmer, hard to detect during normal testing.

Run-Time Errors- Occur when the program is running and tries to do something that is against the rules.

Object-Oriented Programming Paradigm (Imperative Programming Paradigm)

It is designed to remove some drawbacks of POP.

OOP treats data as primary element, data is secure here.

OOPP has many objects as individual task or problem.

Data and functions are built around the object.

Data communicates with functions within an object.

Functions communicate outside the object.

In OOPP emphasis is on data rather than procedures.

It follows bottom-up approach design.

SUBROUTINE & CONTROL ABSTRACTION

Abstraction is a process by which the programmer can associate a name with a potentially complicated program fragment, which can then be thought of in terms of its purpose or function, rather than in terms of its implementation. We sometimes distinguish between control abstraction, in which the principal purpose of the abstraction is to perform a well-defined operation, and data abstraction, in which the principal purpose of the abstraction is to represent information.

Subroutines are the principal mechanism for control abstraction in most programming languages. A subroutine performs its operation on behalf of a caller, who waits for the subroutine to finish before continuing execution.

A subroutine that returns a value is usually called a function. A subroutine that does not return a value is usually called a procedure.

Stack Layout

Each instance of a subroutine at run time has its own frame (also called an activation record) on the stack, containing arguments and return values, local variables, temporaries, and bookkeeping information. Arguments to be passed to subsequent routines lie at the top of the frame, where the callee can easily find them.

When a subroutine returns, its frame is popped from the stack. At any given time, the stack pointer register contains the address of either the last used location at the top of the stack, or the first unused location, depending on convention. The frame pointer register contains an address within the frame.

Calling Sequences

Maintenance of the subroutine call stack is the responsibility of the calling sequence—the code executed by the caller immediately before and after a subroutine call—and of the prologue (code executed at the beginning) and epilogue (code executed at the end) of the subroutine itself. Sometimes the term "calling sequence" is used to refer to the combined operations of the caller, the prologue, and the epilogue.

Tasks that must be accomplished on the way into a subroutine include passing parameters, saving the return address, changing the program counter, changing the stack pointer to allocate space, saving registers, executing initialization code for any objects in the new frame that require it. Tasks that must be accomplished on the way out include passing return parameters or function values, executing finalization code for any local objects that require it, deallocating the stack frame (restoring the stack pointer), restoring other saved registers (including the frame pointer), and restoring the program counter.

Parameter Passing

There are two parameters passing mechanism in Python:

Pass by references and Pass by value

By default, all the parameters (arguments) are passed "by reference" to the functions. Thus, if you change the value of the parameter within a function, the change is reflected in the calling function as well. It indicates the original variable.

The pass by value is that whenever we pass the arguments to the function only values pass to the function, no reference passes to the function. It makes it immutable that means not changeable. Both variables hold the different values, and original value persists even after modifying in the function.

EXCEPTIONS

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e. at run time, that disrupts the normal flow of the program's instructions.

Error: An Error indicates serious problem that a reasonable application should not try to catch.

Exception: Exception indicates conditions that a reasonable application might try to catch.

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block. After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible

You can use a finally: block along with a try: block. The finally block is a place to put any code that must execute, whether the try-block raised an exception or not.

User-Defined Exceptions

Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.

Here is an example related to RuntimeError. Here, a class is created that is subclassed from RuntimeError. This is useful when you need to display more specific information when an exception is caught.

In the try block, the user-defined exception is raised and caught in the except block. The variable e is used to create an instance of the class Networkerror.

```
class Networkerror(RuntimeError):
   def __init__(self, arg):
      self.args = arg
```

So once you defined above class, you can raise the exception as follows –

```
try:
   raise Networkerror("Bad hostname")
except Networkerror,e:
   print e.args
```

COROUTINES

A coroutine is represented by a closure (a code address and a referencing environment), into which we can jump by means of a nonlocal goto, in this case a special operation known as transfer. The principal difference between the two abstractions is that a continuation is a constant—it does not change once created—while a coroutine changes every time it runs.

Input Statement in Python

The input() function is used to take the input from user. We can pass a string parameter as well to print what the user has to input.

Output Statement in Python

The print() function is used to display the output on the terminal. We can pass string parameter or variables or can concatenate both to display the desired output.

