Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in
Machine Learning and Artificial Intelligence

Python & Its Applications

Question Bank 1

edureka!

1. Using escape sequence print each word in different lines. Sample:

   Input : 'Hey! Welcome to edureka!'

   Output:

   ```
   Hey!
   Welcome
   to
   edureka!
   ```

In [1]:

```
my_text='Hey! Welcome to edureka!'
for i in my_text.split(' '):
    print(i)
```

```
Hey!
Welcome
to
edureka!
```

2. Write a program to create a list of even numbers starting from x to y.

   Input:100 200

   Output:[100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200]

In [2]:

```
x,y=input().split(' ')
even_num=list(range(100,201,2))
print(even_num)
```

```
100 200
[100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128,
130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 1
60, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 19
0, 192, 194, 196, 198, 200]
```

3. Write a program to replace the given word **X with Y** from a string.

   Input: Hello! I love apples

In [3]:

```
string=input()
X=input()
Y=input()
print(string.replace(X,Y))
```

```
Hello! I love apples
apples
oranges
Hello! I love oranges
```

---

4. Write a program generate **N** numbers of fibbonacci series seperated by space.

   Input: 5

   Output: 0 1 1 2 3

In [4]:

```
N=int(input())-1
x=0
y=1
print(x,end=' ')
while N:
    c=x+y
    y=x
    x=c
    print(x,end=' ')
    N=N-1
```

```
5
0 1 1 2 3
```


---

5. Write a program to find sum of digit from a given integer number **N**.

   Input:4235

   Output:14

In [5]:

```
N=int(input())
N_sum=0
while N:
    N_sum=N_sum+(N%10)
    N=N//10
print(N_sum)
```

```
4235
14
```

6. Write a program to find factorial of a given number **N**.

   Input:5

   Output:120

In [6]:

```
N=int(input())
fact=1
while N:
    fact=fact*N
    N=N-1
print(fact)
```

```
5
120
```


7. Write a program to generate the following pattern:

```
*
**
***
****
*****
```

In [7]:

```
for i in range(5):
    print('*'*(i+1))
```

```
*
**
***
****
*****
```

8. Write a program to generate the following pattern.

```
    *
   ***
  *****
 *******
*********
```

In [8]:

```
for i in range(5):
    print(' '*(5-i+1)+'*'*(2*i+1))
```

```
      *
     ***
    *****
   *******
  *********
```

9. Write a program to generate the following pattern for given depth N.

   Input:5

   Output:

```
          *
         ***
        *****
       *******
      *********
```

In [9]:

```
N=int(input())
for i in range(N):
    print(' '*(N-i+1)+'*'*(2*i+1))
```

```
5
      *
     ***
    *****
   *******
  *********
```

10. Write a program to generate the following output.

```
1
2 3
3 4 5
4 5 6 7
5 6 7 8 9
```

In [10]:

```
N=list(range(1,11))
for i in range(5):
    for j in range(i+1):
        print(N[i+j],end=' ')
    print()
```

```
1
2 3
3 4 5
4 5 6 7
5 6 7 8 9
```

11. Write a program that accepts **N** values and print it as a list

    Input:

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

    Output:

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


In [11]:

```
N=int(input())
my_list=[]
for i in range(N):
    my_list.append(int(input()))
print (my_list)
```

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

12. Write a program to find the given number is **Armstrong number** or not and print out as **'Yes'** or **'No'**.

    [**Hint**: Armstrong Number is sum of its own digits raised to power of 3]

    Input:153

    Output:Yes

In [12]:

```
num=input()
s=0
for i in list(num):
    s=s+int(i)**3
if s==int(num):
    print('Yes')
else:
    print('No')
```

```
153
Yes
```

13. Write a program to find the N th number of a **Arthematic Progression** when initial number(**a**) and difference(**d**) are given.

    Input:

    ```
    The input is N, a, d each in newline
            4
            2
            3
    ```

    Output:

    ```
            11
    ```


In [13]:

```
N=int(input())
a=int(input())
d=int(input())
num=a+(N-1)*d
print(num)
```

```
4
2
3
11
```

14. Write a program to generate a sequence of **N** numbers of a geometric progression when intial number (**a**) and common ratio(**r**) is given. print the sequence as a list.

    Input:

    ```
    The input is N,a,r seperated by comma.
            5,2,3
    ```

    Output:

In [14]:

```
N,a,r=map(int,input().split(','))
seq=[]
for i in range(N):
    seq.append(a*r**(i))
print(seq)
```

```
5,2,3
[2, 6, 18, 54, 162]
```

15. Write a program to find the count of each aplhabet in the given sentence.

    Input: life of pi

    Output:

```
f 2
p 1
  2
o 1
l 1
e 1
i 2
```


In [15]:

```
sentence=input()
set_sentence=set(list(sentence))
for i in set_sentence:
    print(i,sentence.count(i))
```

```
life of pi
i 2
  2
f 2
o 1
e 1
l 1
p 1
```

16. Write a program that calculate number of lowercase, uppercase and numbers alphabets.

    Input:Happy Birthday Draven! Now you are 26.

    Output:

```
Uppercase characters:  4
Lowercase characters:  24
Numbers:  2
```

In [16]:

```
sentence=input()
up=0
low=0
num=0
for i in sentence:
    if i.isupper():
        up=up+1
    elif i.islower():
        low=low+1
    elif i.isdigit():
        num=num+1
print('Uppercase characters: ',up)
print('Lowercase characters: ',low)
print('Numbers: ',num)
```

```
Happy Birthday Draven! You are now 26.
Uppercase characters:  4
Lowercase characters:  24
Numbers:  2
```

17. Write a program to find whether a given string is palindrome or not

    Input: madam

    Output: Yes

In [17]:

```
word=input()
if word==word[::-1]:
    print('Yes')
else:
    print('No')
```

```
madam
Yes
```

18. Write a program that accepts a sequence of numbers and print the numbers after sorting them seperated by comma.

    Input:

    ```
    5
    2
    3
    7
    4
    ```

    Output:

    ```
    2,3,4,5,7
    ```

In [18]:

```
N=input()
my_list=[]
for i in range(int(N)):
    my_list.append(int(input()))
my_list.sort()
print(my_list)
```

```
5
2
3
5
7
4
[2, 3, 4, 5, 7]
```

19. Given below a list of integers, find the triplets whose sum if equal to **zero** and print them as tuples.

    [0, 2, 3, 4, 5, 6, 9, -1, -9, -7, -6, -5, -4, -3, -2]

    Example:

    [4,0,-2,-1,2,3]

    Output:

    (4,-2,-2)

    (-2,0,2)

    (-2,-1,3)

In [19]:

```
int_list=[0, 2, 3, 4, 5, 6, 9, -1, -9, -7, -6, -5, -4, -3, -2]
temp=int_list
int_list.sort()
out=[]
for mid in range(len(int_list)-1):
    right,left=mid+1,mid-1
    adr=1
    adl=0
    temp=int_list.copy()
    m_item=temp[mid]
    while temp[left]!=m_item or temp[right]!=m_item:

        s=temp[left]+temp[mid]+temp[right]
        if s==0:
            out.append(tuple([temp[left],temp[mid],temp[right]]))
            r_item=temp[right]
            l_item=temp[left]
            temp.remove(r_item)
            temp.remove(l_item)
        right=right+adr
        left=left-adl

        if right >= len(temp):
            right=mid+1
            if right >=len(temp):
                break
            adr=0
            adl=1
            temp=int_list.copy()
        if -left >= len(temp):
            break
for i in set(out):
    print(i)
```

```
(-5, 2, 3)
(5, -3, -2)
(-2, 0, 2)
(6, -9, 3)
(-5, -4, 9)
(3, -2, -1)
(-3, -2, 5)
(9, -9, 0)
(-2, -1, 3)
(-7, 3, 4)
(-9, 4, 5)
(9, -5, -4)
```

20. Using for loop print aplhabets from a-z seperated by space.

In [20]:

```
for alphabet in range(97,123):
    print(chr(alphabet),end=' ')
```

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

'edureka' from the output or not.

A={'c', 'a', 'd', 'e', 'k', 'r', 'u', 'q', 'r', 'y'}

B={'b', 'h', 'k', 'a', 'd', 'e', 'k', 'r', 'u'}

In [21]:

```
A={'c', 'a', 'd', 'e', 'k', 'r', 'u', 'q', 'r', 'y'}
B={'b', 'h', 'k', 'a', 'd', 'e', 'k', 'r', 'u'}
word=set(list('edureka'))
if len(A.intersection(B).difference(word))==0:
    print('Yes')
else:
    print('No')
```

```
Yes
```

22. Write a program which accepts a string from console and print it in reverse order.

Input:

```
Always keep one finger on the Escape key
```

Output:

```
yek epacsE eht no regnif eno peek syawlA
```

Difficulty: Easy

In [22]:

```
my_str=input()
print(my_str[::-1])
```

```
Always keep one finger on the Escape key
yek epacsE eht no regnif eno peek syawlA
```

23. Write a program which accepts a string from console and print the words in reverse order.

Input:

Always keep one finger on the Escape key

Output:

syawlA peek eno regnif no eht epacsE yek

Difficulty: Easy

In [23]:

```
my_str=input().split(' ')
rev_str=''
for word in my_str:
    rev_str=rev_str+word[::-1]+' '

print(rev_str)
```

```
Always keep one finger on the Escape key
syawlA peek eno regnif no eht epacsE yek
```

24. Write a program to calculate the numbers between **X** and **Y** which are divisible by 5 and 13 but not divisible by 10 seperated by space.

