Module 3 - Python Fundamentals

and

Module 4 - Control Flow & Functions

Python Fundamentals include

- Syntax Rules
- Indentation
- Tokens
- Data Types

Control Flow is the order in which the code is executed

- In Control Flow, we make computer think and make decisions
- There are two ways of control flow:
  - Conditional Statements
  - Loops

Question 1 - Beginner

Task to be performed:

1. Using escape sequence, print each word in different lines.

Example:

```
Input  - 'Hey! Welcome to edureka!'
Output -  Hey!
          Welcome
          to
          edureka!
```

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

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

Question 2 - Beginner

Tasks to be performed:

1. Write a program to generate the following pattern:

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

2. Write a program to generate the following pattern.

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

3. Write a program to generate the following output.

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

In [ ]:
```
#Task 1:
for i in range(5):
    print('*'*(i+1))
```

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

In [ ]:
```
#Task 2:
for i in range(5):
    print(' '*(5-i+1)+'*'*(2*i+1))
```

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

In [ ]:
```
#Task 3:
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
```

Question 3 - Beginner

Task to be performed:

1. Write a Python 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 [ ]:
```
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: 256
Square:  65536
Cube:  16777216
Square Root:  16.0
```

Question 4 - Beginner

Task to be performed:

1. Write a program to find if the given number is an Armstrong number or not and print 'Yes' or 'No'.

(Hint: Armstrong Number is sum of its own digits raised to power of 3)

Example:

```
Input:153

Output:Yes
```

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

```
371
Yes
```

Question 5 - Beginner

Task to be performed:

1. Write a program to find the sum of digits from a given integer number N.

Example:

```
Input:4235

Output:14
```

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

```
3476
20
```

Question 6 - Beginner

Task to be performed:

1. Write a program to generate N numbers of fibbonacci series seperated by space.

Example:

```
Input: 5

Output: 0 1 1 2 3
```

In [ ]:
```
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
```

```
7
0 1 1 2 3 5 8
```

Question 7 - Beginner

Task to be performed:

1. Write a Python program to find the factorial of a given number N.

Example:

```
Input:5

Output:120
```

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

```
6
720
```

Question 8 - Beginner (Bridging Question)

Task to be performed:

Write a Python program to find the factors of a natural number given as input. Print all the factors separated by commas.

Example:

```
Input: Enter Number: 6

Output: 1,2,3,6
```

In [ ]:
```
def factors(x,i):
    s = 0
    if i >= x:
        return x
    if x % i == 0:
        s += i
        print(s,end=',')
    return factors(x,i+1)
n = int(input('Enter Number: '))
c = 1
print(factors(n,c))
```

```
Enter Number: 9
1,3,9
```

Question 9 - Intermediate

Task to be performed:

1. Write a Python program to find the difference between two dates.

Topics covered:

1. datetime function

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 [ ]:
```
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: 1998-08-08
Enter a date in YYYY-MM-DD format: 2018-08-08
7305 days
```

Question 10 - Beginner (Bridging Question)

Task to be performed:

1. Write a Python function to get either two or three numbers from the user and find the largest number.

In [1]:
```
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("How many numbers do you want to compare? "))
if n==2:
    x=int(input("Enter first number: "))
    y=int(input("Enter second number: "))
    print("Max number: ", 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 number: ",max_of_three(x,y,z))
else:
    print("Enter a smaller value of N!")
```

```
How many numbers do you want to compare? 3
Enter first number: 234
Enter second number: 563
Enter third number: 65
Max number:  563
```

Question 11 - Intermediate

Task to be performed:

1. Write a Python program to debit and credit money from a bank account using functions.

Example:

```
Input: Press D to deposit/W to withdraw: W
       Enter amount to be withdrawn: 200

Output: Deposit Balance: -200
```

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

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

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

a=make_account()

press = str(input("Press D to deposit/W to withdraw: "))

if(press == 'W' or press == 'w' ):

 amt=int(input("Enter amount to be withdrawn: "))
 withdraw(a,amt)

elif(press == 'D' or press == 'd'):

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

else:
 print('Press correct key')
```

```
Press D to deposit/W to withdraw: D
Enter amount to be deposited: 54790
Deposit Balance: 54790
```

Question 12 - Intermediate

Task to be performed:

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

Topics covered:

1. Opening and closing a file
2. Writing a file

In [ ]:
```
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()
```

