***Please study the slides of CS1101 (first-year python lab) shared with you at WeLearn before trying to solve the following questions.**

**** You must try to execute the solutions of these questions in your Python3 IDEs/compilers.**

*****These are ungraded and are only for practice (you don't have to submit their solutions to WeLearn)**

****** The initial Sets are to help students brush up on their CS1101 knowledge.**

******* The next sets will contain some new concepts taught in CS2201 Theory/Tutorial classes (Wed)**

Set 1

Q1) Lists

(a) Create a list of integers from 0 to 9 and store the list in variable x

```
x = list(range(10))
```

(b) Create a list of integers from 3 to 12 and store the list in variable y

```
y=list(range(3,13,1))
```

(c) Using a single print command, print the list in x in reverse

```
print(x[-1:-len(x)-1:-1])
```

(d)Using a single print command, print the list of odd entries (starting from index 1) in x and then the list of even entries (starting from index 0) in x

```
print(x[1:10:2])
print(x[0:10:2])
```

(e) Check whether the 4th item of x (0th index element is the 1st element) is same as the 1st item of y by extracting those items

```
if (x[3]==y[0]):
        print ("True")
else:
        print ("False")
```

(f) Print the location of the number 10 is in the list x using index()

print (x.index(10))

(g) Print the location of the number 7 is in the list y using index()

print (y.index(7))

(h) Get a combined (appended) list of the items of x and y

print(list(x)+list(y))

(i) Create a combined list that consists of x and y. Find the location of the maximum and minimum numbers in this combined list (use max() and min()).

(x+y).index(max(x+y))

(x+y).index(min(x+y))

Q2) Strings are lists

(a) Store a string:“The quick brown fox jumps over the lazy dog” in a variable x

x="The quick brown fox jumps over the lazy dog"

(b) Check whether the word fox is in this sentence

```
x="The quick brown fox jumps over the lazy dog"

if 'fox' in x:
   print("Present")
```

(c) Print the sentence in reverse order

print (x[-1:-len(x)-1:-1])

(d) Print every third character of the above sentence (0th, 3rd, 6th….characters).

print (x[0:len(x):3])

(e) Print every fourth character of the above sentence (0th, 4th, 8th…characters).

```
print (x[0:len(x):4])
```

(f) Find how many characters are there in the sentence (iprint(Including spaces)

```
len(x)
```

(g) Print every second character of the sentence starting from the last character in reverse order

```
print (x[-1:-len(x)-1:-2])
```

(h) Store the first four characters of x in a variable y and the last three letters in a variable z.

```
>>> y=x[0:4]
>>> print (y)
The
>>> z=x[40:]
>>> print (z)
dog
```

Print the output of y + z

```
>>> print (y+z)
The dog
```

(i) Print the output of y*10

```
print (y*10)
```

Q.3 Miscellaneous

a) Print 'Hello World' on the screen

b) Add a comment to the above program

c) Store your name (e.g. Tintin), age (e.g. 20) and roll number (e.g. 20MS1234) in three variables and print them separately and also together like *Hello! My name is Tintin . I am 20 years old. My roll number is 20MS1234* using these variables. Use string concatenation. Also, use formatted output.

```
Name='Tintin'
Age=20
RollNo='20MS1234'

print('Name=', Name,' Age=', Age, ' Roll Number=', RollNo)
print('Hello! My name is', Name,'. I am',Age,'years old. My roll number is', RollNo)
```

```
print("Hello! My name is {}. I am {} years old. My roll number is
{}".format(Name, Age, RollNo))
```

d) Take two number strings as input and print their sum

```
a=input('Give a number:')
b=input('Give another number')
print('The sum=', a+b)
```

e) Take two integers as input and print their sum

```
a=int(input('Give a number:'))
b=int(input('Give another number'))
print('The sum=', a+b)
```

f) Use print() to print the statement -- It's good to learn Python

```
print("It's good to learn Python")
```

g) Use print() to print the statement -- The man asked, "Where to meet you?" I said, "Well, use Google Meet!"

```
print('The man asked, "Where to meet you?" I said, "Well, use Google Meet!"' )
```

h) Store an integer, floating point number and character in different variables and print the data type of each variable.

```
integer = 1
floating = 1.5
str = 'c'

print(type(integer))
print(type(floating))
print(type(str))
```

i) Take your name (e.g. Feluda) in the variable 'Name' as input and print *My name is Feluda* using Name and string concatenation. Do the same using %s.

```
name=input("Give your name: ")
print("My name is " + name)

name=input("Give your name: ")
print("My name is %s" %(name))
```

Q4. Write a Python function that takes two lists and returns True if they have at least one common member.

```
# Define a function called 'common_data' that takes two lists, 'list1' and 'list2', as input
def common_data(list1, list2):
    # Initialize a variable 'result' to False to indicate no common elements initially
    result = False

    # Iterate through each element 'x' in 'list1'
    for x in list1:
        # Iterate through each element 'y' in 'list2'
        for y in list2:
            # Check if the current elements 'x' and 'y' are equal
            if x == y:
                # If there's a common element, set 'result' to True and return it
                result = True
                return result

# Call the 'common_data' function with two lists and print the result
print(common_data([1, 2, 3, 4, 5], [5, 6, 7, 8, 9]))
# Call the 'common_data' function with two lists and print the result
print(common_data([1, 2, 3, 4, 5], [6, 7, 8, 9]))
```

Q5. Write a Python program to convert a list of characters into a string.

```
# Define a list 's' containing individual characters
s = ['a', 'b', 'c', 'd']

# Use the 'join' method to concatenate the characters in the list 's' into a single string
str1 = ''.join(s)

# Print the concatenated string 'str1'
print(str1)
```

