**G1\_comment\_variable**

# *python comment*

# *using # sign / ''' ''' single quotation / " " " " " " double quotation*

''' hello python . this line dont show '''

""" this is the another way of comment """

print("comment")

# *above of the code only show the print command.*

# *======== Variable and their type ======== #*

a = " shuvro baset" #*string type data*

b = ' habijabi' #*string type data*

c = 75 # *int type data*

d = 4.75 # *float type data*

e = True # *boolean type data*

print(a,type(a))

print(b,type(b))

print(c,type(c))

print(d,type(d))

print(e,type(e))

OUTPUT:

shuvro baset <class 'str'>

habijabi <class 'str'>

75 <class 'int'>

4.75 <class 'float'>

True <class 'bool'>

**G2\_operator**

# *======= Operators of python ============== #*

x = 5

y = 6

# *now i show the different types of arithmetic operator and there operations*

sum = x + y

min = x - y

mult = x \* y

dev = x / y

print(sum)

print(min)

print(mult)

print(dev)

# *when we need the reminder (vag sas) using modulous(%) operator*

rmnd = y % x

print(rmnd)

# *without float result / floor devision(//)*

flrdvsn = 10//3

print(flrdvsn)

# *power operator power (\*\*) exponential*

pwr = x\*\*y

print(pwr)

# *2nd way of exponential(power)*

k = 2

res = pow (k,3)

print(res)

a = 75

a = -75

print(a)

print(abs(a))# *here we can see that, negative value change into positive value*

# *print devision value and reminder both // in this case we use (divmod) function*

s = 75

p = 10

r = divmod(s,p)

print(r)

OUTPUT:

11

-1

30

0.8333333333333334

1

3

15625

8

-75

75

(7, 5)

**G3\_string**

#*===== declaring a string value and print their elements =====#*

str = 'python Course'

print(str)

print(str[0],str[2],str[-1], str[-3])

OUTPUT

python Course

p t e r

name = 'shuvro baset'

print ( name.title() ) # *showing value as a title case*

print ( name.upper() ) # *showing value uppercase*

print ( name.lower() ) # *showing value lower case*

print ( name.upper().lower().title() ) # *3 operation finally last one is shown*

OUTPUT

Shuvro Baset

SHUVRO BASET

shuvro baset

Shuvro Baset

# *== string concatenation ======= #*

frst\_name = ' Abdul'

last\_name = 'baset'

name1 = frst\_name.lstrip() + ' ' + last\_name# *space hiding left strip by calling lstrip function ........./rstrip is for hide right side of space*

print(name1)

# *if I use only strip then, all the spaces will be hide*

hidename = ' s h u v r o '

hide = hidename.strip()

print(hide)

#*=====Note : it will hide only first and last spaces, not inner spaces*

#*======== use quotation in string value ========*

abs = "hi! are u understand it. won't u ? "

sba = 'hi! are u understand it. won\'t u ? '

print(abs)

# *note : use different type of quotation for this*

#*========= Find word in sentence ====== #*

sntnce = "check my word from here using find function"

print( sntnce.find('word'))# *in this method it will show the index number . if there is no word then it will show (-1)*

#*===== replacing word using replace function ====*

sntnce = sntnce.replace('check', 'finding')

print(sntnce)

# *note : if there is multiple same word and i wanna change every same word then we have to use replaceAll function*

# *print multiple value with seperate them*

print("ami ", " baset",sep=".")

x = 'shuvro'

y = 'baset'

z = 'python programmer'

print(x,y,z, sep='| ' )

#*string interpolation*

me = "{nam}'s age is {age}"

print(me.format(nam='baset', age=27))

me = '%s\'s age is %d'

print (me % ('baset',27))

#*slice*

slce = "shuvo ratri"

pushpo = slce[6:]

print(pushpo)

#*[start : end : step]*

t = [1,5,3,4,6,4,6,54,6]

print(t[0:5:1])# *hehe elements are showing one by one, but only first 5 elements shown here.*

print(t[::2])# *in this case it will the elements after skipping two elements*

#

ch = input("enter your letter")

if ch.islower():

print("the enter letter is lowercase")

else:

print("the enter letter is uppercase")