    Input: X and Y are seperated by space:

    50 200

    Output:

    65 195

In [24]:

```
X,Y=map(int,input().split(' '))
div=13*5
for i in range(X,Y+1):
    if i%div==0 and i%10!=0:
        print(i, end=' ')
```

```
50 200
65 195
```

25. Write a program to calculate all possible permutations for a given sequence of numbers.

    Input: N is length of sequence and next line contains sequence seperated by space.

    ```
    3
    4,6,2
    ```

    Output: [[2, 6, 4], [6, 2, 4], [6, 4, 2], [2, 4, 6], [4, 2, 6], [4, 6, 2]]

In [25]:

```
N=int(input())
seq=list(map(int,input().split(',')))
perms = [[]]
for num in seq:
    temp = []
    for perm in perms:
        for i in range(len(perm)+1):
            temp.append(perm[:i] + [num] + perm[i:])
            perms = temp

print("Permutation: ",perms)
```

```
3
4,6,2
Permutation:  [[2, 6, 4], [6, 2, 4], [6, 4, 2], [2, 4, 6], [4, 2, 6], [4, 6,
2]]
```

26. Samantha represents the performance of students using dictionary in tests as follows:

```
{'English':{'Sam':60,'Jackson':74}, 'History':{'Gloria':83,'Sam':35} ,....}
```

Your task is to create a dictionary with student names as keys and their average marks from all subjects. Consider the below dictionary as input.

```
{'English':{'Sam':60,'Jackson':74,'Ahree':85},
 'History':{'Gloria':83,'Sam':65,'Isla':78,'Aron':72,'Gray':61},
 'Geography':{'Jackson':92,'Gloria':95,'Isla':82,'Aron':75,'Ahree':76},
 'Mathematics':{'Sam':99,'Gloria':74,'Jackson':89,'Ahree':85,'Gray':95},
 'Science':{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}
}
```

In [26]:

```
tests={'English':{'Sam':60,'Jackson':74,'Ahree':85},
 'History':{'Gloria':83,'Sam':65,'Isla':78,'Aron':72,'Gray':61},
 'Geography':{'Jackson':92,'Gloria':95,'Isla':82,'Aron':75,'Ahree':76},
 'Mathematics':{'Sam':99,'Gloria':74,'Jackson':89,'Ahree':85,'Gray':95},
 'Science':{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}
}

keys=[]
for i in tests:
    keys.extend(tests[i].keys())
keys=set(keys)
Average_marks=dict.fromkeys(keys,0)

for i in tests:
    for key in tests[i]:
        Average_marks[key]=Average_marks.get(key)+tests[i][key]

for i in Average_marks:
    Average_marks[i]=Average_marks[i]/5
print(Average_marks)
```

```
{'Isla': 50.6, 'Ahree': 66.6, 'Aron': 45.8, 'Gray': 46.8, 'Sam': 62.6, 'Glor
ia': 50.4, 'Jackson': 51.0}
```

27. Create a dictionary from the following keys: 'Name', 'Class', 'Age'

    with empty lists and input the data of five students.

In [27]:

```
data={'Name':[],'Class':[],'Age':[]}
for i in range(5):
    data['Name'].append(input())
    data['Class'].append(input())
    data['Age'].append(input())
```

```
John
10
26
Peter
11
27
Amy
10
23
Schumi
12
31
Anna
12
26
```

28. Write a program to concatenate the following dictionaries:

    dict1={'A':10, 'B':20}

    dict2={'C':30, 'D':40}

In [28]:

```
dict1={'A':10, 'B':20}
dict2={'C':30, 'D':40}
dict3={'E':50,'F':60}
new_dict=dict()
new_dict.update(dict1)
new_dict.update(dict2)
new_dict.update(dict3)
print(new_dict)
```

```
{'A': 10, 'B': 20, 'C': 30, 'D': 40, 'E': 50, 'F': 60}
```

29. Write a program to generate prime numbers from 1 to 1000 and save them to a text file. Read N lines of the previous file.

In [29]:

```
file=open('Sample.txt','w')
for num in range(1,1000):
    flg=True
    for i in range(2,num):
        if (num%i)==0:
            flg=False
            break
    if flg:
        file.write(str(num)+'\n')

file.close()
N=int(input())
file=open('Sample.txt','r')
file.readlines(N)
file.close()
```

```
45
```

30. Create a game of Rock, Papers and Scissors against computer using Python. Rules for the game are given as follows:

    - Rock beats Scissor
    - Scissor beats Paper
    - Paper beats Rock

    If the play loses ask him either he wants to payback or not. If the response is **'Yes'** then continue the game. If the player says **'No'** then display the final scores. Input moves as **R for Rock, P for Paper and S for scissors**. Response to every move should be proper example:

    Input:

    ```
    R S
    ```

    Output:

    ```
    Win
    Play Again?
    ```

In [30]:

```
import random
P1=0
P2=0
win_lose={'Lose':['RP','PS','SR'],'Draw':['RR','PP','SS'],'Win':['RS','PR','SP']}
repeat='Yes'
c=['R','S','P']
while repeat=='Yes':
    turn=input()
    comp=random.choice(c)
    for i in win_lose:
        if str(turn)+comp in win_lose[i]:
            print(i)
            repeat=input('Play?')
```

```
R
Win
Play?N
```

31. Write a program to create a pandas series from range **X** to **Y**.
    Input: Range X and Y is given in single line seperated with a space

    ```
    10 20
    ```

    Ouput:

    0 10

    1 11

    2 12

    3 13

    4 14

    5 15

    6 16

    7 17

    8 18

    9 19

    10 20

    dtype: int32

In [31]:

```
import pandas as pd
import numpy as np
X,Y=map(int,input().split(' '))
series=pd.Series(np.arange(X,Y+1))
print(series)
```

```
10 20
0     10
1     11
2     12
3     13
4     14
5     15
6     16
7     17
8     18
9     19
10    20
dtype: int32
```

32. Write a Python program to add, subtract, multiple and divide two Pandas Series.

Input: Two series in each line

```
1, 2, 3, 4, 5
6, 7, 8, 9, 10
```


Output:

Addition: 0 7

```
1     9
2    11
3    13
4    15
dtype: int64
Substraction:  0   -5
1   -5
2   -5
3   -5
4   -5
dtype: int64
Multiplication:  0     6
1    14
2    24
3    36
4    50
dtype: int64
Division:  0    0.166667
1    0.285714
2    0.375000
3    0.444444
4    0.500000
dtype: float64
```

In [32]:

```
import pandas as pd
l1=map(int,input().split(','))
l2=map(int,input().split(','))
S1=pd.Series(l1)
S2=pd.Series(l2)
print('Addition: ',S1+S2)
print('Substraction: ',S1-S2)
print('Multiplication: ',S1*S2)
print('Division: ',S1/S2)
```

```
1,2,3,4,5
6,7,8,9,10
Addition:  0     7
1     9
2    11
3    13
4    15
dtype: int64
Substraction:  0   -5
1   -5
2   -5
3   -5
4   -5
dtype: int64
Multiplication:  0     6
1    14
2    24
3    36
4    50
dtype: int64
Division:  0    0.166667
1    0.285714
2    0.375000
3    0.444444
4    0.500000
dtype: float64
```


33. From the raw data below create a Pandas Series

```
[ '  Aron', 'Jackson   ', '   Ahree   ', 'Sam']
```

- Print all elements after stripping spaces from the left and right
- Print all the elements after removing spaces from the left only
- Print all the elements after removing spaces from the right only

```
s6a=pd.Series([ '  Aron', 'Jackson   ', '   Ahree   ', 'Sam'])
for i in s6a:
    print(str(i).strip(),end='|')
print('\n')
for i in s6a:
    print(str(i).lstrip(),end='|')
print('\n')
for i in s6a:
    print(str(i).rstrip(),end='|')
```

```
Aron|Jackson|Ahree|Sam|

Aron|Jackson   |Ahree   |Sam|

  Aron|Jackson|   Ahree|Sam|
```

34. Write a program to convert a dictionary into Pandas Series.

    Input:{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}

    Output:

```
Sam      89
Aron     82
Gray     78
Isla     93
Ahree    87
dtype: int64
```


In [34]:

```
import pandas as pd
my_dict={'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}
series=pd.Series(my_dict)
print(series)
```

```
Sam      89
Aron     82
Gray     78
Isla     93
Ahree    87
dtype: int64
```

35. Write a program to convert the below dictionary into DataFrame and print head(2) and tail(2) of the DataFrame.

    Input: {'Name':['Sameer','Leona','Samuel','Jackson','Gray','Sylphia'], 'Class':[11,11,12,12,11,12], 'Age':[17,17,18,19,17,21,23] }

    Output:

1. Write a python script to simulate a dice roll.

Example:

```
Input : Generated by Random method

Output: 6
```

In [1]:

```
import random
num = random.randint(1,6)
print(num)
```

```
3
```

2. Write a python script to debit and credit money from a bank account using functions.

Example:

```
Input: Enter amount to be deposited or withdrawn: 200

Output: Deposit Balance: 200
```


In [2]:

```
def make_account():
    return {'balance': 0}

def deposit(account, amount):
    account['balance'] += amount
    print("Deopsit Balance:",account['balance'])
    return account['balance']

def withdraw(account, amount):
    account['balance'] -= amount
    print("Withdrawal Balance: ",account['balance'])
    return account['balance']

a=make_account()
amt=int(input("Enter amount to be deposited or withdrawn: "))
deposit(a,amt)
```

```
Enter amount to be deposited or withdrawn: 200
Deopsit Balance: 200
```

Out[2]:

```
200
```

3. Write a python script to find difference between two

dates.