Q6. Write a Python program to remove the K'th element from a given list, and print the updated list.
Original list:
[1, 1, 2, 3, 4, 4, 5, 1]
After removing an element at the kth position of the said list:
[1, 1, 3, 4, 4, 5, 1]

```
# Define a function 'remove_kth_element' that takes a list 'n_list' and an integer 'L' as input
def remove_kth_element(n_list, L):
    # Return a modified list by removing the element at the kth position (L-1) from the input list
    return n_list[:L - 1] + n_list[L:]

# Create a list 'n_list' containing integers
n_list = [1, 1, 2, 3, 4, 4, 5, 1]

# Print a message indicating the original list
print("Original list:")
# Print the original list
print(n_list)

# Assign an integer 'kth_position' with the value 3
kth_position = 3

# Call the 'remove_kth_element' function with 'n_list' and 'kth_position'
# and store the result in the 'result' variable
result = remove_kth_element(n_list, kth_position)

# Print a message indicating the list after removing an element at the kth position
print("\nAfter removing an element at the kth position of the said list:")
# Print the 'result' list
print(result)
```

Set 2

1. Write a program which prints the following sequence of strings using a for loop

mitochondria
mitochondria
mitochondri
mitochondr
mitochond
mitochon
mitocho
mitoch
mitoc
mito
mit
mi
m

```
a = "mitochondria"

for i in range(len(a)+1, 0, -1):
    print(a[:i:])
```

2. Consider a list of names of chemical elements (e.g. sodium, potassium etc.) and write a program that finds the longest name in that list. Using “format()” print the longest name thus found with its length.

```
words = ["sodium", "potassium", "phosphorus", "calcium", "iodine"]

l = 0


for w in words:
    if len(w) > l:
        word_longest = w
        l = len(w)

print("The longest word is {} with length {}".format(word_longest, l))
```

3. Take your full name as input and make an abbreviation of your name based on the initials of the names.

```
name=input("Your name: ")
l = name.split()
print(l)
abbre=""

for n in range(0, len(l)-1):
    abbre = abbre + l[n][0] + ". "

abbre = abbre + l[len(l)-1]
print(abbre)
```

4. Write a program that prints the first 10 elements of a Fibonacci series 1, 1, 2, 3,5, 8, 13,....., where each element is the sum of the two previous elements (the first two numbers are defined to be 1).

```
f1 = 1
f2 = 1
print(str(f1))
print(str(f2))


for i in range(1, 9, 1):
    f = f1 + f2
    print(str(f))
    f1 = f2
    f2 = f
```

5. Print the pattern using nested for loops (for n lines):

1

1 2
1 2 3
1 2 3 4
1 2 3 4 5

```
n = 5
for i in range(1, n+1):
   for j in range(1, i+1):
       print(str(j), end = " ")
   print("\n")
```

6. Print the pattern using nested for loops (for n lines):

1

1 3

1 3 5

1 3 5 7

1 3 5 7 9

```
n = 5

for i in range(1, n+2, 1):
    for j in range(1, 2*i-1, 2):
        print(str(j), end=" ")

    print()
```

7. Print the pattern for n lines:

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

8. Print the pattern for n lines (mind the left spaces):

1 2 3 4 5 4 3 2 1

  2 3 4 5 4 3 2

    3 4 5 4 3

      4 5 4

        5

```
n = 5
for i in range(1, n+1):
    for l in range(1, i):
        print("  ", end="")
    for j in range(i, n+1):
        print(str(j) + " ", end="")
    for j in range(n-1, i-1, -1):
        print(str(j) + " ", end="")

    print("\n")
```

9. Without taking using input create a dictionary with some userid: password pairs as key:value pairs.
   i) Now take a userid from the user (use input()). If this userid is in your dictionary, print, ask for password; if the password is also correct print “Welcome to the portal!”; otherwise (if userid or password is wrong) print “Invalid credentials!”
   ii) Create a service for password change. Take a userid from the user (use input()). If this userid is in your dictionary, ask for a new password (twice). If the provided passwords are the same (and different from the existing password),

update the corresponding password in the dictionary. If the userid is not in your dictionary, print “Invalid username”.

i)

```
useridpassword = {"kripa": "kr123", "arun": "xyz34", "asha": "asha1990", "disha": "dish99"}

userid = input("Give user id: ")

if userid in useridpassword:
    password = input("Password? ")
    if useridpassword[userid] == password:
        print("Welcome to the portal!")
    else:
        print("Invalid credentials!")
```

ii)

```
useridpassword = {"kripa": "kr123", "arun": "xyz34", "asha": "asha1990", "disha": "dish99"}

userid = input("Give user id: ")

if userid in useridpassword:
    password = input("New Password? ")
    password1 = input("New password (again)? ")
    if password == password1 and password != useridpassword[userid]:
        useridpassword[userid] = password
        print(useridpassword)
    else:
        print("Updation failed!")



else:
   print("Invalid username!")
```

10. Using the Tabulation Method find an interval containing a root of the equation $\sin(x) + x^2 - 1$ in the interval [0, 1]. You may use the sin() function in the math module.

```
import math

def f(x):
    return math.sin(x) + x*x  - 1


import numpy as np

i1 = 0.0 #left interval point
i2 = 1.0 #right interval point

l = i1
r = i2
1
f1=f(l) #value at 0.0
l1 = l #0.0

step = 0.1

l+=step #0.1

for n in np.arange(l, r+step, step):
    f2=f(n)
    print("l1=" + str(l1) + " f(l1)=" + str(f1) + " n=" + str(n) + " f2=" + str(f2))
    if f1*f2 < 0.0:
        print("Desired interval :" + str(l1)+ " " + str(n))

        break



    l1 = n
    f1 = f(l1)
```