```
13
```

Question 13 - Intermediate (Bridging Question)

Task to be performed:

1. Write a Python code to calculate the area of the object based on the parameters given by the user.

Topics covered:

1. User-defined functions

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 [ ]:
```
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? Triangle
Give the length of base: 3
Give the height: 5
Area:  7.5
```

Question 14 - Intermediate

Task to be performed:

1. Write a Python program to find whether a sequence of numbers is an additive sequence or not.

(Hint: 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))
```

```
Enter Sequence: 66121830
True
```

Question 15 - Intermediate

Task to be performed:

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

Example:

```
Input: 5

Output: 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))
```

```
Enter a number: 14
Roman numeral:  XIV
```

Question 16 - Beginner

Task to be performed:

1. Write a program that accepts a sequence of numbers and print the numbers after sorting them in ascending order seperated by comma.

Example:

```
Input:  5
        5 2 3 7 4

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

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

```
5
5
6
2
9
1
[1, 2, 5, 6, 9]
```

Question 17 - Intermediate (Bridging Question)

Greatest Common Divisor (GCD) of two or more integers is the largest positive integer that divides the numbers without a remainder.

Task to be performed:

1. Write a Python program to read two integers from the input and print their GCD.

Example:

```
Input: Enter first number: 8
       Enter second number: 12

Output: The GCD of 8 and 12 is 4
```

In [ ]:
```
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
if x > y:
    smaller = y
else:
    smaller = x
for i in range(1,smaller + 1):
    if((x % i == 0) and (y % i == 0)):
        hcf = i

print("The GCD of", x,"and", y,"is", hcf)
```

```
Enter first number: 5
Enter second number: 15
The GCD of 5 and 15 is 5
```

Question 18 - Intermediate

Task to be performed:

1. Write a Python class which has two methods *get_String* and *print_String*.

- get_String accepts a string from the user.
- print_String prints 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()
```

```
no big deal
NO BIG DEAL
```

Question 19 - Intermediate

Task to be performed:

1. Write a Python program to calculate the discriminant value of a quadratic equation and number of possible solutions of the equation. For a quadratic equation of the form xt^2 + yt + z = 0, the discriminant is given as (y^2 - 4 *x* z).

Example:

```
Input: The x value: 10
       The y value: 5
       The z value: 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()
```

```
The x value: 19
The y value: 2
The z value: 32
No Real Solutions. Discriminant value is: -2428.0
```

Question 20 - Intermediate (Bridging Question)

Task to be performed:

1. Create a function Sequence() which will read an integer X as input. Print an 'X' character-long alphanumeric string as per the following rules:

- The string should have a digit at every **odd** index character
- The string should have a lower case alphabet at every **even** index character

Example:

```
Input: 17

Output: b5n4c1l4o5p3y2w3t
```

In [ ]:
```
import random
def sequence(X):
    alphanumeric=''
    for i in range(X):
        if(i%2==0):
            alphanumeric = alphanumeric + chr(random.randrange(97,122))
        else:
            alphanumeric = alphanumeric + chr(random.randrange(48,58))
    print(alphanumeric)


s=int(input())
sequence(s)
```

```
32
x9c4w1w5j6q9d5j4d6n2h4j1b8c1t1u4
```

Question 21 - Intermediate

Task to be performed:

1. 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));
```

```
0.125
243
1
```

Question 22 - Intermediate (Bridging Question)

Task to be performed:

1. Create four functions: *listadd, listsubtract, listmax, and listsort*:

- *listadd* takes two lists as arguments and returns a list with elements from both the lists
- *listsubtract* takes two lists as arguments and returns a list with elements in the first list but not in the second list
- *listmax* takes a list as an argument and returns the maximum element from the list
- *listsort* takes a list as an argument and returns a list sorted in ascending order

Note: All the arguments to the functions will be the list of integers.

In [ ]:
```
list_1 = []

# number of elemetns as input
n = int(input("Enter number of elements for list 1 : "))

# iterating till the range
for i in range(0, n):
    ele = int(input())

    list_1.append(ele) # adding the element


list_2 = []

# number of elemetns as input
n = int(input("Enter number of elements for list 2: "))

# iterating till the range
for i in range(0, n):
    ele = int(input())

    list_2.append(ele) # adding the element


def listadd(list_a, list_b):
    return list_a + list_b

def listsubtract(list_a, list_b):
    return [i for i in list_a if i not in list_b]

def listmax(list_a):
    return max(list_a)

def listsort(list_a):
    return sorted(list_a)

print(listadd(list_1, list_2))
print(listsubtract(list_1, list_2))
print(listmax(list_1))
print(listsort(list_1))
```

