**Python Notes**

######## Python Comments ####### (28-08-2024)

""" 1.Comments can be used to explain python code.

    2.Comments can be used to make code more readable.

"""

######## Python Variable #######

# Variable:

""" 1.Variables are containers for storing the data.

    2.Variable names must begin with a letter or an underscore, but they can be a group of both letters and digits.

"""

#############################"Boolean"###############################

#Boolean:

'''Booleans represent one of two values: True or False.'''

#Boolean Values:

'''

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer

'''

print(12 >  2)

print(3 < 1)

print(4 == 2)

'''When you run a condition in an if statement, Python returns True or False:'''

a = 100

b = 200

if a > b :

    print("Yes a is greater than b")

else:

    print("No a is not greater than b")

   # Evaluate Values and Variables:

'''The bool() function allows you to evaluate any value, and give you True or False in return,'''

print(bool("Bhargav"))

print(bool(15))

'''One more value, or object in this case, evaluates to False, and that is if you have an object that is made from a class with a \_\_len\_\_ function that returns 0 or False:

'''

class myclass():

  def \_\_len\_\_(self):

    return 0

myobj = myclass()

print(bool(myobj))

'''Functions can return boolean value'''

def my\_function()

 return True

print(my\_function())

#####

def my():

   return False

if my():

   print("yes")

else:

   print("No")

"""Python also has many built-in functions that return a boolean value, like the isinstance() function,

which can be used to determine if an object is of a certain data type:"""

x = 100

print(isinstance(x, int))

**29-8-2024**

#Python Classes and Objects:

'''

\*.Python is an object oriented programming language.

\*.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.

'''

#Create a Class:

'''To create a class, use the keyword class'''

class hello:

print("hi there")

#create an Object:

'''Now we can use the class named MyClass to create objects'''

class my\_class:

    x = 5

p1 = my\_class()

print(p1.x)

#The \_\_init\_\_() Function:

'''

\*.To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.

\*.All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.

\*.Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created

'''

class Person:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

p1 = Person("John", 36)

print(p1.name)

print(p1.age)

print(p1.name, p1.age)

###############

class Name:

   def \_\_init\_\_(self, fname, age):

      self.fname = fname

      self.age = age

x = Name("Bhargav", 24)

print(x.fname)

print(x.age)

   #Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

#The \_\_str\_\_() Function:

'''\*.The \_\_str\_\_() function controls what should be returned when the class object is represented as a string.

\*.If the \_\_str\_\_() function is not set, the string representation of the object is returned.

'''

class Person:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def \_\_str\_\_(self):

    return f"{self.name}({self.age})"

p1 = Person("John", 36)

print(p1)

#Object Method:

'''Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class'''

class Person:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def myfunc(self):

    print("Hello my name is " + self.name)

p1 = Person("John", 36)

p1.myfunc()

   #Note: The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

class Details:

   def \_\_init\_\_(self, name, age):

      self.name = name

      self.age = age

   def info(self):

      print("hello my name is " + self.name)

x1 = Details("bhargav",24)

x1.info()

#Modify Object Properties:

class Person:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def myfunc(self):

    print("Hello my name is " + self.name)

p1 = Person("John", 36)

p1.age = 40

print(p1.age)

#delete the object;

'''You can delete properties on objects by using the del keyword'''

#class Person:

  #def \_\_init\_\_(self, name, age):

    #self.name = name

    #self.age = age

  #def myfunc(self):

    #print("Hello my name is " + self.name)

#p1 = Person("John", 36)

#del p1.age

#print(p1.age)

#delete object:

'''You can delete objects by using the del keyword.'''

'''class Person:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def myfunc(self):

    print("Hello my name is " + self.name)

p1 = Person("John", 36)

del p1

print(p1)'''

#The Pass Statement:

'''class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.'''

class Person:

  pass