11. Using Tabulation method find a better approximate interval in the interval [0, 1] and then on this better interval use Bisection to find the root of the equation sin(x) + x^2 -1 = 0 . You may use the sin() function in the math module.

```
def f(x):
     return math.sin(x) + x*x -1

def bisection(a, b):
   if(f(a)*f(b)>=0):
      print("Invalid interval!")
      return

   mid = a


   while (b - a) >= 0.001:

      mid = (a + b)/2.0

    print("%f %f %f %f %f %f" %(a, f(a), b, f(b), mid, f(mid)))
      if(f(a)*f(mid) < 0):
         b = mid
      else:
         a = mid



   return mid

sol = bisection(0.6, 0.7)
print("The root is ", "%.4f"%sol)
```

12. Consider a ball dropping from a height ‘h’ (value, in cms, taken as input from the user). Each time the ball hits the ground, it bounces up half the immediate previous height it was at. That is after it is dropped from height h, it hits the ground and bounces up height h/2, drops down again, hits the ground, and bounces up height h/4, and so on. Write a program that prints these heights h, h/2, h/4…n times, where n is taken as input from the user. However, if the height reaches 0.05 cm, you should stop immediately. Also, you should print these heights as floating-point numbers using formatted output.

```
h = int(input("Height: "))
n = int(input("Bounce no: "))
```

```
i = 1
ht = h

while ht > 0.05 and i <= n:
    print("Height is %f" %ht)
    ht = ht/2
    n = n + 1
```

Set 3

1. Given a list L = [1, 3, 5, 7] add the elements 2, 4, 6 after 1, 3, 5 respectively.

```
L = [1, 3, 5, 7]

L.insert(1, 2)
L.insert(3, 4)
L.insert(5, 6)
```

2. In the above list, remove the last element.

```
L.pop(len(L)-1)
```

3. Write a Push() function that adds an element at the end of a List L and returns the updated List L.

```
def Push(L, e):
    L.append(e)

L = []
Push(L, 1)
Push(L, 2)
Push(L, 3)
Push(L, 4)
Push(L, 5)
print(L)
```

4. Write a Pop(L) function that removes the last element in a List L and returns the deleted element. Apply Pop() on list L in Q3 to remove the last element.

```
def Pop(L):
```

```
    return L.pop(len(L)-1)

print(Pop(L))
print(L)
```

5. Use Push() to add five names to a list Name and using Pop() display the names in the reverse order.

```
def Push(L, e):
    L.append(e)

def Pop(L):
    return L.pop(len(L)-1)

l = ["messi", "neymar", "ronaldo", "lukaku"]
Names = []
for i in range(0, len(l)):
    Push(Names, l[i])

print(Names)

for i in range(0, len(l)):
    print(Pop(Names))
```

6. Given a list L with repeating elements, remove the duplicates. You should not use any temporary list.

```
L = [1, 2, 2, 3, 3, 3, 4, 5]

L = list(set(L))
```

7. Given a list L = [1, 2, 3, 4, 5], use list comprehension to generate another list L1 containing the even numbers in L

```
L1 = [1, 2, 3, 4, 5]

L2 = [x for x in L1 if x%2 == 0]
```

8. Given two lists L1 = [1, 2, 3, 4, 5] and L2 = [5, 4, 10, 12], use list comprehension to generate another list L3 containing the sum of the odd elements in L1 and L2.

```
L1 = [1, 2, 3, 4, 5]
L2 =  [5, 4, 10, 12]
L3 = [(x+y) for x in L1 for y in L2 if x%2 == 1 and y%2 == 1]
```

9. Use a while loop to take a number as input from the user and continue the loop until the user inputs an even number.

```
while True:
    n = int(input("Give an even number: "))
    if n%2 == 1:
        print("The number is odd!")
    else:
        break
```

10. Use while loop to reverse an integer stored in a variable n and store the reversed number in a variable rev. You can't use a list, string operations or any Python functions for reversal.

```
n = 123
rev = 0

while n != 0:
    rev = rev*10 + n%10
    n = n//10


print(rev)
```

11. Take a number as input (use input()) and using the dictionary d = {1 : 'ONE', 2 : 'TWO', 3 : 'THREE', 4 : 'FOUR', 5 : 'FIVE', 6 : 'SIX', 7 : 'SEVEN', 8 : 'EIGHT', 9 : 'NINE'} only print the number as words. If the input is 1234, the output should be ONE TWO THREE FOUR. Note that, you have to use only this dictionary (and no

other source) to convert each digit of the input number to the corresponding word equivalent.

```
d = {1 : 'ONE', 2 : 'TWO', 3 : 'THREE', 4 :
'FOUR', 5 : 'FIVE', 6 : 'SIX', 7 : 'SEVEN', 8 :
'EIGHT', 9 : 'NINE'}

s = "3456"

for e in s:
   print(d[eval(e)], end=" ")

print()
```