Example:

```
Input: Enter a date in YYYY-MM-DD format: 2018-01-02
       Enter a date in YYYY-MM-DD format: 2018-02-02

Output: 31 days
```

In [3]:

```
import datetime

def numOfDays(date1, date2):
    return (date2-date1).days

date1_ent = input('Enter a date in YYYY-MM-DD format: ')
year, month, day = map(int, date1_ent.split('-'))
date1 = datetime.date(year, month, day)
date2_ent = input('Enter a date in YYYY-MM-DD format: ')
year, month, day = map(int, date2_ent.split('-'))
date2 = datetime.date(year, month, day)
print(numOfDays(date1, date2), "days")
```

```
Enter a date in YYYY-MM-DD format: 2019-01-02
Enter a date in YYYY-MM-DD format: 2019-02-02
31 days
```


4. Write a python script to verify Email-ID syntax.

Example:

```
Input: Enter text: john.doe@gmail.com

Output: ['john.doe@gmail.com']
```

In [4]:

```
import re

s=input("Enter text: ")
lst = re.findall('\S+@[a-z]+.com$', s)
print(lst)
```

```
Enter text: john.doe@example.com
['john.doe@example.com']
```

HINT: Generate a password using random() function considering Uppercase, lowercase characters and digits.

Example:

Input: Enter length of password: 10

Output: 7dkjbAf9q0

In [5]:

```
import string,random
n=int(input("Enter length of password: "))
passw= ''.join(random.choice(string.ascii_uppercase + string.ascii_lowercase + string.digit
print(passw)
```

```
Enter length of password: 10
6nNKQ54BnM
```

6. Python code to check the strength of a password.

HINT: Check for a password on the criterion of uppercase, lowercase characters and digits. **Example**:

```
Input: Enter a password

       The password must be between 6 and 12 characters.

       Password: Test123$

Output: Password is Strong
```

In [6]:

```
import re

def password():
    print ('Enter a password\n\nThe password must be between 6 and 12 characters.\n')

    while True:
        password = input('Password: ')
        if 6 <= len(password) < 12:
            break
        print ('The password must be between 6 and 12 characters.\n')

    password_scores = {0:'Horrible', 1:'Weak', 2:'Medium', 3:'Strong'}
    password_strength = dict.fromkeys(['has_upper', 'has_lower', 'has_num'], False)
    if re.search(r'[A-Z]', password):
        password_strength['has_upper'] = True
    if re.search(r'[a-z]', password):
        password_strength['has_lower'] = True
    if re.search(r'[0-9]', password):
        password_strength['has_num'] = True

    score = len([b for b in password_strength.values() if b])

    print ('Password is %s' % password_scores[score])
password()
```



```
Enter a password

The password must be between 6 and 12 characters.

Password: Test123$
Password is Strong
```

7. Write a pythonic code to find square, cube and square root of a number.

HINT: Use lambda functions

Example:

```
Input: Enter a number: 16

Output: Square: 256
        Cube: 4096
        Square Root: 4.0
```

In [7]:

```
import math
def sqrfunc(n):
    return lambda n:n*n
def cubefunc(n):
    return lambda n:n*n*n
def sqrtfunc(n):
    return lambda n: math.sqrt(n)
n= int(input("Enter a number: "))
sqr = sqrfunc(n)
cube= cubefunc(n)
sqrrt=sqrtfunc(n)
print("Square: ",sqr(n))
print("Cube: ",cube(n))
print("Square Root: ",sqrrt(n))
```

```
Enter a number: 16
Square:  256
Cube:  4096
Square Root:  4.0
```

8. Write a Python code to calculate the area of the object based on the parameters given.

Example:

```
Input: This program will calculate/narea of some geometric shapes for you
       Enter Square, Rectangle, Triangle, Circle, or Trapezoid
       What area would you like to calculate? Circle
       Give the radius: 6
Output: Area: 113.09733552923255
```

In [8]:

```
import math

#formulas for each geometric figure
def calc_square(a_side):
    square_area = a_side ** 2
    return square_area

def calc_rectangle(w_side, l_side):
    rect_area = l_side * w_side
    return rect_area

def calc_triangle(base, height):
    triangle_area = (base * height) / 2
    return triangle_area

def calc_circle(radius):
    circle_area = math.pi * radius ** 2
    return circle_area

def calc_trapezoid(short_base, long_base, height):
    trapezoid_area = ((short_base + long_base) / 2 ) * height
    return trapezoid_area

#function determining which formula to calculate
def area_calc_logic(user_calc):
    if user_calc == "square":
        a_side = float(input("Give length of side: "))
        print("Area: ",calc_square(a_side))
    elif user_calc == "rectangle":
        l_side = float(input("Give the length: "))
        w_side = float(input("Give the width: "))
        print("Area: ",calc_rectangle(w_side, l_side))
    elif user_calc == "triangle":
        base = float(input("Give the length of base: "))
        height = float(input("Give the height: "))
        print("Area: ",calc_triangle(base, height))
    elif user_calc == "circle":
        radius = float(input("Give the radius: "))
        print("Area: ",calc_circle(radius))
    elif user_calc == "trapezoid":
        short_base = float(input("Give the length of the short base: "))
        long_base = float(input("Give the length of the long base: "))
        height = float(input("Give the height: "))
        print("Area: ",calc_trapezoid(short_base, long_base, height))
    else:
        area_calc_logic(input("Error, Re-enter input: "))

if __name__ == '__main__':
    print("This program will calculate/narea of some geometric shapes for you")
    print( "Enter Square, Rectangle, Triangle, Circle, or Trapezoid")
    shape=(input("What area would you like to calculate? ")).lower()
    area_calc_logic(shape)
```

```
This program will calculate/narea of some geometric shapes for you
Enter Square, Rectangle, Triangle, Circle, or Trapezoid
What area would you like to calculate? Circle
Give the radius: 6
Area:  113.09733552923255
```

9. Write a python code to Divide a number using try and except.

In [9]:

```
try:
    a = int(input("Enter numerator number: "))
    b = int(input("Enter denominator number: "))
    print("Result of Division: " + str(a/b))

except(ZeroDivisionError):
    print("You have divided a number by zero, which is not allowed.")

except(ValueError):
    print("You must enter integer value")

except:
    print("Oops! Something went wrong!")
```

```
Enter numerator number: 10
Enter denominator number: 0
You have divided a number by zero, which is not allowed.
```


10. Write a python program to check for a file in the current directory.

Hint: Use custom exceptions.

In [10]:

```
import os

class Error(Exception):
   """Base class for other exceptions"""
   pass

class FileNotPresent(Error):
   """Base class for other exceptions"""
   pass

try:
    def find_all(name, path):
        result = []
        for root, dirs, files in os.walk(path):
            if name in files:
                result.append(os.path.join(root, name))
            else:
                raise FileNotPresent
        return result
except(FileNotPresent):
    print("File NOT present")

fname = input("Enter the file name: ")
fpath = input("Enter path to search: ")
find_all(fname,fpath)
```



```
Enter the file name: QB1.docx
Enter path to search: C://

---------------------------------------------------------------------------
FileNotPresent                            Traceback (most recent call last)
<ipython-input-10-a92a3bdb9887> in <module>
     23 fname = input("Enter the file name: ")
     24 fpath = input("Enter path to search: ")
---> 25 find_all(fname,fpath)

<ipython-input-10-a92a3bdb9887> in find_all(name, path)
     16                 result.append(os.path.join(root, name))
     17             else:
---> 18                 raise FileNotPresent
     19         return result
     20 except(FileNotPresent):

FileNotPresent:
```

11. Write a python code to match all MAC addresses given in the input.

In [ ]:

```
import re
mac= input("Enter a MAC address: ")
if re.search("^[a-fA-F0-9:]{17}|[a-fA-F0-9]{12}$", mac):
    print("Valid MAC address")
else:
    print("Not valid address")
```

12. Write a Python function to find the Max of three numbers.

In [ ]:

```
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )

n= int(input("Enter no. of digits to compare: "))
if n==2:
    x=int(input("Enter first number: "))
    y=int(input("Enter second number: "))
    print(max_of_two(x,y))
elif n==3:
    x=int(input("Enter first number: "))
    y=int(input("Enter second number: "))
    z=int(input("Enter third number: "))
    print(max_of_three(x,y,z))
else:
    print("Enter a smaller value of N!")
```


13. Write a Python program to convert degree to radian.

Hint: One degree is rounded off to 57.3 radians when arc length is the same.