#*========== declare a list with 3rd brackets*

list1 = ['baset', 'shuvro', 'writer']

print(list1)

print(list1[1])

#*============= mixed list (with defferent type of data)*

list2 = ['shuvro', 75, 4.1, 23, False]

print(list2)

print(list2[-1])

#*================== Empty list*

list3 = []

mat = [[1,2,3,4],[4,5,8,6],[5,6,8,3]]

for x in mat:

for y in x:

print(y, end = ' ')# *stop printing in new line .*

print()

# *==============list iteration by for loop*

car = ['bike','r15','xixer',]

for x in car:

print(x)

if x == 'r15':

print("r15 is the best bike")

#*============== sum of the list numbers*

sum = 0

list3 = [1,5,7,9,6,4]

for y in list3:

sum += y

print("sum is : {}".format(sum))

# *=============sum ignore string value*

sum = 0

list3 = [1,5,7,'asd',6,4]

for y in list3:

if type(y) == int:

sum += y

print("sum is : {}".format(sum))

#*=================== list modification changing value*

list4 = ["shuvro", 75, 23, 4.75, False]

list4[0] = "baset"

print(list4)

# *============add item*

list4.append("hehe")

print(list4)

#*============== adding list specific position*

list4.insert(2, "add")

print(list4)

#*========== delete list item*

del list4[3]

print(list4)

#*============= delete last item*

last\_list4 = list4.pop()

print(list4, "\n", last\_list4)

#*=========== remove list item by value*

rmve = [1,2,3,4,5,6,8,5,6]

print(rmve)

rmve.remove(6)

print(rmve)# *remove 1st 6*

#*============= splitting string into list items*

import re

name = "baset, shuvro, dipto"

name\_list = re.split(',',name)

print(name\_list)

#*================= convert a list into string*

list2str = ['here', 'is', 'converting', 'string']

convrt\_str = ' '.join(list2str)

print(convrt\_str)

#*============= sorting ( alphabetically )*

list2str.sort()

print(list2str)

list2str.sort(reverse=True)

print(list2str)

# *================reverse list*

rmve.reverse()

print(rmve)

#*===========length of list*

print(len(rmve))

#*===================checking a value in the list*

rmve = [1,2,3,4,5,6,8,5,6]

if 1 in rmve:

print("yes it is")

# *tuple*

tp = (1,2,'tuple','string', 4.75, False)

print(tp,type(tp))

#*tuple access is just like as list access*

print(tp[2], tp[-1], tp[3])

# *iteration*

for t in tp:

print(t)

# *tuple is immutable (its unchangable)*

#*tp[0] = 3*

#*print(tp) # this code will not work. because tuple is unchangable*

# *Note : we can change the tuple fully . we cant change individual value of tuple*

# *unpacking or multiple value assignment from the tuple*

t = 1, 2, 3

x, y, z = t

print (x,y,z, sep = ' | ')

# *dictionary*

dict = {'zero' : '0', 'one' : '1', 'two' : 2}

print(dict)

print(dict['zero'])

# *length of dict*

print(len(dict))

#*======= Modification======*

dict['five'] = 5

print(dict) # *add*

del dict['five']

print(dict)

# *Iteration key and value*

dic = {'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4}

for key, value in dic.items(): # *====== if i use (.key) then it will show only key || if i use .values then it will show every values of keys*

print(key, value, sep='=')

# *sorted keys while iterate*

for key in sorted(dic.keys()):

print(key, dic[key])

# *==========condition*

num = 50

if num % 2 == 0:

print("even number")

#*====input function - even/ odd*

a = int(input("enter ur number: "))

if(a % 2 == 0):

print("the enter number is even")

else:

print("the enter number is odd")

# *if-elif-else chained condition*

num = int(input("enter a number: "))

if num == 50:

print("half century")

elif num ==100:

print("century")

elif num > 100:

print("century +")

else:

print("unknown number")

# *logical operator and/or*

nm = 1

if nm >=3 or nm > 0:

print("condition is true")

#*compare string*

name1 = "Shuvro"

name2 = "shuvro"

if name1 == name2: # *if name1.lower()== name2.lower() then name match*

print("same name")

else:

print("name doesn't match") # *case problem.*

# *not equals to*

name = "unknown person"

if name != "shuvro":

print(name)

