INTERNSHIP TASKS

Name : S. Deva Manikanta

Clg Id : 12119003

Course : Python

Org : IGIAT – VSKP

Date : 12-04-2024

Day 11: EXERCISE – 11

Exercise Level 1

#Task 1:

#Declare a function add\_two\_numbers. It should take two parameters and it returns a sum

def add\_two\_numbers(a, b):

    return (int(a)+int(b))*;*

print("Task 1: Sum of 1 and 2 : ", add\_two\_numbers(1, 2))*;*

#Task 2:

#Declare the area of circle is calculated as follows: area = π \* (r^2). Write a function that calculates area\_of\_circle

import math as m;

def area\_of\_circle(radius):

    return (m.pi\*(float(radius)\*\*2))*;*

print("Task 2: Radius 5: Area of Circle is : ", area\_of\_circle(5.0))*;*

#Task 3:

#Write a function called add\_all\_nums which takes arbitrary number of arguments and sums all the arguments. Check if all the list items are number. If not do give a reasonable feeback.

def add\_all\_nums(\*numbers):

    all\_are\_same\_type = all(isinstance(number, int) for number in numbers)*;*

    if(all\_are\_same\_type):

        return sum(numbers)*;*

    else:

        return "All values are not same type"*;*

print("Task 3: Sum of 1, 2, 3: ", add\_all\_nums(1, 2, 3))*;*

print("Task 3: Sum of 1, \'Deva\', 3: ", add\_all\_nums(1, 'Deva', 3))*;*

#Task 4:

#Write a function that converts °C to °F like convert\_celsius\_to\_fahrenheit

def convert\_celcius\_to\_fahrenheit(degrees):

    return float(degrees \* (9/5) + 32)*;*

print("Task 4: 354°C in °F : ", convert\_celcius\_to\_fahrenheit(354))*;*

#Task 5:

#Write a function called check\_season, it takes a month parameter and returns the season: Autumn, Winter, Spring, or Summer

def check\_season(month):

    Summer = ['january', 'february', 'december']

    Autumn = ['september', 'october', 'november']

    Winter = ['march', 'april', 'may']

    Spring = ['june', 'july', 'august']

    month = str(month).lower()*;*

    if(month in Summer):

        return "Summer"*;*

    elif(month in Autumn):

        return "Autumn"*;*

    elif(month in Winter):

        return "Winter"*;*

    elif(month in Spring):

        return "Spring"*;*

    else:

        return "Invalid Month"*;*

print("Task 5: January is in", check\_season("January"), "season")*;*

#Task 6:

#Write a function called calculate\_slope which return the slope of a linear equation.

def calculate\_slope(equation):

    # linear equation -> y = mx + b;

        #eq -> y = mx + b

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

    parts = equation.split("=")*;*

    if(parts[0].strip() != 'y' or len(parts) != 2):

        return "Invalid Linear Equation : Expected format -- y = mx + b"*;*

    parts[1] = parts[1].strip()*;*

    index\_of\_x = parts[1].index('x')*;*

    slope = parts[1][0:index\_of\_x]*;*

    try:

        slope = float(slope)*;*

        return slope*;*

    except ValueError as ve:

        return "Invalid Slope value is given! Try again!"*;*

print("Task 6: The slope of linear equation (y = -532x + 67): ", calculate\_slope("y = -532x + 67"))*;*

#Task 7:

#Quadratic equation is calculated as follows: ax² + bx + c = 0. Write a function which calculates solution set of a quadratic equation, solve\_quadratic\_eqn.