Input : 15

Output: 0.2617993877991494

In [ ]:

```
import math
degree = float(input("Input degrees: "))
radian = degree*(math.pi/180)
print("Radian value is: ",radian)
```

14. Write a Python function to check whether a string is a pangram or not.

Hint: Pangrams are words or sentences containing every letter of the alphabet at least once.

```
Input:  "The quick brown fox jumps over the lazy dog"

Output:  TRUE
```

In [ ]:

```
import string, sys
def ispangram(str1, alphabet=string.ascii_lowercase):
    alphaset = set(alphabet)
    return alphaset <= set(str1.lower())

pan=input("Enter a sentence: ")
print ( ispangram(pan))
```

15. Write a Python class to convert an integer to a roman numeral.

```
Input: 5

Ouput: V
```


In [ ]:

```
class py_Roman:
    def int_to_Roman(self, num):
        val = [
            1000, 900, 500, 400,
            100, 90, 50, 40,
            10, 9, 5, 4,
            1
            ]
        syb = [
            "M", "CM", "D", "CD",
            "C", "XC", "L", "XL",
            "X", "IX", "V", "IV",
            "I"
            ]
        roman_num = ''
        i = 0
        while  num > 0:
            for _ in range(num // val[i]):
                roman_num += syb[i]
                num -= val[i]
            i += 1
        return roman_num

num= int(input("Enter a number: "))
print("Roman numeral: ",py_Roman().int_to_Roman(num))
```

16. Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.

```
Input: numbers= [10,20,10,40,50,60,70], target=90

Output: index1 = 3; index2 = 4
```

In [ ]:

```
class py_solution:
   def twoSum(self, nums, target):
        lookup = {}
        for i, num in enumerate(nums):
            if target - num in lookup:
                return (lookup[target - num], i )
            lookup[num] = i

target=int(input("Enter target sum: "))
print("index1=%d, index2=%d" % py_solution().twoSum((10,20,10,40,50,60,70),target))
```

17. Write a Python program to get the class name of an instance in Python.


```
Input: Instance of class

Output: Class Name

As seen below
```

In [ ]:

```
inst=input("Enter instance: ")
inst.__class__.__name__
```

18. Write a Python program to calculate the Discriminant value of a quadratic equation.

```
Input: x: 10
       y: 5
       z: 20

Output: No real Solution. Discriminant value is -775.0
```

In [ ]:

```
def discriminant():
    x_value = float(input('The x value: '))
    y_value = float(input('The y value: '))
    z_value = float(input('The z value: '))
    discriminant = (y_value**2) - (4*x_value*z_value)
    if discriminant > 0:
        print('Two Solutions. Discriminant value is:', discriminant)
    elif discriminant == 0:
        print('One Solution. Discriminant value is:', discriminant)
    elif discriminant < 0:
        print('No Real Solutions. Discriminant value is:', discriminant)


discriminant()
```

19. Write a Python program to detect the number of local variables declared in a function.

In [ ]:


```
def abc():
    x = 1
    y = 2
    str1= "This is a test example."
    print("Python Exercises")

print(abc.__code__.co_nlocals)
```

20. Write a Python program to get last modified information of a file.

In [ ]:

```
import os, time
def last_modified_fileinfo(filepath):
    filestat = os.stat(filepath)
    date = time.localtime((filestat.st_mtime))
    # Extract year, month and day from the date
    year = date[0]
    month = date[1]
    day = date[2]
    # Extract hour, minute, second
    hour = date[3]
    minute = date[4]
    second = date[5]
    # Year
    strYear = str(year)[0:]
    # Month
    if (month <=9):
        strMonth = '0' + str(month)
    else:
        strMonth = str(month)
    # Date
    if (day <=9):
        strDay = '0' + str(day)
    else:
        strDay = str(day)

    return (strYear+"-"+strMonth+"-"+strDay+" "+str(hour)+":"+str(minute)+":"+str(second))
print()
print(last_modified_fileinfo('878_m4_slides_v1.0.pptx'))
# Enter a file present in the same folder, else mention the absolute path.
print()
```


21. Write a Python program to print a 3-column calendar for an entire year.

In [ ]:

```
import calendar
cal = calendar.TextCalendar(calendar.SUNDAY)
# column width: 2, lines per week: 1
# number of spaces between month columns: 1
# 3: no. of months per column.
year=int(input("Enter the year: "))
print(cal.formatyear(year, 2, 1, 1, 3))
```

22. Python Program to print all permutations of a given string.

Example:

```
Input: ABC

Output: ABC
        ACB
        BAC
        BCA
        CBA
        CAB
```

In [ ]:

```
def toString(List):
    return ''.join(List)
# Function to print permutations of string
# This function takes three parameters:
# 1. String
# 2. Starting index of the string
# 3. Ending index of the string.
def permute(a, l, r):
    if l == r:
        print(toString(a))
    else:
        for i in range(l, r + 1):
            a[l], a[i] = a[i], a[l]
            permute(a, l + 1, r)
            a[l], a[i] = a[i], a[l] # backtrack

string = input("Enter a string: ")
n = len(string)
a = list(string)
permute(a, 0, n-1)
```


23. Write Pythonic code to simulate distribution of a deck of cards.

The Deck class should have a deal method to deal a card from the deck
After a card is dealt, it is removed from the deck.
There should be a shuffle method which makes sure the deck of cards has all 52 cards and then rearranges them randomly.
The Card class should have a suit (Hearts, Diamonds, Clubs, Spades) and a value (A,2,3,4,5,6,7,8,9,10,J,Q,K)

In [ ]:

```
import random
def new_deck():
    """
    create a deck of cards
    suit: club=C, diamond=D, heart=H spade=S
    rank: ace=A, 10=T, jack=J, queen=Q, king=K, numbers=2..9
    ace of spade would be AS, 8 of heart would be 8H and so on
    return a list of a full deck of cards
    """
    rs = [rank + suit for rank in "A23456789TJQK" for suit in "CDHS"]
    return rs
def draw_cards(n, cards_list):
    """
    randomly draw n cards from the deck (cards_list)
    remove those cards from the deck
    since object cards_list is by reference, it will change too
    return a list of n cards
    """
    random.shuffle(cards_list)
    return [cards_list.pop() for k in range(n)]
# new deck
cards_list = new_deck()
print("New deck = %s cards" % len(cards_list))  # test
# draw n cards per hand
n = 5
# draw the hands
hand1 = draw_cards(n, cards_list)
hand2 = draw_cards(n, cards_list)
print('-'*40)
# show the 2 hands
print("hand1 = %s" % hand1)
print("hand2 = %s" % hand2)
print('-'*40)
print("New deck = %s cards" % len(cards_list))  # test
```


24. Write a pythonic method which can calculate square value of number without using arithmetic operators or built in functions.

HINT: use ** operator.

In [ ]:

```
class Square:
    def square(num):
        return num ** 2

n=int(input("Enter number: "))
sq=Square.square(n)
print("Square of number is: ",sq)
```

25. Write a program to show the concept of polymorphism in Python.

In [ ]:

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")

    def language(self):
        print("Hindi the primary language of India.")

    def type(self):
        print("India is a developing country.")

class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

    def language(self):
        print("English is the primary language of USA.")

    def type(self):
        print("USA is a developed country.")

def func(obj):
    obj.capital()
    obj.language()
    obj.type()

obj_ind = India()
obj_usa = USA()

func(obj_ind)
func(obj_usa)
```


26. Create a Python program of a Magic 8 Ball which is a toy used for fortune-telling.

Allow the user to input their question. Show an in progress message. Create 10 responses, and show a random response. Allow the user to ask another question/advice or quit the game.

In [ ]:

```
import random
answers = ['It is certain', 'It is decidedly so', 'Without a doubt', 'Yes – definitely', 'Y
print('Hello World, I am the Magic 8 Ball, What is your name?')
name = input()
print('Hello ' + name)

def Magic8Ball():
    print('Ask me a question.')
    input()
    print (answers[random.randint(0, len(answers)-1)] )
    print('I hope that helped!')
    Replay()


def Replay():
    print ('Do you have another question? [Y/N] ')
    reply = input()
    if reply == 'Y':
        Magic8Ball()
    elif reply == 'N':
        print("Adios!")
        exit()
    else:
        print('I apologies, I did not catch that. Please repeat.')
        Replay()

Magic8Ball()
```


27. Write a Python program to find whether it contains an additive sequence or not.

NOTE: The additive sequence is a sequence of numbers where the sum of the first two numbers is equal to the third one. Leading zeros cannot be included.

Example:

Input: 66121830

Output: True

In [ ]:

```
class Solution(object):
# DFS: iterative implement.
    def is_additive_number(self, num):
        length = len(num)
        for i in range(1, int(length/2+1)):
            for j in range(1, int((length-i)/2 + 1)):
                first, second, others = num[:i], num[i:i+j], num[i+j:]
                if self.isValid(first, second, others):
                    return True
        return False

    def isValid(self, first, second, others):
        if ((len(first) > 1 and first[0] == "0") or
                (len(second) > 1 and second[0] == "0")):
            return False
        sum_str = str(int(first) + int(second))
        if sum_str == others:
            return True
        elif others.startswith(sum_str):
            return self.isValid(second, sum_str, others[len(sum_str):])
        else:
            return False

if __name__ == "__main__":
    seq=input("Enter Sequence: ")
    print(Solution().is_additive_number(seq))
```


28. Demonstrate Inheritance in python using an employee example.

In [ ]:

```
class employee:
    num_employee=0
    raise_amount=1.04
    def __init__(self, first, last, sal):
        self.first=first
        self.last=last
        self.sal=sal
        self.email=first + '.' + last + '@company.com'
        employee.num_employee+=1
    def fullname (self):
        return '{} {}'.format(self.first, self.last)
    def apply_raise (self):
        self.sal=int(self.sal* raise_amount)

class developer(employee):
    raise_amount = 1.10
    def __init__(self, first, last, sal, prog_lang):
        super().__init__(first, last, sal)
        self.prog_lang=prog_lang

class sales(employee):
    raise_amount = 1.01
    def __init__(self, first, last, sal, course_assign):
        super().__init__(first, last, sal)
        self.course_assign=course_assign

emp_1=developer('Guido', 'van Rossum', 1000000, 'Python')
emp_2=developer('Ani', 'Sri', 10000, 'Hadoop')
print(emp_1.prog_lang)
print(emp_2.sal)
print(emp_2.fullname)
```

29. Develop a Pythonic code to demonstrate the concept of Abstraction.

In [ ]:

```
from abc import ABC, abstractmethod

class Polygon(ABC):

    # abstract method
    def noofsides(self):
        pass

class Triangle(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 3 sides")

class Pentagon(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 5 sides")

class Hexagon(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 6 sides")

class Quadrilateral(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 4 sides")

