INTERNSHIP TASKS

Name : S. Deva Manikanta

Clg Id : 12119003

Course : Python

Org : IGIAT – VSKP

Date : 12-04-2024

Day 13: EXERCISE – 13

Exercise Level 1

#Task 1:

#Filter only negative and zero in the list using list comprehension

def only\_postives(list\_of\_values):

    updated\_list = [value for value in list\_of\_values if value != 0 and value > 0]

    return updated\_list*;*

print("Task 1: List with out negative and zero values --> RESULT AS FOLLOWS")*;*

print("List 1 -- [-4, -3, -2, -1, 0, 2, 4, 6]: ", only\_postives([-4, -3, -2, -1, 0, 2, 4, 6]))*;*

print("List 2 -- [0, -1, 2, 0, 4, 5, -2]: ", only\_postives([0, -1, 2, 0, 4, 5, -2]))*;*

#Task 2:

#Flatten the following list of lists of lists to one dimensional list;

list\_of\_lists = [[[1, 2, 3]], [[4, 5, 6]], [[7, 8, 9]]]

list\_of\_values = [list\_item\_of\_list for list in list\_of\_lists for list\_item in list for list\_item\_of\_list in list\_item]*;*

print("Task 2: [[[1, 2, 3]], [[4, 5, 6]], [[7, 8, 9]]] --(flatten)-->",list\_of\_values)*;*

#Task 3:

#Using list comprehension create the following list of tuples in 13.List Comprehension.pptx

output\_task\_3 = [(i, i\*\*0, i\*\*1, i\*\*2, i\*\*3, i\*\*4, i\*\*5) for i in range(0, 11)]*;*

print("Task 3: Output As follows\n")*;*

print("[")

for i in output\_task\_3:

    print("  ",i)*;*

print("]")*;*

#Task 4:

#Flatten the following list to a new list

# [[('Finland', 'Helsinki')], [('Sweden', 'Stockholm')], [('Norway', 'Oslo')]]

list\_of\_countries = [[('Finland', 'Helsinki')], [('Sweden', 'Stockholm')], [('Norway', 'Oslo')]]*;*

country\_codes = {

    'Finland' : 'FIN', 'Sweden' : 'SWE', 'Norway' : 'NOR'

}

output\_task\_4 = [[country.upper(), country\_codes[country], location.upper()] for [(country, location)] in list\_of\_countries]*;*

print("\nTask 4: Result as follows:\n", output\_task\_4)*;*

#Task 5:

# Change the list of tuples to a dictionary

list\_of\_countries = [[('Finland', 'Helsinki')], [('Sweden', 'Stockholm')], [('Norway', 'Oslo')]]*;*

output\_task\_5 = [{'country': country\_name, 'city' : city\_name} for [(country\_name, city\_name)] in list\_of\_countries]

print("\nTask 5: Result as follows:")*;*

print(output\_task\_5)*;*

#Task 6:

# Change the list of lists to a list of concatenated strings

names = [[('Asabeneh', 'Yetayeh')], [('David', 'Smith')], [('Donald', 'Trump')], [('Bill', 'Gates')]]

output\_task\_6 = [first\_name+" "+last\_name for [(first\_name, last\_name)] in names]*;*

print("\nTask 6: Output as follows")*;*

print(output\_task\_6)*;*

#Task 7:

#Write a lambda function which can solve a slope or y-intercept of linear functions

equation = "y = 2x + 3"

def slope\_intercept\_finder(equation):

    equation = equation.replace(" ", '')*;*

    equation\_parts = equation.split("=")

    slope\_y\_intercept\_finder = lambda equation : {'slope(m)' : int(equation[1][0:equation[1].index('x')].strip()), 'y-intercept(b)' : int(equation[1][equation[1].index('x')+1:].strip())}

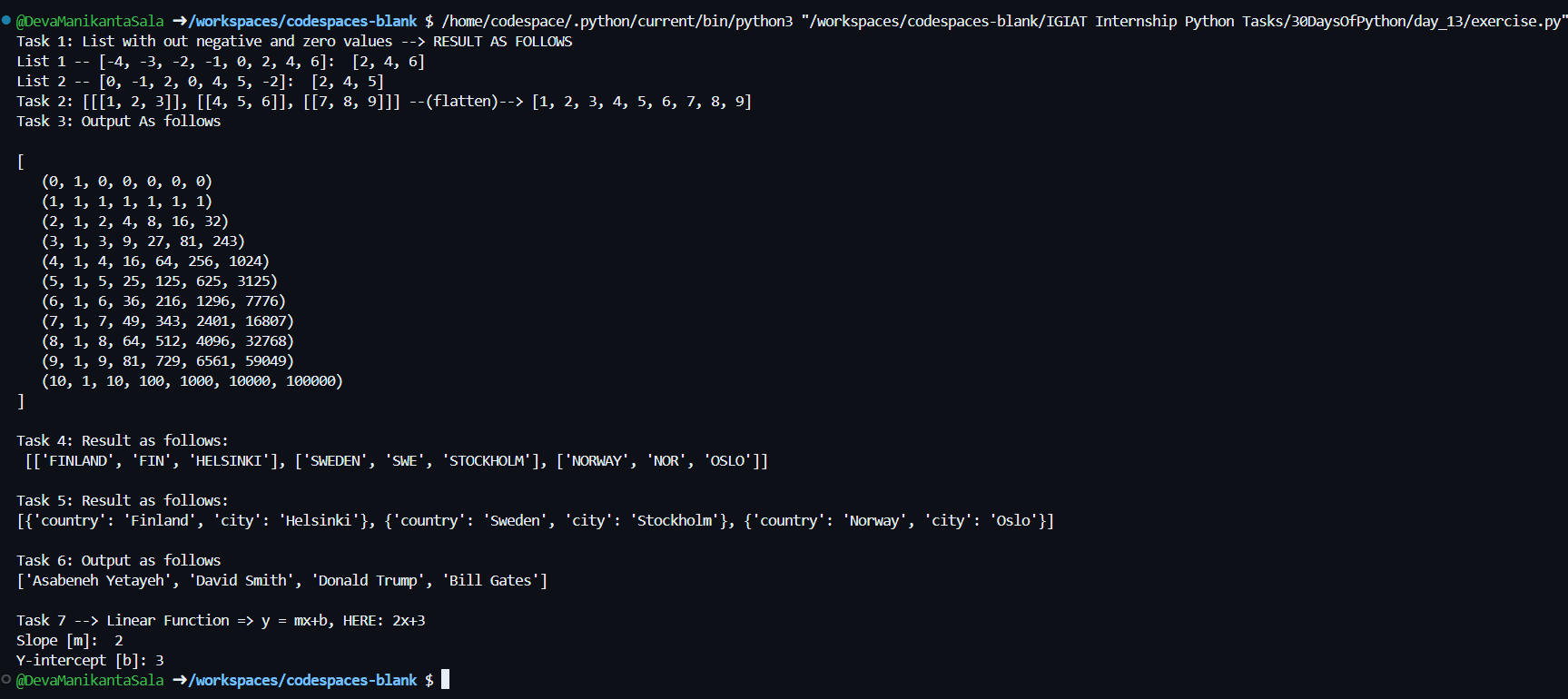
    return slope\_y\_intercept\_finder(equation\_parts)*;*

output\_task\_7 = slope\_intercept\_finder("y = 2x + 3")*;*

print("\nTask 7 --> Linear Function => y = mx+b, HERE: 2x+3")

print("Slope [m]: ", output\_task\_7['slope(m)'], "\nY-intercept [b]:", output\_task\_7['y-intercept(b)'])

**Outputs:**

****