Operators in Python

Operators in Python are of the following types:

1. Arithmetic Operators

Addition: +	Integral Quotient: //
Subtraction: -	Remainder (Modulus): %
Multiplication: *	Exponent: **
Division: /	

2.Comparison Operators

Greater than >	Double Equals to ==
Less than <	Greater than equal to >=
Not equal to !=	Less than equal to <=

3. Logical Operators

and – used to combine conditions that all necessarily be true not – used to negate the value

or – used to combine conditions where either of them can be true and not necessarily all

4.Assignment Operator

The equal to (=) operator called as assignment operator is used to assign the value on RHS to variable present on LHS

Python Operators Precedence

1	**	(Exponentiation (raise to the power))
2	~ + -	(Complement, unary plus and minus)
3	* / % //	(Multiply, divide, modulo and floor division)
4	+ -	(Addition and subtraction)
5	>> <<	(Right and left bitwise shift)
6	&	(Bitwise 'AND')
7	^ |	(Bitwise exclusive `OR' and regular `OR')
8	<= < > >=	(Comparison operators)
9	<> == !=	(Equality operators)
10	= %= /= //= -= += *= **=	(Assignment operators)

Decision making in Python

Decision Making in Python can be done using the if-else statement or the if-elif-else block.

Syntax:

```
 if condition:
   statements to be executed if the condition is true
elif condition:
   statements to be executed if the condition is true
else:
   statements to be executed if none of the conditions were true
```

Loops in Python

For Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

The for loop does not require an indexing variable to set beforehand.

Syntax:

```
for variable_name in  string / list/ range() :
   statements to be executed
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function.

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(start, end)

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(start, end, increment-by-value)

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

The while loop requires relevant variables to be initialized earlier.

Syntax:

```
while condition:
   statements to be executed
```

The break Statement

With the break statement we can stop the loop even if the while condition is true

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next.

The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error

Inheritance : Types – Single, Multiple, Multilevel, Hierarchical

Inheritance is defined as the capability of one class to derive or inherit the properties from some other class and use it whenever needed. Inheritance provides the following properties:

- It represents real-world relationships well.
- It provides reusability of code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

Single Inheritance – where a derived class acquires the members of a single super class.

Multi-level inheritance – In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class.

This is similar to a relationship representing a child and grandfather.

Hierarchical inheritance – from one base class you can inherit any number of child classes

Multiple inheritance – a derived class is inherited from more than one base class.

Syntax for inheritance:

```
 class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
    pass
```

Python also has a super() function that will make the child class inherit all the methods and properties from its parent.

Object Oriented Programming in Python

Python is an object oriented programming language.

It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods.

Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Module

A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables.

Advantages of modules –

- Reusability : Working with modules makes the code reusable.

- Simplicity: Module focuses on a small proportion of the problem, rather than focusing on the entire problem.

- Scoping: A separate namespace is defined by a module that helps to avoid collisions between identifiers.

The import statement is used to import any user-defined / built-in module so as to use the methods present in the module.

File Handling in Python

Python has several functions for creating, reading, updating, and deleting files.

The key function for working with files in Python is the open() function.

The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

r - Read - Default value. Opens a file for reading, error if the file does not exist

a - Append - Opens a file for appending, creates the file if it does not exist

w - Write - Opens a file for writing, creates the file if it does not exist

x - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

t - Text - Default value. Text mode

b - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

f = open(“demofile.txt”)

The code above is the same as:

f = open(“demofile.txt”, ”rt”)

Because ”r” for read, and ”t” for text are the default values, you do not need to specify them.

CSV File

CSV (Comma Separated Values) is a simple file format used to store tabular data, such as a spreadsheet or database.

The CSV module implements classes to read and write tabular data in CSV format.

- fields and rows have been already defined. fields is a list containing all the field names. rows is a list of lists. Each row is a list containing the field values of that row.
- with open(filename, 'w') as csvfile:
- csvwriter = csv.writer(csvfile)
- Here, we first open the CSV file in WRITE mode. The file object is named as csvfile. The file object is converted to csv.writer object. We save the csv.writer object as csvwriter.
- csvwriter.writerow(fields)
- Now we use writerow method to write the first row which is nothing but the field names.
- csvwriter.writerows(rows).

Type Casting

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

LIST is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type. You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

TUPLE is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values.

To access values in lists or tuples , use the square brackets for slicing along with the index or indices to obtain value available at that index.

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered, changeable and does not allow duplicates. Dictionary items are presented in key:value pairs, and can be referred to by using the key name. Dictionaries cannot have two items with the same key.

Sets are a collection of distinct objects. These are useful to create lists that only hold unique values in the dataset . It is unordered but mutable .

Arrays

Python does not have built-in support for Arrays, but Python Lists can be used instead.

Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list

count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort	Sorts the list

NumPy Introduction

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

Mathematical and logical operations on arrays.

Fourier transforms and routines for shape manipulation.

Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

SciPy Introduction

SciPy is a scientific computation library that uses NumPy underneath.

SciPy stands for Scientific Python.

It provides more utility functions for optimization, stats and signal processing.

Pandas Introduction

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

Matplotlib Introduction

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

A bar chart displays categorical data with rectangular bars whose length or height corresponds to the value of each data point.

Bar charts can be visualized using vertical or horizontal bars. Bar charts are best used to compare a single category of data or several. When comparing more than one category of data, the bars can be grouped together to created a grouped bar chart.

Python String Functions

capitalize()

method returns a copy of the string with only its first character capitalized.

center()

returns centered in a string of length width. Padding is done using the specified fillchar. Default filler is a space.

count()

returns the number of occurrences of substring sub in the range [start, end].

sub − This is the substring to be searched.

start − Search starts from this index. First character starts from 0 index. By default search starts from 0 index.

end − Search ends from this index. First character starts from 0 index. By default search ends at the last index.

endswith() returns True if the string ends with the specified suffix, otherwise return False optionally restricting the matching with the given indices start and end.

Syntax

str.endswith(suffix[, start[, end]])

Parameters

suffix − This could be a string or could also be a tuple of suffixes to look for.

start − The slice begins from here.

end − The slice ends here.

isalnum() checks whether the string consists of alphanumeric characters.

NOTE: it displays true if string contains alphabets or numbers without space

```
str = "this2009";  # No space in this string, space is not alphabet or digit
print str.isalnum()
str = "this is string example....wow!!!";
print str.isalnum()
```

When we run above program, it produces following result −

```
True
False
```

isdigit() checks whether the string consists of digits only.

islower() checks whether all the case-based characters (letters) of the string are lowercase.

The upper() method returns the string in upper case:

The lower() method returns the string in lower case

len() returns the length of the string.

max() returns the max alphabetical character from the string str.

To check if a certain phrase or character is present in a string, we can use the keyword in. To check if it does not exist use not in.

Slicing – obtaining a sub-string from the given string by slicing it respectively from start to end.

Get the characters from position 2 to position 5 (not included)

```
b = "Hello, World!"
print(b[2:5])
```

Slice From the Start

By leaving out the start index, the range will start at the first character

Slice To the End

By leaving out the end index, the range will go to the end

Negative Indexing

Use negative indexes to start the slice from the end of the string

Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

b = "Hello, World!"

print(b[-5:-2])

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

The strip() method removes any whitespace from the beginning or the end:

The replace() method replaces a string with another string

String Concatenation

To concatenate, or combine, two strings you can use the + operator

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

Example

age = 36

txt = "My name is John, I am " + age

print(txt)

We can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

Example

Use the format() method to insert numbers into strings:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

The format() method takes unlimited number of arguments, and are placed into the respective placeholders

Python Functions

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function.

A function can return data as a result.

Advantages:

1. Duplicated codes in a program can be reduced.
2. Complex codes can be divided in to smaller parts.
3. Improves the efficiency of the code.

Creating a Function

In Python a function is defined using the def keyword:

Example

```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

This way the function will receive a tuple of arguments, and can access the items accordingly.

Keyword Arguments

You can also send arguments with the key = value syntax.

This way the order of the arguments does not matter.