# Driver code
R = Triangle()
R.noofsides()

K = Quadrilateral()
K.noofsides()

R = Pentagon()
R.noofsides()

K = Hexagon()
K.noofsides()
```


30. Write a Python program to implement pow(x, n) without using built-in functions.

In [ ]:

```
class py_solution:
   def pow(self, x, n):
        if x==0 or x==1 or n==1:
            return x

        if x==-1:
            if n%2 ==0:
                return 1
            else:
                return -1
        if n==0:
            return 1
        if n<0:
            return 1/self.pow(x,-n)
        val = self.pow(x,n//2)
        if n%2 ==0:
            return val*val
        return val*val*x

print(py_solution().pow(2, -3));
print(py_solution().pow(3, 5));
print(py_solution().pow(100, 0));
```


31. Write a python program for a simple calculator.

In [ ]:

```
def add(x, y):
   return x + y
# This function subtracts two numbers
def subtract(x, y):
   return x - y
# This function multiplies two numbers
def multiply(x, y):
   return x * y
# This function divides two numbers
def divide(x, y):
   return x / y
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
# Take input from the user
choice = input("Enter choice(1/2/3/4):")
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if choice == '1':
   print(num1,"+",num2,"=", add(num1,num2))
elif choice == '2':
   print(num1,"-",num2,"=", subtract(num1,num2))
elif choice == '3':
   print(num1,"*",num2,"=", multiply(num1,num2))
elif choice == '4':
   print(num1,"/",num2,"=", divide(num1,num2))
else:
   print("Invalid input")
```


32. Write a Python class which has two methods get_String and print_String. get_String accept a string from the user and print_String print the string in upper case.

In [ ]:

```
class IOString():
    def __init__(self):
        self.str1 = ""

    def get_String(self):
        self.str1 = input()

    def print_String(self):
        print(self.str1.upper())

str1 = IOString()
str1.get_String()
str1.print_String()
```

1. Write a Python program to multiply two matrices using numpy.

Example:

```
Input: [1, 6, 5],[3 ,4, 8],[2, 12, 3]
       [3, 4, 6],[5, 6, 7],[6,56, 7]

Output: [[ 63 320  83]
         [ 77 484 102]
         [ 84 248 117]]
```

In [1]:

```
import numpy as np

# input two matrices
mat1 = ([1, 6, 5],[3 ,4, 8],[2, 12, 3])
mat2 = ([3, 4, 6],[5, 6, 7],[6,56, 7])
# This will return dot product
res = np.dot(mat1,mat2)

print(res)
```



```
[[ 63 320  83]
 [ 77 484 102]
 [ 84 248 117]]
```

2. Write a Python program to get the floor, ceiling and truncated values of the elements of an numpy array.

Example:

```
Input: [-1.6, -1.5, -0.3, 0.1, 1.4, 1.8, 2.0]

Output: Floor values of the above array elements:
        [-2. -2. -1.  0.  1.  1.  2.]
        Ceiling values of the above array elements:
        [-1. -1. -0.  1.  2.  2.  2.]
        Truncated values of the above array elements:
        [-1. -1. -0.  0.  1.  1.  2.]
```

In [2]:

```
import numpy as np
x = np.array([-1.6, -1.5, -0.3, 0.1, 1.4, 1.8, 2.0])
print("Original array:")
print(x)
print("Floor values of the above array elements:")
print(np.floor(x))
print("Ceiling values of the above array elements:")
print(np.ceil(x))
print("Truncated values of the above array elements:")
print(np.trunc(x))
```

```
Original array:
[-1.6 -1.5 -0.3  0.1  1.4  1.8  2. ]
Floor values of the above array elements:
[-2. -2. -1.  0.  1.  1.  2.]
Ceiling values of the above array elements:
[-1. -1. -0.  1.  2.  2.  2.]
Truncated values of the above array elements:
[-1. -1. -0.  0.  1.  1.  2.]
```

3. Write a Python program to find the inverse of a matrix.

Example:


```
Input: [[2  3]
        [4  5]]

Output: [[-2.5  1.5]
         [ 2.  -1. ]]
```

In [3]:

```
import numpy as np
arr = np.array([[2,3],[4,5]])
try:
    inverse = np.linalg.inv(arr)
    print(inverse)
except numpy.linalg.LinAlgError:
    # Not invertible. Skip this one.
    pass
```

```
[[-2.5  1.5]
 [ 2.  -1. ]]
```

4. Write a Python program to perform addition, subtraction, multiplication and divsion on the given polynomials.

Example:

```
Input: x = (10,20,30)
       y = (30,40,50)

Output: Addition:
        [40. 60. 80.]
        Subtraction:
        [-20. -20. -20.]
        Multiplication:
        [ 300. 1000. 2200. 2200. 1500.]
        Division:
        (array([0.6]), array([-8., -4.]))
```

In [4]:

```
from numpy.polynomial import polynomial as P
x = (10,20,30)
y = (30,40,50)
print("Addition:")
print(P.polyadd(x,y))
print("Subtraction:")
print(P.polysub(x,y))
print("Multiplication:")
print(P.polymul(x,y))
print("Division:")
print(P.polydiv(x,y))
```



```
Addition:
[40. 60. 80.]
Subtraction:
[-20. -20. -20.]
Multiplication:
[ 300. 1000. 2200. 2200. 1500.]
Division:
(array([0.6]), array([-8., -4.]))
```

5. Write a Python program to create a random array with N elements and compute the average, variance, standard deviation of the array elements.

Example:

```
Input: 100

Output: Average of the array elements:
            -0.006971318946870028
        Standard deviation of the array elements:
            0.9249592403302502
        Variance of the array elements:
            0.8555495962723136
```

In [5]:

```
import numpy as np
n=int(input("Enter a number: "))
x = np.random.randn(n)
print("Average of the array elements:")
mean = x.mean()
print("\t",mean)
print("Standard deviation of the array elements:")
std = x.std()
print("\t",std)
print("Variance of the array elements:")
var = x.var()
print("\t",var)
```

```
Enter a number: 100
Average of the array elements:
         0.05880255975537995
Standard deviation of the array elements:
         1.1148959806917216
Variance of the array elements:
         1.2429930477625555
```

6. Write a Python program to compute the reciprocal for all elements in a given array.

Example:

Input: [1. 2. 0.2 0.3]

Output: [1. 0.5 5. 3.33333333]

In [6]:

```
import numpy as np
x = np.array([1., 2., 0.4, .3])
print("Original array: ")
print(x)
r1 = np.reciprocal(x)
r2 = 1/x
assert np.array_equal(r1, r2)
print("Reciprocal for all elements of the said array:")
print(r1)
```

```
Original array:
[1.  2.  0.4 0.3]
Reciprocal for all elements of the said array:
[1.         0.5        2.5        3.33333333]
```

7. Write a Python program to sort the specified number of elements from beginning of a given array.

Input: Enter the number: 3

Output: Original array:
[0.214759 0.80778318 0.54766704 0.65261567 0.04178861 0.81868463
0.82267499 0.55487551 0.17285421 0.22954424]

Sorted first N elements:
[0.04178861 0.17285421 0.214759 0.65261567 0.54766704 0.81868463
0.82267499 0.55487551 0.80778318 0.22954424]

In [7]:

```
import numpy as np
nums =  np.random.rand(10)
print("Original array:")
print(nums)
n= int(input("Enter the number: "))
print("\nSorted first N elements:")
print(nums[np.argpartition(nums,range(n))])
```

```
Original array:
[0.3832014  0.53239101 0.88468573 0.72025648 0.85211553 0.95436835
 0.39335353 0.61034102 0.92221283 0.96553183]
Enter the number: 3

Sorted first N elements:
[0.3832014  0.39335353 0.53239101 0.72025648 0.85211553 0.95436835
 0.88468573 0.61034102 0.92221283 0.96553183]
```


8. Write a Python program to generate N random numbers from the normal distribution.

Example:

Input: 7

Output: [1.25127475 -1.40593623 -0.84415004 0.35449771 -1.46282713 -0.316080
52
1.36096266]

In [8]:

```
import numpy as np
n=int(input("Entyer number: "))
x = np.random.normal(size=n)
print(x)
```

```
Entyer number: 7
[-1.97660116 -0.87551402  0.61796678 -0.72829323  1.14098002 -0.75746754
  0.98867339]
```

9. Write a Python program to create random vector of size N and replace the maximum value by N.

Example:

```
Input: 5

Output: Original array:
        [0.35354479 0.90990917 0.36123112 0.51039696 0.40866078]
        Maximum value replaced by 5:
        [ 0.35354479 5.         0.36123112  0.51039696  0.40866078]
```

In [9]:

```
import numpy as np
n=int(input("Enter value of N: "))
x = np.random.random(n)
print("Original array:")
print(x)
x[x.argmax()] = n
print("Maximum value replaced by -1:")
print(x)
```

```
Enter value of N: 5
Original array:
[0.77044094 0.76291801 0.37463723 0.24472369 0.76049477]
Maximum value replaced by -1:
[5.         0.76291801 0.37463723 0.24472369 0.76049477]
```


10. Write a Python program to find the most frequent value in an array.

Example:

```
Input: *NO user input*

Output: Original array:
            [9 1 5 3 2 7 3 1 3 7 4 1 5 5 8 5 6 7 3 3 9 6 4 0 0 6 6 9 8 1 0 7 5
   7 6 3 3
             1 9 2]
            Most frequent value in the above array:
            3
```

In [10]:

```
import numpy as np
x = np.random.randint(0, 10, 40)
print("Original array:")
print(x)
print("Most frequent value in the above array:")
print(np.bincount(x).argmax())
```

