**HW Assignment#1**

1.

* True and 1 / 0 and False

# True and anything, so the part after and should be analysed, 1/0 gives zero by division error, error and anything results in error.

Output: Zero by division error

* True or 1 / 0 or False

True or anything results in True as for or any one operand is true, the output is true. So, no need to analyse the second part of the statement.

Output:True

* True and 0

Anything and with 0 results in 0.

Output:0

* False or 1

Anything or with 1 always results in 1.

Output:1

* 1 and 3 and 6 and 10 and 15

As all the integers represent True value, the highest among them returns for an and operation.

Output:15

* 0 or False or 2 or 1 / 0

0 or False or 2 or 1/0 ->False or 2 or 1/0 -> 2 or 1/0 -> 2 as 2 is true and or operation does not consider the next part.

Output:2

2.

def welcome():

print('Welcome to', end=’ ’)

return 'Hello'

def IT310():

print('IT310')

return 'Python'

print(welcome(), IT310())

# First welcome function is called, so ‘Welcome to’ is printed and as end=””, the next print statement from the function IT310() prints “IT310”.

Now in the next line the return values of both the functions are printed.

Output:

Welcome to IT310

Hello Python

3.

x = 6

def foo1(x):

print(x)

def foo2(x):

y = x

x = 7

print(x)

y = foo1(x) # prints x value from the function foo1(x=6)

foo2(x) # y is assigned to value of x=6, y=6 and x is assigned to 7(prints 7)

y + foo1(8) # prints 8, y+return value of foo1(8)= 6+NONE=different data types cannot be added

Output:

6

7

8

Integer and string Cannot be added

4.

from operator import sub, mul

def print\_sub(x, y):

print('sub')

return sub(x, y)

def print\_mul(x, y):

print('mul')

return mul(x, y)

print\_sub(print\_mul(4, 504), 2)

# print\_mul function prints “mul” and returns mul(x,y)= 4\*504 = 2016

# print\_sub function print “sub” and returns sub(x,y)=sub(2016,2)=2014

a = print\_sub(print\_mul(4, 504), 2)

# as the same statement is assigned to a, the same output appears and the value of a is the return value of function print\_sub which is sub(2016,2)=2014. So we get 2014 as output

Output:

mul

sub

mul

sub

2014

5.

def sum\_odd(n):

sum=0

for i in range (1,n+1):# to make sure the number is less than n

if(i%2!=0): # to consider only odd numbers

sum=sum+i

return sum

sum\_odd(6) #1+3+5

sum\_odd(7) #1+3+5+7

Output:

9

16

6.

def foo(a,b,c,d):

list1=[a,b,c,d] # making list of given numbers

x=min(list1) # finding minimum of the given numbers and assigning to x

list1.remove(x) # removing the minimum value from list

y=min(list1) # finding minimum value from the list remaining

# this will be the second lowest number from the

# original list

return x\*\*2+y\*\*2 # returning sum of squares of two lowest numbers

x=foo(1,2,3,4)

print(x)

y=foo(-3,1,5,6)

print(y)

Output:

5

10

7.

def df(x, y, z):

if x-y==z or y-x==z:

return True

elif y-z==x or z-y==x:

return True

elif x-z==y or z-x==y:

return True

else:

return False

df(5, 3, 2) # 5 - 3 is 2

#returns True

df(2, 3, 5) # 2-3!=5 or 3-2!=5

#3-5!=2 or 5-3=2

# returns True

df(2, 5, 3) # 2-5=!3 or 5-2=3

# returns True

df(-2, 3, 5) # -2-3!=5 or 3-(-2)=5

# return True

df(-5, -3, -2) # -5-(-3)=-2

# returns True

df(-2, 3, -5) # -2-3=-5

# returns True

df(2, 3, -5) # 2-3!=-5 or 3-2!=-5

# 3-(-5)!=2 or -5-3!=2

#-5-2!=3 or 2-(-5)!=3

# returns False

df(10, 6, 4) # 10-6=4

# returns True

df(10, 6, 3) # 10-6!=3 or 6-10!=3

# 6-3!=10 or 3-6!=10

# 3-10!=6 or 10-3!=6

# returns False