12. Write a Python program to determine the direction (‘increasing’ or ‘decreasing’ or ‘not monotonic’) of monotonic sequence numbers.

```
def test_monotone(nums):
   # Check if all elements in the list 'nums' are in increasing order
   if all(nums[i] < nums[i + 1] for i in range(len(nums) - 1)):
      return "Increasing."
   # Check if all elements in the list 'nums' are in decreasing order
   elif all(nums[i + 1] < nums[i] for i in range(len(nums) - 1)):
      return "Decreasing."
   else:
      return "Not a monotonic sequence!"

# Assign a specific list of numbers 'nums' to the variable
nums = [1, 2, 3, 4, 5, 6]

# Print the original list of numbers 'nums'
print("Original list:")
print(nums)

# Print a message indicating the operation to be performed
```

```
print("Check the direction ('increasing' or 'decreasing') of the said list:")

# Print the result of the test function applied to the 'nums' list
print(test_monotone(nums))
```

https://www.w3resource.com/python-exercises/puzzles/index.php

13. Using Newton-Raphson method find all the roots of the equation $f(x) = x^3 + x^2 - x = 0$ in the interval [0, 1]. Note that the first order derivative of f(x) is $3x^2 + 2x - 1$.

```
import math

def f(x):
    return x**3 + x**2 - x

def df(x):
     return 3.0*x**2 + 2*x - 1


def newtonRaphson(x):
    h = f(x)/df(x)




    while abs(f(x)) >= 0.0001:

        h = f(x)/df(x)

        x = x - h



        print('x, f, df, h values: %f %f %f %f' %(x, f(x), df(x), h))



    return x
```

```
print("The root is ", "%.4f"%newtonRaphson(1.0))
```

14. Design an automatic Quizzing System (choose an apt name like “Proshnobaan”) that asks science questions. Use a Python dictionary (KB) to store the QAs with correct and wrong options and scores for the correct option. Each element in KB is a key:value pair, key being the question (e.g. "What is the smallest known organism?") and the value is a list of tuples. Each element of the list is a tuple in the format (option, correct/wrong, score), e.g. ("Mycoplasma gallicepticum", 1, 2), where "Mycoplasma gallicepticum" is an option for the question, ‘1’ indicates that this is the CORRECT option for the question and ‘2’ is the score that the participant will get if s/he chooses this option. On the other hand, the tuple for a wrong option is ("Valonia ventricosa", 0, 0). For each question, the system should ask the question, show the options and ask for the answer. After the participant enters the answer, the system will print an appropriate message like “Correct answer” or “Wrong answer”.

The system will ask a mix of easy-to-tough questions (the score will be higher for the tougher questions) one after the other and add up the scores obtained by the participant. If the total score at the end of these questions is less than or equal to a threshold, the system will announce a consolation prize. If this total score is more than the threshold, the system will ask a JACKPOT question (an unusual question like "What is the smallest resolvable unit of distance by a given computer mouse pointing device called?") WITHOUT any option. If the participant answers the jackpot question correctly, the system should print “Congratulations” with the announcement of a mega prize; otherwise, it should print an appropriate message.

```
KB = {"What is the smallest known organism?":
     [("Mycoplasma gallicepticum", 1, 2),
      ("Valonia ventricosa", 0, 0)],
     "What is the unit of luminous intensity?":
        [("Candela", 1, 1), ("Kelvin", 0, 0)]
        }

#print(KB["What is the smallest known organism?"])
score_obtained = 0
#while True:
for q in KB:
   print("Question: " + q)
```

```
    print("Your options:")
    correct_ans = ""
    score_allotted = 0
    for op in KB[q]:
        print(op[0])
        if op[1] == 1:
            correct_ans = op[0]
            score_allotted = op[2]

    print()
    ans = input("Your answer: ")
    #print(ans)
    if ans == correct_ans:
        print("Correct")
        print("Your score is %d" %score_allotted)
        score_obtained = score_obtained + score_allotted
    else:
        print("Wrong!")


print("Total score %d" %score_obtained)

if score_obtained == 3:
    print("Your Jackpot question:")
    ans_jackpot = input("What is the smallest resolvable unit of distance by a given
computer mouse pointing device called?")
    ans_jackpot = ans_jackpot.lower()
    if ans_jackpot == "mickey":
        print("Congratulations! Here comes your mega prize!!")
    else:
        print("Well played, but you missed the Jackpot :-( We still have an exciting prize for
you..")
else:
    print("Good try, please collect your consolation prize.")
```

Set 4

1. Use list comprehension to create a list containing all the elements in list L1 and not in L2.

```
L1 = [1, 2, 3, 4]
L2 = [1, 5, 6, 7]

L3 = [e for e in L1 if e not in L2]
```

2. Input the name and basic salary (B) of an employee. The total salary T = DA + HRA + MA, where DA = 17% of B, HRA = 8% of B and MA = 10% of B. Print name right adjusted with 2 padding spaces (assume the length of name is 5) and the total salary right-adjusted (assume that the salary will not be more than 10 digits) and corrected to 3 places of decimal using formatted output applying % operator.

```
name = input('Name:')
basic = float(input('Basic salary:'))

DA = basic*(17/100)
HRA = basic*(8/100)
MA = basic*(10/100)

Total_salary = DA + HRA + MA

print("The salary of %7s is %-10.3f" %(name, Total_salary))
A more generic solution if the length of name is not known:

p_n = len(name)+2
print("The salary of %{}s is %-10.3f".format(p_n) %(name, Total_salary))
```

3. Do as (3) using the format() method

```
name = input('Name:')
basic = float(input('Basic salary:'))

DA = basic*(17/100)
HRA = basic*(8/100)
MA = basic*(10/100)

Total_salary = DA + HRA + MA

print("The salary of {0:7s} is {1:-10.3f}".format(name, Total_salary))
```

4. Use numpy array to generate the following pattern for n (taken from user) lines (shown for n = 4):

1.0

1.0 1.5 2.0

1.0 1.5 2.0 2.5 3.0

1.0 1.5 2.0 2.5 3.0 3.5 4.0

```
import numpy as np

n = 5

for i in range(1, n+1):
    for j in np.arange(1, i+0.5, 0.5):
        print(str(j), end = ' ')

    print("\n")
```