```
Original array:
[9 4 6 8 4 0 4 8 5 7 2 5 2 0 0 2 8 9 4 5 2 5 6 5 3 6 8 8 8 9 9 2 7 6 2 1 8
 8 4 4]
Most frequent value in the above array:
8
```

11. Write a Python program to convert cartesian coordinates to polar coordinates of a random MxN matrix representing cartesian coordinates.

Example:

Input: Enter no. of rows: 10
Enter no. of columns: 5

Output: Cartesian Product:
[0.99328988 1.05908468 0.73245748 0.59807022 1.08785099 0.22876169
0.43354143 0.54565744 1.24634938 0.85174826]
Polar Product:
[0.33842207 0.84472275 0.89358263 0.86716174 0.92161633 0.79728956
0.74207104 0.88689297 0.78518776 1.54633146]

In [11]:

```
import numpy as np
m=int(input("Enter no. of rows: "))
n=int(input("Enter no. of columns: "))
z= np.random.random((m,n))
x,y = z[:,0], z[:,1]
r = np.sqrt(x**2+y**2)
t = np.arctan2(y,x)
print("Cartesian Product: \n",r)
print("Polar Product: \n",t)
```

```
Enter no. of rows: 10
Enter no. of columns: 5
Cartesian Product:
 [0.54677941 1.15446811 1.25063677 0.46438923 1.01691456 1.15472981
 0.65239162 0.87020004 0.21183617 0.91507508]
Polar Product:
 [0.84098257 1.0035094  0.72730818 1.51968856 0.83152064 0.66755084
 1.52922192 1.4607694  0.39684955 0.48050032]
```

Example:

```
Input: Lower limit: 5
       Upper limit: 20

Output: [10.  15.  18.  17.5 13.   5.   9.  19.5 12.5 16.  10.5  7.5  5.5 17.
         15.5 12.   7.   8.  11.  18.5  6.5  9.5  8.5 14.5 16.5 13.5 19.   6.
         14.  11.5]
        Same result using permutation():
         [ 7 11  9  2  4  1 10  5  6 14 13  8  3 12  0]
```

In [12]:

```
import numpy as np
x=int(input("Lower limit: "))
y=int(input("Upper limit: "))
s = np.arange(x,y,0.5)
np.random.shuffle(s)
print(s)
print("Same result using permutation():")
print(np.random.permutation((y-x)))
```

```
Lower limit: 5
Upper limit: 20
[ 7.5 17.  13.   6.  15.   8.   5.5 16.5 19.  18.5  5.  16.   7.  14.5
 15.5 13.5  9.5 11.5 10.5 12.   6.5 17.5  8.5 14.   9.  19.5 10.  12.5
 11.  18. ]
Same result using permutation():
[ 9  5 10  7  3  8 13  6  0 12 14  4  2 11  1]
```


13. Write a NumPy program to create a 3x3x3 array with random values.

Example:

```
Output: [[[0.95755799 0.8894923  0.21848393]
          [0.58257729 0.94365754 0.69440265]
          [0.95108699 0.63190746 0.55467339]]

         [[0.23291382 0.44131661 0.3366771 ]
          [0.47991351 0.35187551 0.14913956]
          [0.95231571 0.6708149  0.46795982]]

         [[0.15628735 0.46414151 0.45751118]
          [0.93117854 0.57598345 0.01820238]
          [0.87935621 0.06270413 0.78463814]]]
```

In [13]:

```
import numpy as np
x = np.random.random((3,3,3))
print(x)
```

```
[[[0.18065872 0.37893368 0.09349314]
  [0.06108608 0.86903899 0.93442426]
  [0.90325783 0.28556145 0.43784506]]

 [[0.98673116 0.00488452 0.2305979 ]
  [0.12712543 0.88151164 0.0117851 ]
  [0.67366313 0.36919219 0.67684042]]

 [[0.89450687 0.4405461  0.63276481]
  [0.66617057 0.54027866 0.94311493]
  [0.1797432  0.17777204 0.70851702]]]
```

14. Write a Python program to find point by point distances of a random vector with shape (J,K) representing coordinates.

Example:


```
Input: Enter J: 5
       Enter K: 5

Output: [[0.         0.96302667 0.23555705 0.31258243 0.34680981]
         [0.96302667 0.         0.80848588 0.70342558 0.77484355]
         [0.23555705 0.80848588 0.         0.10533739 0.42407756]
         [0.31258243 0.70342558 0.10533739 0.         0.4078191 ]
         [0.34680981 0.77484355 0.42407756 0.4078191  0.        ]]
```

In [14]:

```
import numpy as np
j=int(input("Enter J: "))
k=int(input("Enter K: "))
a= np.random.random((j,k))
x,y = np.atleast_2d(a[:,0], a[:,1])
d = np.sqrt( (x-x.T)**2 + (y-y.T)**2)
print(d)
```

```
Enter J: 5
Enter K: 5
[[0.         0.4599731  0.3801513  0.33515084 0.38078576]
 [0.4599731  0.         0.83984384 0.794319   0.6477403 ]
 [0.3801513  0.83984384 0.         0.06802964 0.47195246]
 [0.33515084 0.794319   0.06802964 0.         0.4842152 ]
 [0.38078576 0.6477403  0.47195246 0.4842152  0.        ]]
```

15. Write a NumPy program to check if each element of a given array is composed of digits only, lower case letters only and upper case letters only.

Example:

```
Input: Original Array:
       ['Python' 'PHP' 'JS' 'Examples' 'html5' '5']

Output: Digits only = [False False False False False  True]
        Lower cases only = [False False False False  True False]
        Upper cases only = [False  True  True False False False]
```

In [15]:

```
import numpy as np
x = np.array(['Python', 'PHP', 'JS', 'Examples', 'html5', '5'], dtype=np.str)
print("\nOriginal Array:")
print(x,"\n")
r1 = np.char.isdigit(x)
r2 = np.char.islower(x)
r3 = np.char.isupper(x)
print("Digits only =", r1)
print("Lower cases only =", r2)
print("Upper cases only =", r3)
```

```
Original Array:
['Python' 'PHP' 'JS' 'Examples' 'html5' '5']

Digits only = [False False False False False  True]
Lower cases only = [False False False False  True False]
Upper cases only = [False  True  True False False False]
```


16. Write a Python program to Triangulate a location based on co-ordinates.

Hint: Centroid of Triangle.
Example:

```
Input: [20.0497520, 31.39864012947, 12.30974023]

Output: 21.25271078649 CSV
```

In [16]:

```
import numpy as np

data = [20.0497520, 31.39864012947, 12.30974023]
print(np.mean(data, axis=0))
```

```
21.25271078649
```

17. Write a Python program to evaluate Einstein's summation convention of two given multidimensional arrays.

HINT: Use inbuilt attribute einsum. Example:

```
Input: Original 1-d arrays:
       [1 2 3]
       [0 1 0]

Output: Einstein's summation convention of the said arrays:
            2
        Original Higher dimension:
            [[0 1 2]
             [3 4 5]
             [6 7 8]]
            [[ 3  4  5]
             [ 6  7  8]
             [ 9 10 11]]
        Einstein's summation convention of the said arrays:
            [[ 24  27  30]
             [ 78  90 102]
             [132 153 174]]
```

In [17]:

```
import numpy as np
a = np.array([1,2,3])
b = np.array([0,1,0])
print("Original 1-d arrays:")
print(a)
print(b)
result =  np.einsum("n,n", a, b)
print("Einstein's summation convention of the said arrays:")
print(result)
x = np.arange(9).reshape(3, 3)
y = np.arange(3, 12).reshape(3, 3)
print("Original Higher dimension:")
print(x)
print(y)
result = np.einsum("mk,kn", x, y)
print("Einstein's summation convention of the said arrays:")
print(result)
```

```
Original 1-d arrays:
[1 2 3]
[0 1 0]
Einstein's summation convention of the said arrays:
2
Original Higher dimension:
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
Einstein's summation convention of the said arrays:
[[ 24  27  30]
 [ 78  90 102]
 [132 153 174]]
```


18. Write a Python code to determine the rank of the matrix.

Example:

Input: [[1,3,7],[2,8,3],[7,8,1]]

Output: 3

In [18]:

```
import numpy

A = numpy.matrix([[1,3,7],[2,8,3],[7,8,1]])
numpy.linalg.matrix_rank(A)
```

Out[18]:

```
3
```

19. Write a Python program compute the condition number of a given matrix.

HINT: In the field of numerical analysis, the condition number of a function with respect to an argument measures how much the output value of the function can change for a small change in the input argument. This is used to measure how sensitive a function is to changes or errors in the input, and how much error in the output results from an error in the input.

Example:

```
Input: [[1 2]
        [3 4]]

Output: Condition number of the said matrix:
            14.933034373659268
```

In [19]:

```
import numpy as np
m = np.array([[1,2],[3,4]])
print("Original matrix:")
print(m)
result =  np.linalg.cond(m)
print("Condition number of the said matrix:")
print(result)
```

```
Original matrix:
[[1 2]
 [3 4]]
Condition number of the said matrix:
14.933034373659268
```

20. Write a NumPy program to multiply a MxN matrix by a NxA matrix and create a real matrix product.

Example:

```
Output: First array:
            [[0.94293584 0.8091474  0.72330868]
             [0.91143684 0.54976631 0.37547562]
             [0.3656866  0.94185543 0.98414967]
             [0.81470666 0.80629404 0.46326721]
             [0.40648925 0.70615124 0.4786933 ]]
        Second array:
            [[0.45507569 0.53265048]
             [0.41761748 0.87547934]
             [0.83832095 0.79087463]]
        Dot product of two arrays:
            [[1.37338609 1.78269354]
             [0.95913385 1.26374046]
             [1.38478366 1.79769712]
             [1.09584228 1.50623395]
             [0.8811831  1.21332391]]
```

In [20]:

```
import numpy as np
m=int(input("Enter value M: "))
n=int(input("Enter value N: "))
a=int(input("Enter value A: "))
x = np.random.random((m,n))
print("First array:")
print(x)
y = np.random.random((n,a))
print("Second array:")
print(y)
z = np.dot(x, y)
print("Dot product of two arrays:")
print(z)
```

```
Enter value M: 5
Enter value N: 3
Enter value A: 2
First array:
[[0.22047904 0.33834373 0.70586539]
 [0.48617698 0.98741379 0.8698782 ]
 [0.56179925 0.89656768 0.52160425]
 [0.02549242 0.44727029 0.99117347]
 [0.2702744  0.11856721 0.0060627 ]]
Second array:
[[0.192595   0.56919562]
 [0.0017944  0.34148961]
 [0.10039919 0.16494603]]
Dot product of two arrays:
[[0.1139386  0.35746627]
 [0.18274214 0.75740432]
 [0.16217717 0.71197877]
 [0.10522531 0.33073846]
 [0.05287494 0.1953285 ]]
```

21. Write a Python program to show different methods of converting python structures into Dataframes.

In [21]:

```
import pandas
listx = [10, 20, 30, 40]
table = pandas.DataFrame(listx)
print("List to dataframes")
print(table)
print()

data = [{'a':1, 'b':2}, {'a':2, 'b':4, 'c':8}]
table = pandas.DataFrame(data)
print("Dictionaries to Dataframe")
print(table)
print()

data = [{'a':1, 'b':2}, {'a':2, 'b':4, 'c':8}]
table = pandas.DataFrame(data, index=[ 'first', 'second' ])
print("List of Dictionaries to dataframe")
print(table)
print()

data = {'one': pandas.Series([1, 2, 3], index=['a', 'b', 'c']),'two': pandas.Series([1, 2,
table = pandas.DataFrame(data)
print("Dictionary of series to Dataframe")
print(table)
print()
```



```
List to dataframes
    0
0  10
1  20
2  30
3  40

Dictionaries to Dataframe
   a  b    c
0  1  2  NaN
1  2  4  8.0

List of Dictionaries to dataframe
        a  b    c
first   1  2  NaN
second  2  4  8.0

Dictionary of series to Dataframe
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
```

22. Write a Python program to convert a NumPy array to

a Pandas series.

Example:

```
Input: NumPy array:
        [10 20 30 40 50]

Output: Converted Pandas series:
            0    10
            1    20
            2    30
            3    40
            4    50
```

In [22]:

```
import numpy as np
import pandas as pd
np_array = np.array([10, 20, 30, 40, 50])
print("NumPy array:")
print(np_array)
print()
new_series = pd.Series(np_array)
print("Converted Pandas series:")
print(new_series)
```



```
NumPy array:
[10 20 30 40 50]

Converted Pandas series:
0    10
1    20
2    30
3    40
4    50
dtype: int32
```

23. Write a program to add and delete columns to a Dataframe when a dictionary is provided.

Example:

```
    Input: data = {'one':([1, 2, 3], index=['a', 'b', 'c']),'two':([1, 2, 3, 4], i
ndex=['a', 'b', 'c', 'd'])}
```

In [23]:

```
import pandas

data = {'one': pandas.Series([1, 2, 3], index=['a', 'b', 'c']),
        'two': pandas.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

table = pandas.DataFrame(data)
print("Before adding a column: ")
print(table)
print()
table['three'] = pandas.Series([10, 20, 30], index=['a', 'b', 'c'])
print("After adding a column: ")
print(table)
print()

print("Deleting column three: ")
del table['three']
print(table)
print()

print("Deleting column one: ")
table.pop('one')
print(table)
```

```
Before adding a column:
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4

After adding a column:
   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN

Deleting column three:
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4

Deleting column one:
   two
a    1
b    2
c    3
d    4
```


24. Write a program to add and delete rows to a Dataframe when a dictionary is provided.

Example:

Input: data = {'one':([1, 2, 3], index=['a', 'b', 'c']),'two':([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

In [24]:

```
import pandas
data = {'one': pandas.Series([1, 2, 3], index=['a', 'b', 'c']),
        'two': pandas.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
table = pandas.DataFrame(data)
table['three'] = pandas.Series([10, 20, 30], index=['a', 'b', 'c'])
print(table)
print()
row  = pandas.DataFrame([[11, 13], [17, 19]], columns=['two', 'three'])
table = table.append(row, ignore_index=True)
print(table)
print()

r=int(input("Enter Row no. to drop: "))
table = table.drop(r)
print(table)
```

```
c:\users\nikhilmeduri\appdata\local\programs\python\python37-32\lib\site-pac
kages\pandas\core\frame.py:7123: FutureWarning: Sorting because non-concaten
ation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  sort=sort,

   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN

   one  three  two
0  1.0   10.0    1
1  2.0   20.0    2
2  3.0   30.0    3
3  NaN    NaN    4
4  NaN   13.0   11
5  NaN   19.0   17

Enter Row no. to drop: 4
   one  three  two
0  1.0   10.0    1
1  2.0   20.0    2
2  3.0   30.0    3
3  NaN    NaN    4
5  NaN   19.0   17
```

25. Write a Python program to import from JSON file

In [25]:

```
import json
import csv

employee_data = '{"employee_details":[{"employee_name": "James", "email": "james@gmail.com"
employee_parsed = json.loads(employee_data)
emp_data = employee_parsed['employee_details']
# open a file for writing
employ_data = open('EmployeeData.csv', 'w')
# create the csv writer object
csvwriter = csv.writer(employ_data)
count = 0
for emp in emp_data:
      if count == 0:
              header = emp.keys()
              csvwriter.writerow(header)
              count += 1
      csvwriter.writerow(emp.values())
employ_data.close()
print("DONE")
```

```
DONE
```


26. Write a Python program to add, subtract, multiple and divide two Pandas Series.

In [26]:

```
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 9])
ds = ds1 + ds2
print("Add two Series:")
print(ds)
print()
print("Subtract two Series:")
ds = ds1 - ds2
print(ds)
print()
print("Multiply two Series:")
ds = ds1 * ds2
print(ds)
print()
print("Divide Series1 by Series2:")
ds = ds1 / ds2
print(ds)
print()
```

```
Add two Series:
0     3
1     7
2    11
3    15
4    19
dtype: int64

Subtract two Series:
0    1
1    1
2    1
3    1
4    1
dtype: int64

Multiply two Series:
0     2
1    12
2    30
3    56
4    90
dtype: int64

Divide Series1 by Series2:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```


27. Write a Python program to change the data type of given a column or a Series.

In [27]:

```
import pandas as pd
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print()
print("Change the said data type to numeric:")
s2 = pd.to_numeric(s1, errors='coerce')
print(s2)
```

```
Original Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object

Change the said data type to numeric:
0    100.00
1    200.00
2       NaN
3    300.12
4    400.00
dtype: float64
```


28. Write a Python program to create a subset of a given series based on value and condition.

In [28]:

```
import pandas as pd
s = pd.Series([0, 1,2,3,4,5,6,7,8,9,10])
print("Original Data Series:")
print(s)
print("\nSubset of the above Data Series:")
n=int(input("Enter N: "))
new_s = s[s < n]
print(new_s)
```

```
Original Data Series:
0      0
1      1
2      2
3      3
4      4
5      5
6      6
7      7
8      8
9      9
10    10
dtype: int64

Subset of the above Data Series:
Enter N: 7
0    0
1    1
2    2
3    3
4    4
5    5
6    6
dtype: int64
```


29. Write a Python program to create the mean and standard deviation of the data of a given Series.

In [29]:

```
import pandas as pd
s = pd.Series(data = [1,2,3,4,5,6,7,8,9,5,3])
print("Original Data Series:")
print(s)
print("\nMean of the said Data Series:")
print(s.mean())
print("\nStandard deviation of the said Data Series:")
print(s.std())
```

```
Original Data Series:
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9     5
10    3
dtype: int64

Mean of the said Data Series:
4.818181818181818

Standard deviation of the said Data Series:
2.522624895547565
```


30. Write a Python program to change the order of index of a given series.

In [30]:

```
import pandas as pd
s = pd.Series(data = [1,2,3,4,5], index = ['A', 'B', 'C','D','E'])
print("Original Data Series:")
print(s)
print()
s = s.reindex(index = ['B','A','C','D','E'])
print("Data Series after changing the order of index:")
print(s)
```

```
Original Data Series:
A    1
B    2
C    3
D    4
E    5
dtype: int64

Data Series after changing the order of index:
B    2
A    1
C    3
D    4
E    5
dtype: int64
```

31. Write a Python program to convert a given Series to an array.

In [31]:

```
import pandas as pd
import numpy as np
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print()
print("Series to an array")
a = np.array(s1.values.tolist())
print (a)
```

```
Original Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object

Series to an array
['100' '200' 'python' '300.12' '400']
```

32. Write a Python program to convert Series of lists to

one Series.