def solve\_quadratic\_eqn(equation):

    try:

        parts = equation.split("=")

        j = 0*;*

        for i in parts:

            parts[j] = i.strip()

            j += 1*;*

        if (int(parts[1]) != 0 and int(parts[1]) > 0):

            parts[0] += ' - ' + parts[1]*;*

            parts[1] = '0'*;*

        elif (int(parts[1]) != 0 and int(parts[1]) < 0):

            parts[0] += ' + ' + parts[1][1]*;*

            parts[1] = '0'*;*

        else:

            pass*;*

        parts[0] = parts[0].replace(" ", '')*;*

        a = 0*;*

        if(parts[0][0:parts[0].index('x^2')] == '-'):

            a = -1*;*

        elif(parts[0][0:parts[0].index('x^2')] == ''):

            a = +1*;*

        elif (parts[0][0:parts[0].index('x^2')] != '-') and (parts[0][0:parts[0].index('x^2')] != ''):

            a = int(parts[0][0:parts[0].index('x^2')])

        else:

            a = 0*;*

        b = 0*;*

        if(parts[0][(parts[0].index('^2')+1)+1 : parts[0].rindex('x')]) == '-':

            b = -1*;*

        elif(parts[0][(parts[0].index('^2')+1)+1 : parts[0].rindex('x')]) == '+':

            b = +1

        elif(parts[0][(parts[0].index('^2')+1)+1 : parts[0].rindex('x')]) != '-' or (parts[0][(parts[0].index('^2')+1)+1 : parts[0].rindex('x')]) != '+' :

            b = int(parts[0][(parts[0].index('^2')+1)+1: parts[0].rindex('x')])

        else:

            b = 0*;*

        # c = int(parts[0][parts[0].rindex('x')+1: ]);

        c = 0*;*

        if(parts[0][parts[0].rindex('x')+1: ] == ''):

            c = 0*;*

        else:

            c = int(parts[0][parts[0].rindex('x')+1: ])*;*

        import math as m;

        discriminant = (b\*\*2) - (4\*a\*c)*;*

        if(discriminant > 0):

            print("Real Roots".upper())*;*

            root\_1 = (-(float(b)) + m.sqrt(discriminant)) / (2.0\*a)

            root\_2 = (-(float(b)) - m.sqrt(discriminant)) / (2.0\*a)

            print("Equation : ", equation)*;*

            print("|a =", a , "|b =", b, "|c =", c, "|")*;*

            print("Solution Set : {", root\_1, ",", root\_2, "}")*;*

        elif(discriminant == 0):

            print("Equal Roots".upper())*;*

            roots = -float(b)/(2.0\*a)*;*

            print("Equation : ", equation)*;*

            print("|a =", a , "|b =", b, "|c =", c, "|")*;*

            print("Solution Set : {", roots, ",", roots, "}")*;*

        else:

            print("Imaginary Roots".upper())*;*

            print("Equation : ", equation)*;*

            print("|a =", a , "|b =", b, "|c =", c, "|")*;*

            print("Expensive Computation")*;*

    except Exception as E:

        print("Error Occured : " , E, "\nOr else please enter the equation in format : ax^2 + bx + c = 0/or/ax^2 + bx = c");

print("Task 7 --> RESULT is as follows: ")*;*

solve\_quadratic\_eqn("2x^2 - 4x = 9")*;*

#Task 8:

#Declare a function named print\_list. It takes a list as a parameter and it prints out each element of the list.

def print\_list(list\_of\_items):

    j = 1*;*

    for i in list\_of\_items:

        print("Item -", j , ": ", i)*;*

        j += 1*;*

print("Task 8 --> RESULT is as follows")*;*

print\_list(['Hi', 'Hello', 'Vanakam', 'Adabhbarsey', 'Namasthe', 'Ciao'])*;*

#Task 9:

#Declare a function named reverse\_list. It takes an array as a parameter and it should return the reverse of the array (use loop).

def reverse\_list(list\_of\_items):

    reversed\_list = []*;*

    for i in range(len(list\_of\_items)-1, -1, -1):

        reversed\_list.append(list\_of\_items[i])*;*

    return reversed\_list*;*

print("Task 9 : Actual list --> [1, 2, 3, 4, 5], Reversed List --> ", end = "")*;*

print(reverse\_list([1,2,3,4,5]))*;*

#Task 10:

def capitalize\_list\_items(list\_of\_items):

    new\_list = []

    for i in list\_of\_items:

        new\_list.append(i.capitalize())*;*

    return new\_list*;*

print("Task 10 : Actual list --> ['ab cd', 'ef gh', 'ij kl'], Reversed List --> ", end = "")*;*

print(capitalize\_list\_items(['ab cd', 'ef gh', 'ij kl']))*;*

#Task 11:

#Declare a function named add\_item. It takes a list and an item as parameter. It returns a list with the item added at the end.

def add\_item(list\_of\_items, item):

    list\_of\_items.append(item)*;*

    return list\_of\_items*;*

print("Task 11: Actual List --> [1, 2, 3, 4, 5] Add:6, Added List --> ", end="")*;*

print(add\_item([1,2,3,4,5], 6))*;*

#Task 12:

#Declare a function named remove\_item. It takes a list and an item as paramter. It returns a list with the item removed at the end.

def removed\_item(list\_of\_items, item):

    list\_of\_items.remove(item)*;*

    return list\_of\_items*;*

print("Task 12: Actual List --> [1, 2, 3, 4, 5] Remove:4, Removed List --> ", end="")*;*

print(removed\_item([1,2,3,4,5], 4))*;*

#Task 13:

def sum\_of\_numbers(upto\_number):

    sum = 0*;*

    for i in range(upto\_number+1):

        sum += i*;*

    return sum*;*

print("Task 13: Sum of numbers from 1 to 5:", sum\_of\_numbers(5))*;*

#Task 14:

#Declare a function named sum\_of\_odds. It takes a number parameter and it adds all the odd numbers in that range.

def sum\_of\_odds(upto\_number):

    sum = 0*;*

    for i in range(upto\_number+1):

        if i%2 != 0:

            sum += i*;*

    return sum*;*

print("Task 14: Sum of odd numbers from 1 to 5:", sum\_of\_odds(5))*;*

#Task 15:

#Declare a function named sum\_of\_evens. It takes a number parameter and it adds all the even numbers in that range.

def sum\_of\_evens(upto\_number):

    sum = 0

    for i in range(upto\_number+1):

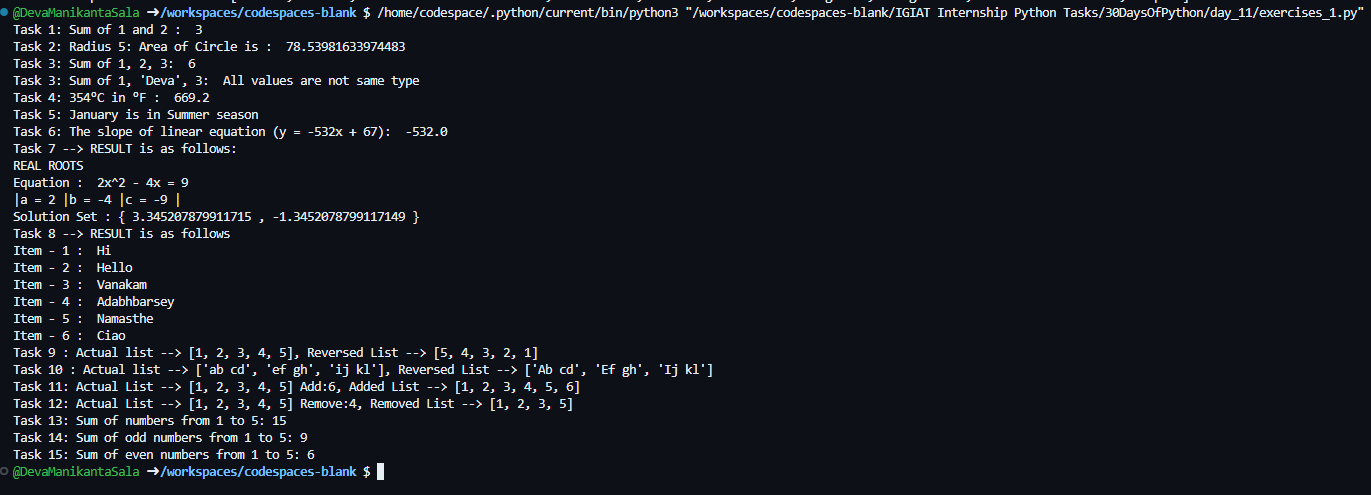
        if(i%2 == 0):

            sum += i*;*

    return sum*;*

print("Task 15: Sum of even numbers from 1 to 5:", sum\_of\_evens(5))*;*

**Outputs:**

****

Exercise Level 2

#Task 1:

#Declare a function named evens\_and\_odds. It takes a positive integer as a parameter and it counts number of even and odds from 1 to number.

def evens\_and\_odds(number):

    count\_even, count\_odd = 0, 0

    for i in range(number+1):

        if(i % 2 == 0):

            count\_even += 1*;*

        else:

            count\_odd += 1*;*

    return str("Even :{" + str(count\_even) + "}" + " Odd :{" + str(count\_odd) + "}")*;*

print("Task 1: Count Evens and Odds from 1 to 100 : ", evens\_and\_odds(100))*;*

#Task 2:

#Call your function factorial, it takes a whole number as a parameter and it return factorial of the number

def factorial(number):

    if (number == 1):

        return 1*;*

    elif (number == 0):

        return 0*;*

    else:

        return number \* (factorial(number-1))*;*

print("Task 2: The factorial of 5 is:", factorial(5))*;*

#Task 3:

#Call your function is\_empty, it takes a parameter and it checks if it is empty or not

def is\_empty(variable):

    if variable is None or len(variable) == 0:

        return True*;*

    else:

        return False*;*

print("Task 3: A --> [1,2] is empty? :", is\_empty([1,2]))*;*

print("Task 3: A --> [] is empty? :", is\_empty([]))*;*

#Task 4:

''' Write different functions which take lists.

    They should calculate\_mean, calculate\_median, calculate\_mode, calculate\_range, calculate\_variance, calculate\_std (standard deviation).

'''

def calculate\_mean(values\_list):

    mean = sum(values\_list)/len(values\_list)*;*

    return mean*;*

def calculate\_median(values\_list):

    values\_list.sort()*;*

    median = 0*;*

    if (len(values\_list) % 2 == 1):

        median = values\_list[len(values\_list)//2]*;*

    else:

        median = (values\_list[len(values\_list)//2 - 1] + values\_list[len(values\_list)//2]) /2*;*

    return median*;*

def calculate\_mode(values\_list):

    counts = []*;*

    for i in values\_list:

        counts.append(values\_list.count(i))*;*

    counts.sort()*;*

    counts.reverse()*;*

    for i in values\_list:

        if(counts[0] == values\_list.count(i)):

            return i*;*

def calculate\_range(values\_list):

    range = max(values\_list) - min(values\_list)*;*

    return range*;*

def calculate\_variance(values\_list):

    mean = calculate\_mean(values\_list)*;*

    variance = sum((x - mean)\*\*2 for x in values\_list) / len(values\_list)*;*

    return variance*;*

import math as m

def calculate\_std(values\_list):

    standard\_deviation = m.sqrt(calculate\_variance(values\_list))*;*

    return standard\_deviation*;*

print("Task 4:")*;*

print("Mean --> [1, 2, 3, 4, 5] :", calculate\_mean([1, 2, 3, 4, 5]))*;*

print("Median --> [1, 2, 3, 4, 5] : ", calculate\_median([1, 2, 3, 4, 5]))*;*

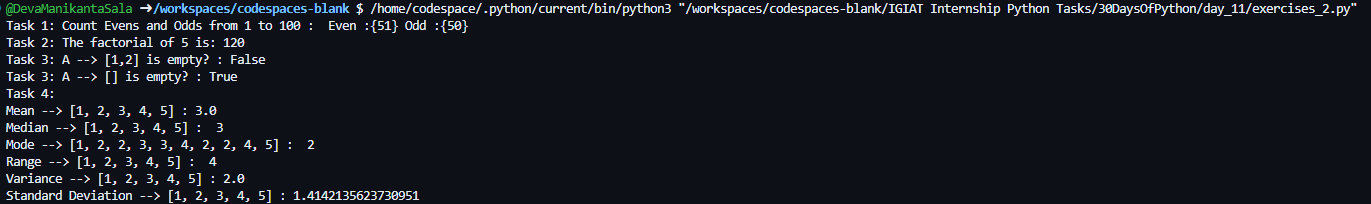
print("Mode --> [1, 2, 2, 3, 3, 4, 2, 2, 4, 5] : ", calculate\_mode([1, 2, 2, 3, 3, 4, 2, 2, 4, 5]))*;*