5. Write a function encrypt() that accepts a string and maps each character of the string to the corresponding alphabet in the opposite order of alphabets. E.g. ‘a’ will be mapped to ‘z’, ‘b’ will be mapped to ‘y’, ‘z’ will be mapped to ‘a’ and so on. So, encrypt() will map ‘zbc’ to ‘ayx’. Assume that the input string is in the lower case. Note that ord(‘a’) gives the ASCII value of the character ‘a’ (i.e. 97) and chr(97) gives the character equivalent of 97 (i.e. ‘a’).

```
def encrypt(s):
    s_out = ''
    for c in s:
        s_out += chr(122-ord(c)+ 97)
     return s_out

  print(encrypt("zbc"))
```

6. Write a python program to swap the left and right halves of an input string s; the middle element will be unchanged for an odd length string. E.g. ‘kripa’ will be converted to ‘paikr’; ‘aman’ will be converted to ‘anam’.

```
s = 'kripa'
mid=len(s)//2

if(len(s)%2 == 0): #even
```

```
    sout = s[mid:len(s)] + s[0:mid]
else:
    sout = s[mid+1:len(s)] + s[mid] + s[0:mid]



print(sout)
```

7. Input a list from the user (containing duplicates) and create another list containing only the non-repeating elements. E.g. if the input list is [1, 2, 2, 3, 4, 4, 5], the output list will be [1, 3, 5].

```
lst = input('Give a list: (space separated, put an enter to end)').split(" ")

print(lst)


l = len(lst)

lst_out = []

for i in range(0, l):
    r = 0
    for j in range(0, l):
        if i != j:
            if lst[i] == lst[j]:
                r = 1
                break
    if r == 0:
        lst_out.append(lst[i])

print(lst_out)
```

8. Print the numpy array [[1,2,3], [4, 5, 6]] column-wise by using slicing.

```
import numpy as np


A = np.array([[1,2,3], [4, 5, 6]])
c = A.shape[1] #no of cols
for i in range(0, c):
```

```
        print(A[:,i])
```

9. Define a function calc_area (using def) that takes a radius (r) and returns the area of a circle (use math.pi). Take a list radius_list containing five radii values and use calc_area to store the corresponding areas in another list area_list.

```
import math
import numpy as np

def calc_area(r):
    return math.pi*r*r

radius_list = [1, 2, 3]
area_list = []
for l in radius_list:
    area_list.append(calc_area(l))
```

10. Repeat the problem in (2) using a numpy array radius_np containing five radii values (same as in radius_list) to store the corresponding areas in another numpy array area_np.

```
import numpy as np
radius_np = np.array([1, 2, 3])
area_np = calc_area(radius_np)
```

11. Input (use input()) the roll number and marks of three courses for 5 students and store it in a dictionary (d) such that the roll number is the key and the marks are stored in a numpy array of type float64 (the type of a numpy array A can be changed to float64 type by A.astype('float64')). For example, for a student with roll number 'ms1901' and marks 70, 80, 90, the dictionary entry will look like 'ms1901': array([70., 80., 90.])

12. Continue with Q6 to create another dictionary 'd_sum' that contains the sum of the marks. So, the corresponding dictionary entry will look like 'ms1901': 240.0. To compute sum use numpy function sum() (Note: The sum of a numpy array A is calculated as numpy.sum(A))

```
d = {}
for i in range(0, 6):
```

```
    rollno = input("Roll no ")
    marks = input("Three marks, space separated, press enter to end: ").split(" ")
    d[rollno] = np.array(marks).astype('float64')


#print(d)
d_sum = {}

for e in d:
    d_sum[e] = np.sum(d[e])


#print(d_sum)
```

13. Check if two 1-D numpy arrays are equal or not using a for loop. Verify the result with the numpy function numpy.array_equal(a1, a2) that returns True if the arrays a1, a2 are equal.

```
import numpy as np

a1 = np.array([1, 2, 3, 4])
a2 = np.array([1, 2, 3, 5])

if len(a1) != len(a2):
    print("The arrays are not equal!")
else:
    unequal = False
    for i in range(0, len(a1)):
        if a1[i] != a2[i]:
            unequal = True
            break
    if unequal == False:
        print("The arrays are equal!")
    else:
        print("The arrays are not equal!")

print(np.array_equal(a1, a2))
```

14. Consider a game “Give me thy name” with n players P1, P2, …, Pn. Each player has a dedicated variable ‘sum’ (n ‘sum’s for n players). The game goes like this: P2 takes the length of P1’s name and adds up to his/her existing sum (initially it is zero for all the players). Then this continues to P3 who adds the length of P2’s name to his/her sum and so on. This goes from left to right (P1 to Pn) and then right to left (where the summing is

done right to left; Pn-1 adds the length of the name of Pn to his/her current sum and passes the control to Pn-2 and so on) to constitute one pass. There can be ‘m’ such passes (left to right and then right to left). Each time a sum is updated for a player, the sum is checked to be prime or not. If the sum is prime, the corresponding player is eliminated. Then the process continues from the next player.

You should stop the game, the moment you are left with one player – who wins the game. If at the end of m passes, you have more than one player, the one with the highest sum wins. You should print the winner’s name.