Default Parameter Value

If we call the function without argument, it uses the default value

Example

```
def my_function(country = "Norway"):
  print("I am from " + country)
```

```
my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

Example

```
def my_function(food):
  for x in food:
    print(x)

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

my_function(fruits)
```

Return Values

To let a function return a value, use the return statement:

Anonymous Functions (Lambda, Map, Reduce, Filter)

The Anonymous Functions – Lambda Functions

These functions are called anonymous because they are not declared in the standard manner by using the def keyword. You can use the lambda keyword to create small anonymous functions.

Lambda forms can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.

An anonymous function cannot be a direct call to print because lambda requires an expression

Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.

Although it appears that lambda's are a one-line version of a function, they are not equivalent to inline statements in C or C++, whose purpose is by passing function stack allocation during invocation for performance reasons.

Syntax

The syntax of lambda functions contains only a single statement, which is as follows −

lambda [arg1 [,arg2,.....argn]]:expression

Example

Add 10 to argument a, and return the result:

x = lambda a : a + 10

print(x(5))

Example

Maps

Python Maps also called ChainMap is a type of data structure to manage multiple dictionaries together as one unit.

The combined dictionary contains the key and value pairs in a specific sequence eliminating any duplicate keys.

The best use of ChainMap is to search through multiple dictionaries at a time and get the proper key-value pair mapping.

Creating a ChainMap

We create two dictionaries and club them using the ChainMap method from the collections library.

Then we print the keys and values of the result of the combination of the dictionaries.

If there are duplicate keys, then only the value from the first key is preserved

```
import collections

dict1 = {'day1': 'Mon', 'day2': 'Tue'}

dict2 = {'day3': 'Wed', 'day1': 'Thu'}

res = collections.ChainMap(dict1, dict2)

# Creating a single dictionary

print(res.maps)

print(list(res.keys()))

print('Keys = {}'.format(list(res.keys())))

print('Values = {}'.format(list(res.values())))

print()

# Print all the elements from the result

print('elements:')

for key, val in res.items():

    print('{} = {}'.format(key, val))

print()

# Find a specific value in the result

print('day3 in res: {}'.format(('day3' in res)))

print('day4 in res: {}'.format(('day4' in res)))
```

Map Reordering

If we change the order the dictionaries while clubbing them in the above example we see that the position of the elements get interchanged as if they are in a continuous chain. This again shows the behaviour of Maps as stacks.

```
import collections
dict1 = {'day1': 'Mon', 'day2': 'Tue'}
dict2 = {'day3': 'Wed', 'day4': 'Thu'}
res1 = collections.ChainMap(dict1, dict2)
print(res1.maps,'\n')
res2 = collections.ChainMap(dict2, dict1)
print(res2.maps,'\n')
```

Updating Map

When the element of the dictionary is updated, the result is instantly updated in the result of the ChainMap.

In the below example we see that the new updated value reflects in the result without explicitly applying the ChainMap method again.

TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

The Menu widget is used to create various types of menus (top level, pull down, and pop up) in the python application.

The top-level menus are the one which is displayed just under the title bar of the parent window. We need to create a new instance of the Menu widget and add various commands to it by using the add() method.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

CONSTRUCTOR IN PYTHON

Constructors are generally used for instantiating an object.The task of constructors is to initialize(assign values) to the data members of the class when an object of class is created.In Python the __init__() method is called the constructor and is always called when an object is created.

Syntax of constructor declaration :

