PYTHON PROGRAMMING LANGUAGE

INTRODUCTION TO PYTHON

Not many programmers would develop programming languages and name it after a TV show they love for uniqueness and mystery. Python was implemented in 1989 by Guido van Rossum at Centrum Wiskunde Informatica, Netherlands. Python was developed out of the need to help beginners understand a programming language easily. Python is a high-level scripting language it is also a true Object Oriented Language and is available on many platforms. Has properties such as classes, several core data types, exceptional handling and functions. Python is transferable and is preferably used for illustrative projects.

INTENDED USE FOR PYTHON LANGUAGE

The main reason why Python was developed was to rectify the flaws of ABC Programming Language which was interfacing with the Amoeba Operating System. Some of the flaws included its inability to make new changes such as creating a Graphical User Interface therefore undermining access to the underlying file system and the operating system.

Another reason why it was developed was to aid in code readability and enhance developer productivity. This is because Python statements may be simple but have high capabilities.

one statement.

Programmers are able to express their concepts by just using a few lines of codes written in Python. Python has a structure that is similar to ABC Programming Language.

APPLICATIONS OF PYTHON LANGUAGE

Used by data scientists due to stability for Machine Learning and Artificial Intelligent projects as the Artificial Intelligence and Machine Learning packages and libraries are available.

Used by programmers to program Graphical User Interfaces and Application Programming Interfaces and read files.

Web developers use it for web development frameworks due to its extensive libraries, database access and content management. Some of them include Mozilla.

Data scientists use Python to be able to excavate the required information to be able to analyze it.

Due to stability it is used for multitasking programs such as applications for music and playing games.

Used to program highly intellectual information systems.

Used to make back end databases due to the use of object oriented paradigm.

Used for accuracy especially when creating sites or programs that require user interaction.

PARADIGMS USED FOR PYTHON

These refer to ways in which issues arising from program execution can be solved using python programming language.Paradigms expound on how a programming language specifically Python in this case how codes are acted upon by the interpreter to produce the desired result of the programmer.

Python supports three paradigms which include:

OBJECT ORIENTED PROGRAMMING PARADIGM

An object is formed from a class that has both Variables and Methods or Processes. This style relates data variables and methods by use of inheritance and encapsulation thus making the code reusable.

Advantages

The fact that it uses encapsulation prevents members outside the specified class from viewing.

Inheritance increases the code’s capability to be used a couple of times.

FUNCTIONAL PROGRAMMING PARADIGM

This style uses mathematical functions and overstatements. Statements are used as functional expressions as an expression to produce a value. Focuses mainly on what to solve and not how to solve a problem.

Advantage

The use of mathematical functions makes it accurate in solving problems related to python.

Disadvantage

It may have low speed in execution due to recursion.

PROCEDURAL PROGRAMMING PARADIGM

In this paradigm computational steps are divided into modules or grouped in functions and executed. A serial code instructs a computer in each step as it performs a task. It helps in modularity of the code.

Functional uses Declarative programming model while Object Oriented uses Imperative programming model.

Advantage

Offers step by step procedure thus codes are executed in small chunks and not as a whole.

Disadvantage

Requires a lot of memory as codes are broken down and solved in small chunks.

THE DISIMILARIES BETWEEN FUNCTIONAL AND OBJECT ORIENTED

In functional programming a fixed function only executes a particular job while in object oriented classes and objects are employed.

Information once imputed cannot be swapped in functional while in object oriented information can be substituted.

In functional a task is specified while in object oriented it shows the ways in which the issue can be solved.

No particular sequence is required for functional programming while in object oriented a particular structure for the code is expected for execution.

Object oriented requires one to know how inheritance works while functional knowledge on inheritance is not important.

EXPRESSIVENESS IN PYTHON

A single line of a python code can do more than a single task. It supports multiple assignments in one statement. It is reputable for batteries include philosophy whereby when acquiring Python it has in-built libraries; arbitrary data can be stored in a tuple without declaring a class and Python can be used on any Operating system provided that it has an interpreter for Python codes.