You should not use another variable to store the sum; it should be stored with each player's information. You should have a list of lists, where each element is a list of three elements: player name, name length, and current sum. As an example,

Initial players:

[['Soumyadeep Sar', 12, 0], ['Anurag Sharma', 13, 0], ['Aakash Ghosh', 12, 0], ['Tisha Dash', 10, 0], ['Sirswa Kuldeep Shree Ram', 24, 0]]

Pass: 1
Left to right==>

[['Soumyadeep Sar', 12, 0], ['Anurag Sharma', 13, **12**], ['Aakash Ghosh', 12, 0], ['Tisha Dash', 10, 0], ['Sirswa Kuldeep Shree Ram', 24, 0]]

[['Soumyadeep Sar', 12, 0], ['Anurag Sharma', 13, 12], ['Aakash Ghosh', 12, **13**], ['Tisha Dash', 10, 0], ['Sirswa Kuldeep Shree Ram', 24, 0]]

Aakash Ghosh gets eliminated as ‘13’ is prime to produce the updated list as:

[['Soumyadeep Sar', 12, 0], ['Anurag Sharma', 13, 12], ['Tisha Dash', 10, 0], ['Sirswa Kuldeep Shree Ram', 24, 0]]

This continues and finally Anurag Sharma wins.

```
team = [ ["Kripa", 5, 0], ["Anurag", 6, 0], ["Aakash", 6, 0], ["Tisha", 5, 0], ["Richa", 5, 0]]

def isPrime(num):

    # define a flag variable
    flag = False
```

```
    if num <= 1:
        return False
    elif num > 1:
        # check for factors
        for i in range(2, num):
            if (num % i) == 0:
                # if factor is found, set flag to True
                flag = True
                # break out of loop
                break


    return not flag


def max_candidate(lst):
    max = 0
    max_name = ""

    for l in lst:
        if l[2] > max:
            max_name = l[0]
            max = l[2]

    return max_name

#print(isPrime(4))
print(team)
i = 0
is_single_flag = 0

for n in range(0, 5):
    print("Pass: %d" %(n+1))
    print("Left to right==>")

    i = 0
    length = len(team)
    while i < length:
        if i != 0:
            team[i][2] = team[i][2] + team[i-1][1]
            print(team)
        if isPrime(team[i][2]):
            print("Is Prime: ")
            print(team[i])
```

```
            #team[i][2] = 0
            team.remove(team[i])
            length = len(team)
            i = i - 1
            print(team)
        i = i + 1
        if length == 1:
            is_single_flag = 1
            break
        print("End: ")
        print(team)
    if(is_single_flag):
         break

    print("Right to left <==")


    length = len(team)
    i = length - 1

    while i >= 0:
        if i != length - 1:
            team[i][2] = team[i][2] + team[i+1][1]
            print(team)
        if isPrime(team[i][2]):
            print("Is Prime: ")
            print(team[i])
            #team[i][2] = 0
            team.remove(team[i])
            length = len(team)
            i = i + 1
            print(team)
        i = i - 1
        if length == 1:
            is_single_flag = 1
            break
        print("End: ")
        print(team)
    if(is_single_flag):
         break

if is_single_flag:
    print(team[0][0])
else:
```

```
print(max_candidate(team))
```

Set 5

1. Input (using input()) a list of alternating names and ages e.g. [‘kripa’, ‘37’, ‘arun’, ‘45’, ‘dipa’, ‘40’] and create a dictionary such the (i, i+1)th elements (i = 0, 2, …) form the key value pairs. For the mentioned example, the dictionary will be {‘kripa’: ‘37’, ‘arun’: ‘45’, ‘dipa’: ‘40’}

```
inpt = input("Give alternative names and ages, press enter to end: ").split(" ")
lst = list(inpt)
d = {}

for i in range(0, len(lst)-1, 2):
    d[lst[i]] = lst[i+1]
```

2. For the same input of (1), create a list of tuples of the (i, i+1)th elements (i = 0, 2, …) of the input list. For the mentioned example, the output list will be [('kripa', ‘37’), ('arun', ‘32’), ('dipa', ‘40)]

```
inpt = input("Give alternative names and ages, press enter to end: ").split(" ")
lst = list(inpt)
lst_out = []

for i in range(0, len(lst)-1, 2):
    lst_out.append((lst[i], lst[i+1]))
```

3. Use numpy linalg.solve to find the point of intersection of the three planes x + 2y + 3z = 2, 4x + 8y +66z = 3 and 7x + 81y + 9z = 4. Perform the said operation if the coefficient determinant is non-singular. This can be checked by the function np.linalg.det(). You may consider a determinant to be singular if np.linalg.det() is close to zero (say abs() value less than 0.00001).

```
A1 = np.array([[1, 2, 3], [4, 8, 66], [7, 81, 9]])
B1 = np.array([2, 3, 4])
```

```
dval = np.linalg.det(A1)

if(abs(dval) > 0.00001):
    print(np.linalg.solve(A1, B1))
else:
    print("Singular!")
```

4. Write a function d2b() that takes a decimal number as argument and returns its binary equivalent. Write a numpy ufunc function numpy_DecimalToBinary that uses d2b() and applies the same operation elementwise on a numpy array A. E.g. if A is 1, 2, 3, 4, 5, numpy_DecimalToBinary(A) will produce **1 10 11 100 101**. Note that all the elements of numpy_DecimalToBinary(A) should be of 'int' type and not 'str' type.

```
def d2b(x):

    dgts = ""
    while x != 0:
        digit = x%2
        dgts = str(digit) + "" + dgts
        x = x//2

    return int(dgts)

arr1 = np.array([1, 2, 3, 4, 5])
numpy_DecimalToBinary = np.frompyfunc(d2b, 1, 1)
arr2 = numpy_DecimalToBinary(arr1)
print(arr2)
```

5. Consider a numpy 2-D array of the form [[50, 60, 70], [67, 88, 90], [60, 78, 97]] where the ith 1-D array contains the marks of the ith student in three subjects (in this order. E.g. 50, 60, 70 are the subject1, subject2, subject3 marks respectively of student-0. Use numpy sum function only to i) create a 1-D numpy array with the sum of the marks of individual students ii) create a 1-D numpy array with the sum of subject-wise marks. Also, do these operations to produce 2-D numpy arrays with the same content.