#*## Loop #####*

#*while loop #*

x = 1

while x <=5:

print (x)

x += 1

#*=======infinity loop #*

#

#*======== break loop #*

# *b = 1*

# *while True:*

# *print(b)*

# *b += 1*

# *if b > 10:*

# *break*

# *continue loop #*

x = 0

while x < 20:

x += 1

if x % 2 == 0:

continue

print(x)

# *For loop #*

sum = 0

for num in range(1,11):

print(num)

sum += num

print("sum is {sum}".format(sum=sum))

# *for loop for string #*

name = "shuvro"

for char in name:

print(char)

# *welcome to function*

def welcome(name):

print("welcome {}".format(name))

welcome('shuvro')

welcome('baset')

# *positional argument*

def prsn\_detail(name,age,devision):

print(name,age,devision, sep='|')

prsn\_detail('baset',27,'dhaka')

prsn\_detail('ratri',23,'jessore')

# *argument order doesn't matter / keyword argument*

prsn\_detail('shuvro', age=27, devision = 'dhaka')

prsn\_detail(age = 27, name = 'shuvro', devision= 'dhaka')

# *Default value*

def cstmr\_details(name, age, country='bangladesh'): # *default value maintain sequence from last side*

print(name,age,country, sep = '|')

cstmr\_details('shuvro',25)

#*return value*

def square(num):

return num \* num

print(square(3),square(5.5), sep = '|')

# *name concate*

def name(first\_name, last\_name):

return first\_name + " " + last\_name

print( name('shuvro','baset'))

# *Optional argument*

def get\_name(first\_name, last\_name, middle\_name=''):

complete\_name = first\_name

if middle\_name:

complete\_name += ' ' + middle\_name

complete\_name += ' ' + last\_name

return complete\_name

print( get\_name('shuvro', 'baset'))

print( get\_name('shuvro','baset', 'rudro') )

# *reference type parameter*

nm = 50

def new\_num(nm):

nm += 25

print( "inner number : {}".format(nm))

new\_num(nm)

print( ' 1st\_number :' + str(nm))

# *list, dictionary (this reference type parameter will effect )*

num\_list = [1,2,3,4,5]

num\_dict = { 'one':1, 'two' : 2, 'three' : 3}

def change\_num\_list(list,dict):

del list[0]

list[-1] = 75

del dict['one']

dict['three'] = 23

print("Inner List : ", list)

print("Inner dict: ", dict)

print("before")

print(" Outer list: ", num\_list)

print( "Outer dict: ", num\_dict)

change\_num\_list(list=num\_list, dict= num\_dict)

print("after")

print("outer List: ", num\_list)

print("outer List: ", num\_dict)

#*=== Arbitrary number of arguments (unknown parameter)*

def students(\*students\_name):

print(students\_name, type(students\_name))

for students in students\_name:

print (students)

students('shuvro', 'baset', 'rudro', 'dipto')

students()

students('pushpo')

# *Positional and arbitrary arguments mixing*

def team(cap, \*member):

print('captain', cap)

print('others', member)

team('shuvro', 'forhad', 'rubel', 'shemanto')

# *Arbitrary keyword arguments (\*\* dictionary )*

def team2(cap,\*\*others):

print('captain',cap)

print('other', others)

team2(cap= 'shuvro', sec\_cap = 'baset', third\_cap = 'shemanto')

#*====== LAMBDA ======= #*

# *anonymous or Inline function*

add\_numbers = lambda a,b : a + b

print( add\_numbers(2,3))

man = lambda name: "we love: " + name

print(man('bangladesh'))

#*===== Class variable and Instance Variable details*

class prj:

st = 70 #*class variable*

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

self.name = name#*Instance variable*

# *Instantiation*

prj1 = prj('shuvro')

prj2 = prj('baset')

print(prj1.name, prj2.name)# *Accessing Instance variable*

print(prj1.st, prj2.st)# *Accessing class variable*

#*====== changing class variable*

prj.st = 75# *change value of class variable/ parmenantly change*

print(prj.st, prj2.st)

prj1.\_\_class\_\_.st = 7# *this is the best way to change a class variable fixed*

print(prj1.st, prj2.st)