GARBAGE COLLECTION IN PYTHON

It refers to release of memory when an object is no longer in use, unused objects are destroyed and reuses its memory slots for new objects. It uses an aspect known as Reference Counting. Another aspect is Generational Garbage Collecting which focuses on allocating memory to new items. It has levels called generations and classifies objects according to the amount of time they will be in the memory. There is a higher generation and a lower generation. There are two higher generations which store objects that are permanent while the lower generation has objects that are temporary . An object’s reference count should be less than one for the garbage collector to run. An object’s reference count increases when it is assigned new names and reduces when it is deleted or the object being referenced is out of scope. Example;

This \_\_del\_\_() destructor shows what should be removed –

#!/usr/bin/python

class Point:

def \_\_init\_\_( self, x=0, y=0):

self.x = x self.y = y def \_\_del\_\_(self):

class\_name = self.\_\_class\_\_.\_\_name\_\_ print class\_name, "destroyed"

pt1 = Point()

pt2 = pt1 pt3 = pt1

print id(pt1), id(pt2), id(pt3) # prints the ids of the objects

del pt1

del pt2

del pt3

OUTPUT:

3083401324 3083401324 3083401324

Point destroyed

Advantage

It is an automatic procedure that saves the time the programmer would have used to partition space for the objects.

Programmer does not need to keep track of the memory space.

Disadvantage

Tampering with the garbage collector functionality may lead to unpleasant results.

BENCHMARKING

This refers to the speed at which the code is being executed and where the problem is when running the program. This helps to optimize the code. A whole program is benchmarked as a current state, combined and decomposed into a smaller program. It uses the Timeit module which helps avoid errors during performance of a code and time small codes in Python. Example:

import timeit

import time

def functionA():

print("Function A starts the execution:")

print("Function A completes the execution:")

def functionB():

print("Function B starts the execution")

print("Function B completes the execution")

start\_time = timeit.default\_timer()

functionA()

print(timeit.default\_timer() - start\_time)

start\_time = timeit.default\_timer()

functionB()

print(timeit.default\_timer() - start\_time)

OUTPUT:

Function A starts the execution:

Function A completes the execution:

0.0014599495514175942

Function B starts the execution

Function B completes the execution

0.0017024724827479076

import timeit

import time

One can also use another timer apart from using the Timeit module. This function is known as a Decorator.

Advantage

One is able to know the issues associated with the program and why it is slow.

Disadvantage

The wrong function for checking may mislead the programmer.

MULTITHREADING IN PYTHON

It is when many threads run at the same time. Threads run independently. They are created in three ways;

Without creating a class- Main thread creates a child that executes the function. For Example;

from threading import \*

print(current\_thread().getName())

def mt():

print("Child Thread")

child=Thread(target=mt)

child.start()

print("Executing thread name :",current\_thread().getName())

This Returns:

MainThread

Child Thread

Executing thread name : MainThread

Extending a thread class-The child thread overrides two methods the run () and \_Init\_ (). The first variable shows where the object is. Sample can be as follows;

import threading

import time

class mythread(threading.Thread):

def run(self):

for x in range(5):

print("Hello from child")

a = mythread()

a.start()

a.join()

print("Bye from",current\_thread().getName())

Result is:

Hello from child

Hello from child

Hello from child

Hello from child

Hello from child

Bye from MainThread

Without extending thread class-The main thread executes the last print statements and child executes last .

from threading import \*

class ex:

def myfunc(self): #self necessary as first parameter in a class func

for z in range(4):

print("Child")

myobj=ex()

thread1=Thread(target=myobj.myfunc)

thread1.start()

thread1.join()

print("Bye")

Prints child 4 times:

Child

Child

Child

Child

Bye

EXCEPTIONAL HANDLING

It is a condition whereby a program stops executing due to interruption of the flow. When an exception occurs the program stops the execution thus other codes are not executed. There are many built in exceptions such as ZeroDivisionError which occurs when the number is divided by zero,NameError appears when the name is out of scope,IndentationError is found when one does not use the proper indentation while coding and InputOutput Error is identified when the either Input Or output process is not successful. Statements in the Try block may make a program not to be executed. Statements in exception help in manouvering over the statements in the Try block. Example;

a = 12

s = "hello"

try:

print("inside try")

print(a + s) # will raise TypeError

print("Printed using original data types")

except TypeError: # will handle only TypeError

print("inside except")

print(str(a) + s)

print("Printed using type-casted data types")

HIGH ORDER FUNCTION CAPABILITIES OF PYTHON

High order function can take one or multiple functions as an argument or return a function as a result. These functions include;

MAP FUNCTION

Functions and sequences are used as arguments and returns a list after applying the function to each sequence,

Example;

#function for performing the multiplication operation

def multiply(n):

#returning the values

return n\*n

#driver code to test the program

num = (10,15,78,25)

#map function(function, sequence/iterables)

result = map(multiply, num)

#returning the result in a list

print(list(result))

Output :