6. Given two 1-D numpy arrays A and B, remove the elements in A which are also in B and store the resulting array in C. Use numpy set operations.

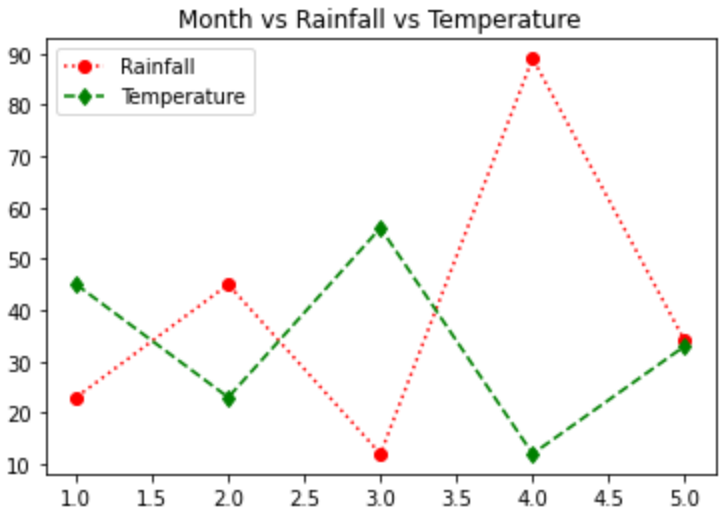
7. Given two numpy 2-D arrays arr1 = np.array([[1, 2], [4, 5]]), arr2 = np.array([[3, 3], [1,1]]) explore the difference between np.multiply(arr1, arr2) and np.matmul(arr1, arr2).

8. Take months (say 2, 3 etc.) and rainfall (in millimeters) as input from the user and plot months (x-axis) vs rainfall (y-axis) using plot in matplotlib.

9. Add another series for temperature (in degree celsius) and add it in the plot in (1) against months. Make sure you use different colors and markers for the two series (rainfall and temperature) against months.

10. Play around with line styles and colors of the two series. Also add title and legend.

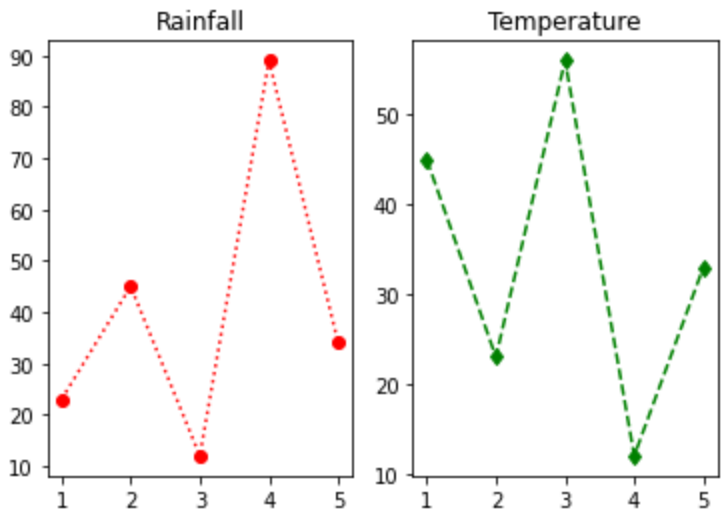
```
month = input("Give space-separated values: ").split(" ")
rain = input("Give space separated marks: ").split(" ")
temp = input("Give space separated marks: ").split(" ")

x = np.array(month)
y = np.array(rain)
z = np.array(temp)


import matplotlib.pyplot as pt

pt.title("Month vs Rainfall vs Temperature")

pt.plot(x, y, 'o:r', label="Rainfall")
pt.plot(x, z, 'd--g', label = "Temperature")

pt.legend()
```



11. Show the above two plots as subplots i) along the same row and ii) along the same column. Put the title on each subplot.

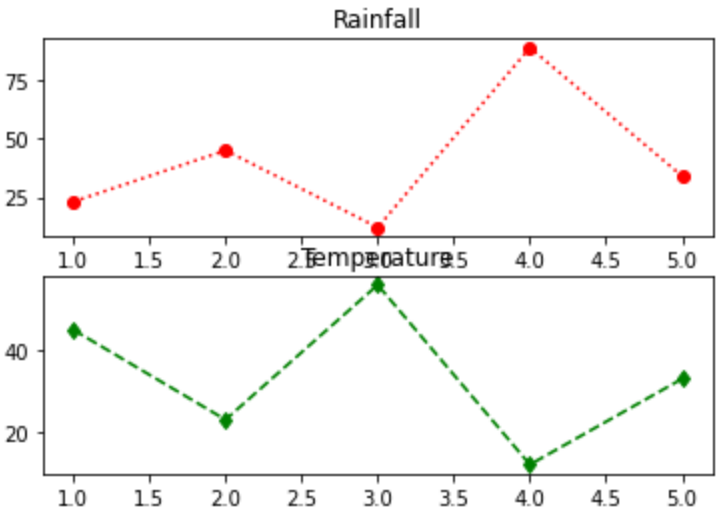
i)

```
month = input("Give space-separated values: ").split(" ")
rain = input("Give space separated marks: ").split(" ")
temp = input("Give space separated marks: ").split(" ")

x = np.array(month)
y = np.array(rain)
z = np.array(temp)


import matplotlib.pyplot as pt

pt.subplot(1, 2, 1)
pt.title("Rainfall")
pt.plot(x, y, 'o:r', label="Rainfall")

pt.subplot(1, 2, 2)
pt.title("Temperature")
pt.plot(x, z, 'd--g', label = "Temperature")
```