```
def __init__(self):
    # body of the constructor
```

Types of constructors :

Default constructor :The default constructor is simple constructor which doesn’t accept any arguments.It’s definition has only one argument which is a reference to the instance being constructed.

Parameterized constructor :constructor with parameters is known as parameterized constructor.The parameterized constructor take its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

Generators in python

Functions that return an iterable set of items are called generators.

Comments in python

Comments in Python start with a # character. However, alternatively at times, commenting is done using docstrings(strings enclosed within triple quotes).

Self in Python?

Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. It helps to differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

What are the key features of Python?

Python is an interpreted language. That means that, unlike languages like C and its variants, Python does not need to be compiled before it is run. Other interpreted languages include PHP and Ruby.

Python is dynamically typed, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error

Python is well suited to object orientated programming in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++’s public, private).

In Python, functions are first-class objects. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects

Writing Python code is quick but running it is often slower than compiled languages. Fortunately, Python allows the inclusion of C-based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it’s really quite quick because a lot of the number-crunching it does isn’t actually done by Python

Python finds use in many spheres – web applications, automation, scientific modeling, big data applications and many more. It’s also often used as “glue” code to get other languages and components to play nice.

Applications of Python:

1. Artificial Intelligence and Machine Learning

Python’s simplicity, consistency, platform independence, great collection of resourceful libraries, and an active community make it the perfect tool for developing AI and ML applications. Some of the best Python packages for AI and ML are:

SciPy for advanced computing

Pandas for general-purpose data analysis

Seaborn for data visualization

Keras, TensorFlow, and Scikit-learn for ML

NumPy for high-performance scientific computing and data analysis

2.Web Development

Django, Pyramid, Flask, and Bottle for developing web frameworks and even advanced content management systems like Plone and Django CMS. These web frameworks are packed with standard libraries and modules which simplify tasks like content management, database interaction, and interfacing with internet protocols like HTTP, SMTP, XML, JSON, FTP, IMAP, and POP.

3.Game Development

Python comes loaded with many useful extensions (libraries) that come in handy for the development of interactive games. For instance, libraries like PySoy (a 3D game engine that supports Python 3) and PyGame are two Python-based libraries used widely for game development. Python is the foundation for popular games.

4. Scientific and Numeric Applications

Python has become a crucial tool in scientific and numeric computing. In fact, Python provides the skeleton for applications that deal with computation and scientific data processing. Apps like FreeCAD (3D modeling software) and Abaqus (finite element method software) are coded in Python.

Some of the most useful Python packages for scientific and numeric computation include:

SciPy (scientific numeric library)

Pandas (data analytics library)

IPython (command shell)

Numeric Python (fundamental numeric package)

Natural Language Toolkit (Mathematical And text analysis)

5. GUI Desktop Applications

Python not only boasts of an English-like syntax, but it also features a modular architecture and the ability to work on multiple operating systems. These aspects, combined with its rich text processing tools, make Python an excellent choice for developing desktop-based GUI applications.

Python offers many GUI toolkits and frameworks that make desktop application development a breeze. PyQt, PyGtk, Kivy, Tkinter, WxPython, PyGUI, and PySide are some of the best Python-based GUI frameworks that allow developers to create highly functional Graphical User Interfaces (GUIs).

6.Web Scraping Applications

Python is a nifty tool for extracting voluminous amounts of data from websites and web pages. The pulled data is generally used in different real-world processes, including job listings, price comparison, R&D, etc.

BeautifulSoup, MechanicalSoup, Scrapy, LXML, Python Requests, Selenium, and Urllib are some of the best Python-based web scraping tools.

7. Image Processing and Graphic Design Applications

Alongside all the uses mentioned above, Python also finds a unique use case in image processing and graphic design applications.

8. Operating Systems

Python is the secret ingredient behind many operating systems as well, most popularly of Linux distributions.

Popular Packages

SciPy for advanced computing

Pandas for general-purpose data analysis

Seaborn for data visualization

Keras, TensorFlow, and Scikit-learn for Machine Learning

NumPy for high-performance scientific computing and data analysis

Django, Pyramid, Flask – Web Development

Tkinter, PyQt5, Kivy – GUI

PyGame – Game Development

BeautifulSoup BS4 – Web Scraping

OpenCV , PyTorch – Real Time Computer Vision – used to operate camera functions

What is the usage of help() and dir() function in Python?

Help() and dir() both functions are accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

Help() function: The help() function is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.

Dir() function: The dir() function is used to display the defined symbols.

Whenever Python exits, why isn't all the memory de-allocated?

Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.

It is impossible to de-allocate those portions of memory that are reserved by the C library.

On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

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

Ans: We use *args when we aren't sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. **kwargs is used when we don't know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments.

Method overloading

In method loading, we create multiple methods with the same name but with different data types of arguments or with different no of parameters.

METHOD OVERRIDING