In [32]:

```
import pandas as pd
s = pd.Series([
    ['Red', 'Green', 'White'],
    ['Red', 'Black'],
    ['Yellow']])
print("Original Series of list: ")
print(s)
print()
s = s.apply(pd.Series).stack().reset_index(drop=True)
print("One Series: ")
print(s)
```

```
Original Series of list:
0    [Red, Green, White]
1            [Red, Black]
2                [Yellow]
dtype: object

One Series:
0       Red
1     Green
2     White
3       Red
4     Black
5    Yellow
dtype: object
```


33. Write a Python program to sort a given Series.

In [33]:

```
import pandas as pd
s = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s)
print()
new_s = pd.Series(s).sort_values()
print("Sorted series: ")
print(new_s)
```

```
Original Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object

Sorted series:
0       100
1       200
3    300.12
4       400
2    python
dtype: object
```


34. Write a Python program to sort the DataFrame first by 'name' in descending order, then by 'score' in ascending order using pandas.

Example:

```
    Input: exam_data={'name':['Anastasia','Dima','Katherine','James','Emily','Mich
ael','Matthew','Laura','Kevin','Jonas'],
            'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
            'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
            'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
 'yes']}
             labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
             Values for each column will be:
             name : "Suresh", score: 15.5, attempts: 1, qualify: "yes", label: "k"
```

In [34]:

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matt
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Orginal rows:")
print(df)
print()
df.sort_values(by=['name'], ascending=[False])
print("Sort the data frame first by ‘name’ in descending order, then by ‘score’ in ascendir
print(df)
print()
df.sort_values(by=['score'], ascending=[True])
```

```
Orginal rows:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Sort the data frame first by ‘name’ in descending order, then by ‘score’ in
ascending order:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

Out[34]:

	name	score	attempts	qualify
i	Kevin	8.0	2	no
b	Dima	9.0	3	no
e	Emily	9.0	2	no
a	Anastasia	12.5	1	yes
g	Matthew	14.5	1	yes
c	Katherine	16.5	2	yes

	name	score	attempts	qualify
j	Jonas	19.0	1	yes
f	Michael	20.0	3	yes
d	James	NaN	3	no
h	Laura	NaN	1	no

35. Write a Python program to set a given value for particular cell in DataFrame using index value.

Example:

```
    Input: exam_data={'name':['Anastasia','Dima','Katherine','James','Emily','Mich
ael','Matthew','Laura','Kevin','Jonas'],
          'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
          'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
          'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
 'yes']}
           labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
           Values for each column will be:
           name : "Suresh", score: 15.5, attempts: 1, qualify: "yes", label: "k"
```

In [35]:

```
import pandas as pd
import numpy as np
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matth
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame: ")
print(df)
print("\nSet a given value for particular cell in the DataFrame- ")
row=int(input("Enter Row no. to be changed: "))
col=input("Enter name of column to be changed: ")
val=float(input("Enter the value to replace: "))
df.at[row,col]=val
print(df)
```

```
Original DataFrame:
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James    NaN         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes

Set a given value for particular cell in the DataFrame-
Enter Row no. to be changed: 3
Enter name of column to be changed: score
Enter the value to replace: 999
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James  999.0         3      no
4      Emily    9.0         2      no
5    Michael   20.0         3     yes
6    Matthew   14.5         1     yes
7      Laura    NaN         1      no
8      Kevin    8.0         2      no
9      Jonas   19.0         1     yes
```


36. Write a Python program to find whether a year in the dataframe is a leap year or not.

In [36]:

```
import pandas as pd
import numpy as np
import datetime

dat=input("Enter date in YYYY-MM-DD HH:MM:SS format: ")
prd=int(input("Period of years: "))
date1 = pd.Series(pd.date_range(dat, periods=prd, freq='Y'))
df = pd.DataFrame(dict(date_given=date1))
print(df)

df['Is_leap_year'] = df['date_given'].dt.is_leap_year
print(df)
```

```
Enter date in YYYY-MM-DD HH:MM:SS format: 2019-09-11 10:00:00
Period of years: 10
           date_given
0 2019-12-31 10:00:00
1 2020-12-31 10:00:00
2 2021-12-31 10:00:00
3 2022-12-31 10:00:00
4 2023-12-31 10:00:00
5 2024-12-31 10:00:00
6 2025-12-31 10:00:00
7 2026-12-31 10:00:00
8 2027-12-31 10:00:00
9 2028-12-31 10:00:00
           date_given  Is_leap_year
0 2019-12-31 10:00:00         False
1 2020-12-31 10:00:00          True
2 2021-12-31 10:00:00         False
3 2022-12-31 10:00:00         False
4 2023-12-31 10:00:00         False
5 2024-12-31 10:00:00          True
6 2025-12-31 10:00:00         False
7 2026-12-31 10:00:00         False
8 2027-12-31 10:00:00         False
9 2028-12-31 10:00:00          True
```


37. Write code to calculate Log and natural Logarithmic value of a column in pandas python.

In [37]:

```
import pandas as pd
import numpy as np

#Create a DataFrame
df1 = {
     'Name':['George','Andrea','micheal','maggie','Ravi','Xien','Jalpa'],
   'University_Rank':[6,47,21,74,32,77,8]}

df1 = pd.DataFrame(df1,columns=['Name','University_Rank'])
print(df1)
print()
df1['log_value'] = np.log(df1['University_Rank'])
df1['log2_value'] = np.log2(df1['University_Rank'])
df1['log10_value'] = np.log10(df1['University_Rank'])
print(df1)
```

```
      Name  University_Rank
0   George                6
1   Andrea               47
2  micheal               21
3   maggie               74
4     Ravi               32
5     Xien               77
6    Jalpa                8

      Name  University_Rank  log_value  log2_value  log10_value
0   George                6   1.791759    2.584963     0.778151
1   Andrea               47   3.850148    5.554589     1.672098
2  micheal               21   3.044522    4.392317     1.322219
3   maggie               74   4.304065    6.209453     1.869232
4     Ravi               32   3.465736    5.000000     1.505150
5     Xien               77   4.343805    6.266787     1.886491
6    Jalpa                8   2.079442    3.000000     0.903090
```


38. Develop a code to find the difference between two Timestamps in Seconds, Minutes, hours and nanoseconds in Pandas Python.

In [38]:

```
import pandas as pd
import numpy as np
import datetime
from dateutil.relativedelta import relativedelta
from datetime import date


date1 = pd.Series(pd.date_range('2012-1-1 12:00:00', periods=7, freq='M'))
date2 = pd.Series(pd.date_range('2013-3-11 21:45:00', periods=7, freq='W'))

df = pd.DataFrame(dict(Start_date = date1, End_date = date2))
print(df)
print()
#Difference in seconds
df['diff_seconds'] = df['End_date'] - df['Start_date']
df['diff_seconds']=df['diff_seconds']/np.timedelta64(1,'s')
#Difference in minutes
df['diff_minutes'] = df['End_date'] - df['Start_date']
df['diff_minutes']=df['diff_minutes']/np.timedelta64(1,'m')
#Difference in Hours
df['diff_hours'] = df['End_date'] - df['Start_date']
df['diff_hours']=df['diff_hours']/np.timedelta64(1,'h')
#Difference in Nanoseconds
df['diff_nano_seconds'] = df['End_date'] - df['Start_date']
df['diff_nano_seconds']=df['diff_nano_seconds']/np.timedelta64(1,'ns')
print(df)
```

```
           Start_date            End_date
0 2012-01-31 12:00:00 2013-03-17 21:45:00
1 2012-02-29 12:00:00 2013-03-24 21:45:00
2 2012-03-31 12:00:00 2013-03-31 21:45:00
3 2012-04-30 12:00:00 2013-04-07 21:45:00
4 2012-05-31 12:00:00 2013-04-14 21:45:00
5 2012-06-30 12:00:00 2013-04-21 21:45:00
6 2012-07-31 12:00:00 2013-04-28 21:45:00

           Start_date            End_date  diff_seconds  diff_minutes  \
0 2012-01-31 12:00:00 2013-03-17 21:45:00    35545500.0      592425.0
1 2012-02-29 12:00:00 2013-03-24 21:45:00    33644700.0      560745.0
2 2012-03-31 12:00:00 2013-03-31 21:45:00    31571100.0      526185.0
3 2012-04-30 12:00:00 2013-04-07 21:45:00    29583900.0      493065.0
4 2012-05-31 12:00:00 2013-04-14 21:45:00    27510300.0      458505.0
5 2012-06-30 12:00:00 2013-04-21 21:45:00    25523100.0      425385.0
6 2012-07-31 12:00:00 2013-04-28 21:45:00    23449500.0      390825.0

   diff_hours  diff_nano_seconds
0     9873.75       3.554550e+16
1     9345.75       3.364470e+16
2     8769.75       3.157110e+16
3     8217.75       2.958390e+16
4     7641.75       2.751030e+16
5     7089.75       2.552310e+16
6     6513.75       2.344950e+16
```

39. Write a Python program to change one value to

another value in name column of the data frame.

In [39]:

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matt
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Orginal rows:")
print(df)
print()
nm1=input("Enter Name to be replaced: ")
nm2=input("Enter new name: ")
print("\nChange the name %s to %s: ",nm1,nm2)
df['name'] = df['name'].replace(nm1, nm2)
print(df)
```

```
Orginal rows:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Enter Name to be replaced: Dima
Enter new name: Kama

Change the name %s to %s:  Dima Kama
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Kama    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

40. Write a NumPy program to remove the leading and trailing whitespaces of all the elements of a given array.

In [40]:

```
import numpy as np
x = np.array([' python exercises ', ' PHP  ', ' java  ', '  C++'], dtype=np.str)
print("Original Array:")
print(x)
stripped = np.char.strip(x)
print("\nRemove the leading and trailing whitespaces: ", stripped)
```

```
Original Array:
[' python exercises ' ' PHP  ' ' java  ' '  C++']

Remove the leading and trailing whitespaces:  ['python exercises' 'PHP' 'jav
a' 'C++']
```