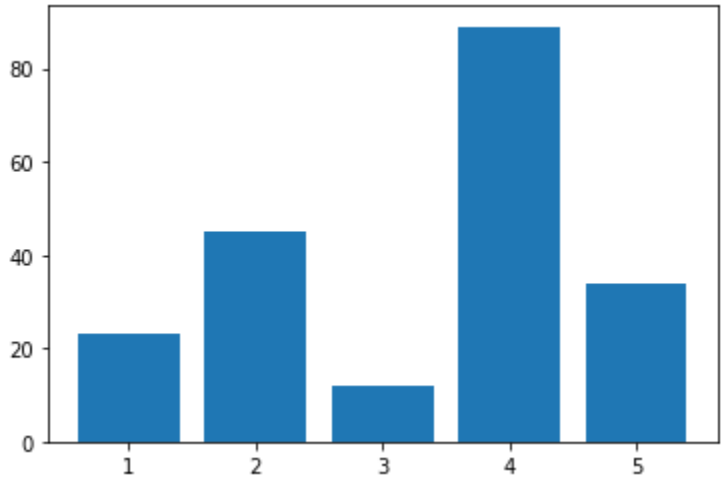
ii)

```
month = input("Give space-separated values: ").split(" ")
rain = input("Give space separated marks: ").split(" ")
temp = input("Give space separated marks: ").split(" ")

x = np.array(month)
y = np.array(rain)
z = np.array(temp)


import matplotlib.pyplot as pt

pt.subplot(2, 1, 1)
pt.title("Rainfall")
pt.plot(x, y, 'o:r', label="Rainfall")

pt.subplot(2, 1, 2)
pt.title("Temperature")
pt.plot(x, z, 'd--g', label = "Temperature")
```



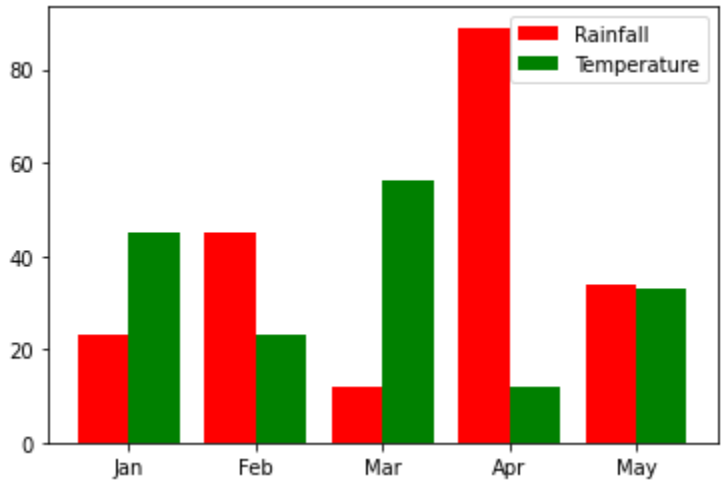
12. Use the data of Q8 to draw a bar chart.

```
x = np.array(month)
y = np.array(rain)

pt.bar(x, y)
```



13. Use the same data to draw a bar graph showing the rainfall and temperature for each month.

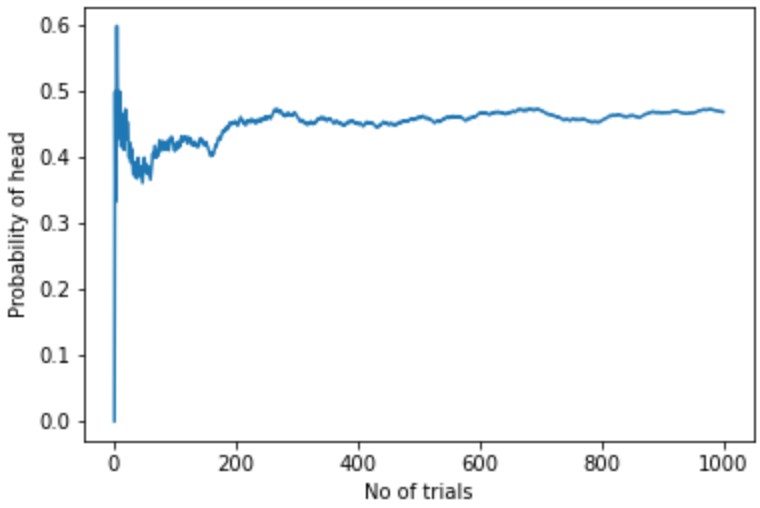
```
x = np.array(month)
y = np.array(rain)
z = np.array(temp)

X = ["Jan", "Feb", "Mar", "Apr", "May"]

pt.bar(x-0.2, y, 0.4, label = "Rainfall", color = "red")
pt.bar(x+0.2, z, 0.4, label = "Temperature", color = "green")
pt.xticks(x, X)
pt.legend()
```



14. Use the same data to draw a pie chart.

15. Consider a trial of tossing an unbiased coin n (a large value) times. Use random.random() [generates a random float number between 0 and 1] to simulate the observation after each coin toss as follows: if the random number is <0.5, assume that H has occurred, tail otherwise. Count the number of heads and compute the fraction of heads generated after each coin toss. This is basically to estimate the probability of head (that is 0.5). Plot the estimated probability (using matplotlib.pyplot.plot) along the y-axis, while the trial numbers are plotted along the x-axis.

```
import random
head = 0
x = []
y = []
n = 1000

for i in range(1, n+1):
    x.append(i)
    r = random.random()
    if r < 0.5:
        print("head")
        head += 1

    y.append(head/i)


print(x)
print(y)

import matplotlib.pyplot as plt
plt.xlabel("No of trials")
plt.ylabel("Probability of head")
plt.plot(x, y)
#plt.show()
plt.savefig("head.png")
```


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Probability of head
0
200
400
600
800
1000
No of trials