print("Range --> [1, 2, 3, 4, 5] : ", calculate\_range([1, 2, 3, 4, 5]))*;*

print("Variance --> [1, 2, 3, 4, 5] :", calculate\_variance([1, 2, 3, 4, 5]))*;*

print("Standard Deviation --> [1, 2, 3, 4, 5] :", calculate\_std([1, 2, 3, 4, 5]))*;*

**Output:**

****

Exercise Level 3

Step 1: Create a folder ‘data’ and download ‘countries.py’ click on [this](https://drive.google.com/uc?export=download&id=19FKmSln0zclQ97yUkFponmuWvyO4OGe9) link to download the file and paste it in ‘data’ folder.

*Try this link if it doesn’t download:*

[*https://drive.google.com/uc?export=download&id=19FKmSln0zclQ97yUkFponmuWvyO4OGe9*](https://drive.google.com/uc?export=download&id=19FKmSln0zclQ97yUkFponmuWvyO4OGe9)

Step 2: Now, download this file and paste it in ‘data’ folder ‘countries\_data.py’ click on [this](https://drive.google.com/uc?export=download&id=1aumrIQiumLBau9hTUPNEQBYgxZp-MwAo) link to download the file.

*Try this link if it doesn’t download:*

[*https://drive.google.com/uc?export=download&id=1aumrIQiumLBau9hTUPNEQBYgxZp-MwAo*](https://drive.google.com/uc?export=download&id=1aumrIQiumLBau9hTUPNEQBYgxZp-MwAo)

#Task 1:

#Write a function called is\_prime, which checks if a number is prime

def is\_prime(number):

    if(number < 2):

        return False*;*

    i = 2*;*

    import math as m;

    while(i <= int(m.sqrt(number))):

        if (number % i == 0):

            return False*;*

        i += 1*;*

    return True*;*

print("Task 1: Is 57 is a prime number -->", is\_prime(57))*;*

#Task 2:

#Write a function which checks if all items are unique in the list.

def is\_all\_items\_unique(values\_list):

    for i in values\_list:

        if (i in values\_list) and (values\_list.count(i) > 1):

            return False*;*

    return True*;*

print("Task 2: All items in this list [1, 6, 2, 3, 4, 7, 5] are unique -->", is\_all\_items\_unique([1, 6, 2, 3, 4, 7, 5]))*;*

print("Task 2: All items in this list [1, 1, 2, 2, 3, 4, 4] are unique -->", is\_all\_items\_unique([1, 1, 2, 2, 3, 4, 4]))*;*

#Task 3:

#Write a function which checks if all the items of the list are of the same data type.

def is\_all\_items\_same\_type(values\_list):

    for i in values\_list:

        if not isinstance(i, type(values\_list[0])):

            return False

            break*;*

    return True*;*

print("Task 3: Are the values are of same type in this [1, 2, 3, 4, 5, 6] list? -->", is\_all\_items\_same\_type([1, 2, 3, 4, 5, 6]))*;*

print("Task 3: Are the values are of same type in this [\'Deva\', \'Manikanta\', 12119003, 20] list? -->", is\_all\_items\_same\_type(['Deva', 'Manikanta', 12119003, 20]))*;*

#Task 4:

#Write a function which check if provided variable is a valid python variable.

def is\_identifier(string):

    if(string.isidentifier()):

        return True*;*

    else:

        return False*;*

print("Task 4: Is the identifier \'12DevaManikanta\' valid? -->", is\_identifier("12DevaManikanta"))*;*

print("Task 4: Is the identifier \'name\_of\_student\' valid? -->", is\_identifier("name\_of\_student"))*;*

#Task 5:

#A) Create a function called the most\_spoken\_languages in the world. It should return 10 or 20 most spoken languages in the world in descending order

from data import countries\_data as cd;

def most\_spoken\_languages():

    languages\_spoken = []

    for item in cd.countries\_data:

        for language in item['languages']:

            languages\_spoken.append(language)*;*

    unique\_languages = set(languages\_spoken)*;*

    counts\_of\_unique\_languages = {}

    for language in unique\_languages:

        counts\_of\_unique\_languages[language] = languages\_spoken.count(language)*;*

    counts = list(counts\_of\_unique\_languages.values())*;*

    counts.sort()*;*

    counts.reverse()*;*

    counts = counts[0:20]*;*

    i = 0*;*

    twenty\_most\_spoken\_languages = []

    while(i < len(counts)):

        for language, count in counts\_of\_unique\_languages.items():

            if(count == counts[i]):

                twenty\_most\_spoken\_languages.append(language)*;*

        i += 1*;*

    return twenty\_most\_spoken\_languages*;*

print("\n\nTask 5.A: Top 20 most spoken languages : \n", most\_spoken\_languages())*;*

#B) Create a function called the most\_populated\_countries. It should return 10 or 20 most populated countries in descending order.

from data import countries\_data as cd;

def most\_populated\_countries():

    population\_of\_countries = []

    for item in cd.countries\_data:

        population\_of\_countries.append(item['population'])

    population\_of\_countries.sort()*;*

    population\_of\_countries.reverse()*;*

    population\_of\_countries = population\_of\_countries[0:20]*;*

    i = 0*;*

    twenty\_most\_populated\_countries = []*;*

    while(i < len(population\_of\_countries)):

        for item in cd.countries\_data:

            if(item['population'] == population\_of\_countries[i]):

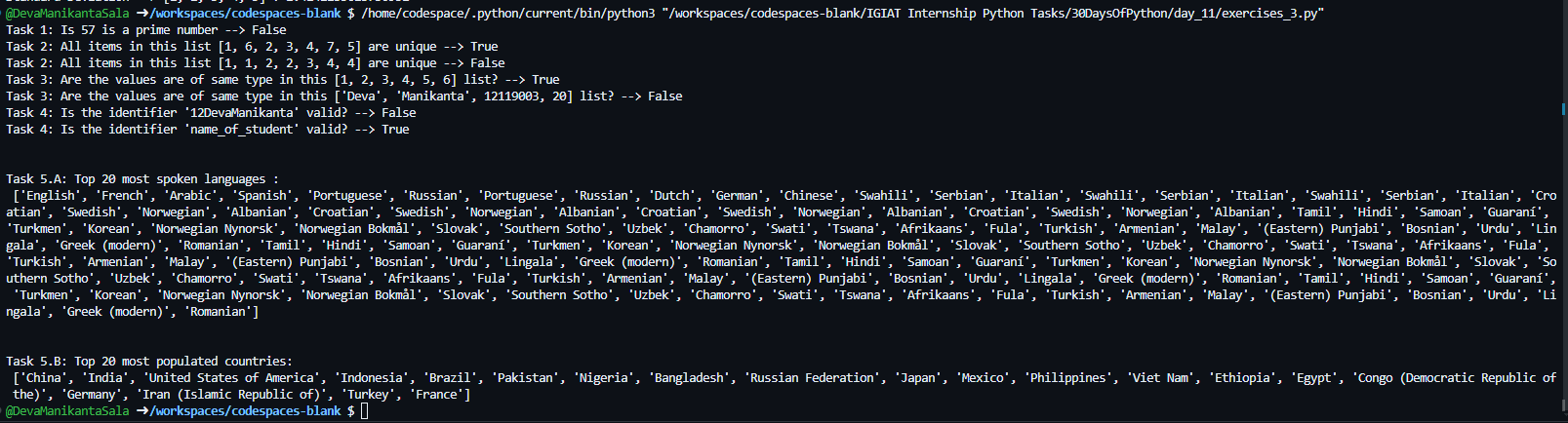
                twenty\_most\_populated\_countries.append(item['name'])*;*

        i += 1*;*

    return twenty\_most\_populated\_countries*;*

print("\n\nTask 5.B: Top 20 most populated countries:\n", most\_populated\_countries())*;*

**Output:**

****