[100, 225, 6084, 625]

FILTER FUNCTION

It uses a function and sequences. It returns an iterator which is passed through the function test and returns a true value. Example;

#driver code to test the program

numbers = [55, 92, 27, 48, 34, 62, 71, 18, 28, 43]

#function to check the numbers whether they are greater than 50 or not

def numCheck(n):

if n < 50:

return False

else:

return True

#passing through the filter function

result = filter(numCheck, numbers)

#displaying the result

for n in result:

print(n)

Output :

55

92

62

71

SORTED FUNCTION

Function arranges names and numbers to return a list that is in a particular order. Example;

#declaring string

str = "codespeedy"

#calling sorted function

result = sorted(str) # sorting string

#displaying result

print(result)

Output :

['c', 'd', 'd', 'e', 'e', 'e', 'o', 'p', 's', 'y']

METHOD OVERLOADING IN PYTHON

This is where the same operator has different meanings. This is because Python does not allow two methods to have the same name. Example; (The +operator invokes an \_add\_ in which it is defined in the operator).

# Python Program illustrate how

# To overload an binary + operator

Class A:

def \_\_init\_\_(self, a):

self.a = a

# adding two objects

def \_\_add\_\_(self, o):

return self.a + o.a

ob1 = A(1)

ob2 = A(2)

ob3 = A("Geeks")

ob4 = A("For")

print(ob1 + ob2)

print(ob3 + ob4)

OUTPUT:

3

GeeksFor

Advantage

One operator can have multiple capabilities thus it can be reused.

Summoning the operator saves time.

Disadvantage

Can be delimiting especially when the operator is used repeatedly thus it becomes very hectic to work with it.

STATIC TYPING AND DYNAMIC TYPING IN PYTHON

Python is a dynamically typed language which means checking of the variable is done at runtime, it is easier to write as initialization of variables is not done. Operations can be done based on the type of value assigned to it. In statically typed language checking of variables is done at the compile time. The Type () function identifies which object a variable currently references. In other languages such as C# variables are initialized. Example;

String message = 'Hello';

In Python dynamic typing :

Message = 'Hello'

EXAMPLE 2:

message = 100

print(type(message))

THE OUTPUT CHANGES AS A VALUE HAS BEEN ASSIGNED TO IT:

<class 'int'>

SIMILITUDE BETWEEN STATIC AND DYNAMIC TYPING

Both static and dynamic check for variables when a program is being runned.

CONTRAST BEWTEEN STATIC AND DYNAMIC TYPING

In static a variable is inspected at runtime while dynamic inspection is done during compile time.

CURIOSITIES OF PYTHON PROGRAMMING LANGUAGE

One line of code can return multiple values .

Codes are not written in braces unlike other languages but indentation is used.

If a string statement is separated by space, Python links them together to form one word.

Same values can be assigned to multiple variables in Python Language.

In Python there can be multiple operators to be compared unlike other languages used for programming.

Programs written in Python Language are Referenced. Objects passed by a function have to be referenced first.

Python supports only three programming paradigms which are Object Oriented Programming,Functional Programming Paradigm and Procedural Paradigm.

The high order functions enable Python to be able to arrange items to make a list which is in a particular order.

The use of few lines of codes makes writability and readability of codes easy for programmers and users.

Takes less time to write a program and understand.

An operator with many functions is only summoned during program execution.

Python supports three High Order functions which include Sorted Function which returns a list of items arranged in a specific order, Filter Function that returns a true value after program execution and Main Function.

Inappropriate indentation may result in errors.

Since Python Language does not use any compilers, errors tend to appear when the code is being executed or at the runtime.

Python has a high demand for memory as codes are solved in small chunks and the result is stored.

Since Python is an easy language, programmers become accustomed to it such that it may be frantic to work with a programming language such as C.

ADVANTAGES OF USING PYTHON LANGUAGE

It is easy to understand both for the interpreter and user.

Has libraries which make it easier for a programmer to use.

Can be used by any platform.

Can be acquired freely.

Use of paradigms enable problem solving in Python to be easy.

Can easily be deployed.

It is suitable for creating Graphical User Interfaces.

DISADVANTAGES OF USING PYTHON LANGUAGE

It is not suitable for mobile applications as it is not known to the operating systems thus acquiring it may be slow.

Errors occur only at runtime and not at compiling time.

A lot of memory has to be set aside when using Python.

May not be suitable for a program that should be executed at a faster speed.

Python does not interact with hardware therefore it is troublesome when accessing database files that are stored permanently.