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding. In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

What is split used for?

The split() method is used to separate a given string in Python.

What is pickling and unpickling?

Ans: Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

What is Polymorphism in Python?

Ans: Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

Define encapsulation in Python?

Ans: Encapsulation means binding the code and the data together. A Python class in an example of encapsulation.

How do you do data abstraction in Python?

Ans: Data Abstraction is providing only the required details and hiding the implementation from the world. It can be achieved in Python by using interfaces and abstract classes.

Does python make use of access specifiers?

Ans: Python does not deprive access to an instance variable or function. Python lays down the concept of prefixing the name of the variable, function or method with a single or double underscore to imitate the behavior of protected and private access specifiers.

How to create an empty class in Python?

Ans: An empty class is a class that does not have any code defined within its block. It can be created using the pass keyword. However, you can create objects of this class outside the class itself. PASS command does nothing when its executed. it's a null statement.

Assertion ?

An assertion is a sanity-check that you can turn on or turn off when you are done with your testing of the program

The assert keyword is used when debugging code.

The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

Frame in GUI

Frames are useful for the organization of a GUI. Frames get built into the grid of the window and then in turn each have a grid of their own. This means that nesting frames may be necessary if the GUI is going to look nice.

Canvas in GUI

A canvas allows for various shapes and designs to be drawn onto it. These shapes will remain if more are added unless the shape's pixels are completely overwritten

What do you mean by Python literals?

Literals can be defined as a data which is given in a variable or constant.

Strings, numbers, Boolean etc.

What is zip() function in Python?

Python zip() function returns a zip object, which maps a similar index of multiple containers. It takes an iterable, convert into iterator and aggregates the elements based on iterables passed. It returns an iterator of tuples.

Give an example of shuffle() method?

This method shuffles the given string or an array. It randomizes the items in the array. This method is present in the random module. So, we need to import it and then we can call the function. It shuffles elements each time when the function calls and produces different output.

What is the usage of enumerate () function in Python?

The enumerate() function is used to iterate through the sequence and retrieve the index position and its corresponding value at the same time.

How is memory managed in Python?

Memory is managed in Python by the following way:

The Python memory is managed by a Python private heap space. All the objects and data structures are located in a private heap. The programmer does not have permission to access this private heap.

We can easily allocate heap space for Python objects by the Python memory manager. The core API gives access of some tools to the programmer for coding purpose.

Python also has an inbuilt garbage collector, which recycle all the unused memory and frees the memory for the heap space.

What are the rules for a local and global variable in Python?

In Python, variables that are only referenced inside a function are called implicitly global. If a variable is assigned a new value anywhere within the function's body, it's assumed to be a local. If a variable is ever assigned a new value inside the function, the variable is implicitly local, and we need to declare it as 'global' explicitly. To make a variable globally, we need to declare it by using global keyword. Local variables are accessible within local body only. Global variables are accessible anywhere in the program, and any function can access and modify its value.

Explain docstring in Python?

The Python docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. It provides a convenient way to associate the documentation.

String literals occurring immediately after a simple assignment at the top are called "attribute docstrings".

String literals occurring immediately after another docstring are called "additional docstrings".

Python uses triple quotes to create docstrings even though the string fits on one line.

Docstring phrase ends with a period (.) and can be multiple lines. It may consist of spaces and other special chars.

Example

```
# One-line docstrings
def hello():
    """A function to greet."""
    return "hello"
```

How Python does Compile-time and Run-time code checking?

In Python, some amount of coding is done at compile time, but most of the checking such as type, name, etc. are postponed until code execution. Consequently, if the Python code references a user-defined function that does not exist, the code will compile successfully. The Python code will fail only with an exception when the code execution path does not exist.

How to send an email in Python Language?

To send an email, Python provides smtplib and email modules. Import these modules into the created mail script and send mail by authenticating a user.

Database Connection in Python

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application

You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following −

Importing the API module.

Acquiring a connection with the database.

Issuing SQL statements and stored procedures.

Closing the connection

Mention the differences between Django, Pyramid and Flask.

Flask is a “microframework” primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.

Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.

Django can also be used for larger applications just like Pyramid.

Is Django better than Flask?

Ans: Django and Flask map the URL’s or addresses typed in the web browsers to functions in Python.

Flask is much simpler compared to Django but, Flask does not do a lot for you meaning you will need to specify the details, whereas Django does a lot for you wherein you would not need to do much work. Django consists of prewritten code, which the user will need to analyze whereas Flask gives the users to create their own code, therefore, making it simpler to understand the code. Technically both are equally good and both contain their own pros and cons.