#*==========value change using object bt .......*

# *if we change variable by calling object then it will change only that onject.*

# *bt actual class variable as lke as same*

prj1.st = 23

print(prj1.st, prj2.st)

#*==========8.2===========#*

# *---------- ('==' vs 'is' operator)*

x = [1,2,3]

xx = x

print(x == x)

print(x is xx)

y = list(x)

print(x == y)

print(x is y)

#*====== 8.2======*

#*====== OOP class to string =====*

class prsn:

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

self.name = name

def \_\_str\_\_(self):

return f'{self.\_\_class\_\_.\_\_name\_\_} class, obj name: {self.name}'

p1 = prsn("shuvro")

p2 = prsn("baset")

print(p1)

print(p2)

class Coachng:

st\_name = '' # *global variable/ class variable*

st\_class = ''

def details(self):

print(self.st\_name, self.st\_class)

def details\_with\_address(self, address):

print(self.st\_name, self.st\_class)

print(address)

c = Coachng()

c.details

c.st\_name = "shuvro"

c.st\_class = "hons"

c.details

c.details\_with\_address("dhaka")

print(type(c))

# *Contrusctor method*

class Person:

def \_\_init\_\_(self, name, age):# *1st method called autometically that is init method*

self.name = name

self.age = age

def details(self):

print(self.name, self.age, sep = '|')

shuvro = Person('shuvro', 27)

shuvro.details()

print( shuvro.name, shuvro.age)

#*creating more than 1 object*

class Person:

def \_\_init\_\_(self, name, age):# *1st method called autometically that is init method*

self.name = name

self.age = age

def details(self):

print(self.name, self.age, sep = '|')

people\_list = []#*??????????*

for x in range(0,3):

prsn = Person("Person " +str(x), 30+x)

people\_list.append(prsn)

#*people\_list += [prsn]*

for x in people\_list:

x.details()

# *print(people\_list[0:]) #?????????*

#*========= attribute value modification*

chnge = Person(name='baset', age=25)

chnge.details()

chnge.name="shuvro"

chnge.details()

#*========== Lifecycle ======*

class X:

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

self.name = name

print(self.name + " created")

def \_\_del\_\_(self):

print (self.name + " is destroyed ")

x = X('X')

y = X('Y')

print("hello world")

def hello():

x = X('hello \_ X')

y = X('hello \_y')

hello()

#*============ Inheritence =========*

class Math:

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

self.x = x

self.y = y

def sum(self):

return self.x + self.y

class MathExtended1(Math):

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

super().\_\_init\_\_(x,y)

def subtract(self):

return self.x - self.y

math\_ext\_obj = MathExtended1(10,5)

print( math\_ext\_obj.sum() )

#*=========== Multiple Inheritance ==========*

class MathExtra:

def division(self, x, y):#*===== without \_\_init\_\_ method*

return x / y

class MathExtended2(MathExtended1, MathExtra): #*child class*

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

super().\_\_init\_\_(x, y)

def multiplication(self):

return self.x \* self.y

math\_ext2 = MathExtended2(10,2)

print("Sum", str(math\_ext2.sum()) )

print( "Subtract", str(math\_ext2.subtract()))

print ("muliplication", str(math\_ext2.multiplication()))

print( "division", str(math\_ext2.division(x=math\_ext2.x, y = math\_ext2.y)))

#*print( "division", str(math\_ext2.division(x = 10, y = 5)))#= no \_\_init\_\_ method . so we have to call a keyword argument*

#*=======Method Override=============*

class Math:

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

self.x = x

self.y = y

def sum(self):

return self.x + self.y

def sum(self):#*Method Override*

return self.x + self.y + 75

def show\_all(self):

print("sum of child class: "+ str(super().sum()))

print("sum of override : "+ str(self.sum()))

#*its not finish yet. I have to do something here.*

#*======= Model ====== Namespace ========== Package========*

#*======= Namespace=========*

#*=========== Import =========*

import G7\_function\_lambda

print( G7\_function\_lambda.square(7))# *??????????? why all function comes here.*

# *Import specific functions*

from G7\_function\_lambda import new\_num

print(new\_num(75))