```
Enter number of elements for list 1 : 4
12
23
34
45
Enter number of elements for list 2: 4
87
5
49
76
[12, 23, 34, 45, 87, 5, 49, 76]
[12, 23, 34, 45]
45
[12, 23, 34, 45]
```

Question 23 - Advanced

Create a game of "Rock, Paper, Scissors" with the computer using Python.

Rules for the game are given as follows:

- Rock beats Scissors
- Scissors beats Paper
- Paper beats Rock

Input moves as R for Rock, P for Paper and S for Scissors.

If a player loses the game, ask if he/she wants to play again. If the response is *'Yes'* then continue the game.

In [1]:
```
import random
P1=0
P2=0
win_lose={'Lose':['RP','PS','SR'],'Draw':['RR','PP','SS'],'Win':['RS','PR','S
P']}
repeat='Yes'
c=['R','S','P']
while repeat=='Yes':
    turn=input("Input your move: ")
    comp=random.choice(c)
    for i in win_lose:
        if str(turn)+comp in win_lose[i]:
            print(i)
            repeat=input('Play again? ')
```

```
Input your move: S
Lose
Play again? Yes
Input your move: S
Lose
Play again? Yes
Input your move: S
Win
Play again? No
```

Question 24 - Advanced

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

Task to be performed:

- Allow the user to input the 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 – de
finitely', 'You may rely on it', 'As I see it, yes', 'Most likely', 'Outlook g
ood', 'Yes Signs point to yes', 'Reply hazy', 'try again', 'Ask again later',
'Better not tell you now', 'Cannot predict now', 'Concentrate and ask again',
'Dont count on it', 'My reply is no', 'My sources say no', 'Outlook not so goo
d', 'Very doubtful']
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()
```

```
Hello World, I am the Magic 8 Ball, What is your name?
P
Hello P
Ask me a question.
Will I get to see a dog today?
Yes – definitely
I hope that helped!
Do you have another question? [Y/N]
N
Adios!
```

Question 25 - Advanced

Task to be performed:

1. Write a Python code to check the strength of a password.

(Hint: Check for the strength of 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 [ ]:
```
import re

def password():
    print ('Enter a password\n\nThe password must be between 6 and 12 characte
rs.\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'], F
alse)
    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: Qwerty7
Password is Strong
```

Question 26 - Advanced

Write a Python code to simulate the distribution of a deck of cards.

Task to be performed:

- The Deck class should have a deal method to deal with 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
```

```
New deck = 52 cards
----------------------------------------
hand1 = ['8S', '4S', '4H', '3C', 'TD']
hand2 = ['QH', 'QC', '7H', 'KC', 'AD']
----------------------------------------
New deck = 42 cards
```

Question 27 - Advanced

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")
```

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):4
Enter first number: 34
Enter second number: 17
34 / 17 = 2.0
```

Question 28 - Advanced

Task to be performed:

1. Write a Python 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)
```

```
New Delhi is the capital of India.
Hindi the primary language of India.
India is a developing country.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
USA is a developed country.
```

Question 29 - Advanced

Task to be performed:

1. Demonstrate Inheritance in Python using an example of employees.

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)
```

```
Python
10000
<bound method employee.fullname of <__main__.developer object at 0x7f9b5ada18
d0>>
```

Question 30 - Advanced

Task to be performed:

1. Develop a Python 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()
```

```
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
```

Question 31 - Advanced

Task to be performed:

1. Develop a Python code to generate sample data out of existing data and append it to the dataset.
2. Use *filter()* function to subset all columns in a dataframe that has the letter provided by the user in its name.
3. Develop a Python code to generate sample data and replace all the null values with dummy data.

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/xlko3qnazi445w6/employees.csv
```

```
--2020-06-23 12:03:58--  https://www.dropbox.com/s/xlko3qnazi445w6/employees.
csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/xlko3qnazi445w6/employees.csv [following]
--2020-06-23 12:03:59--  https://www.dropbox.com/s/raw/xlko3qnazi445w6/employ
ees.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc479bf93fad3dd619bac4a1e5c4.dl.dropboxusercontent.com/cd/
0/inline/A6N3kYPWzojl9eqfSIilXJWfTrGEoTnruKV4Z-q_opONjjRTIV-DzbDLt8ZWZF47M3UQ
TSUFHGoeypxukTo-PrPTaKwcrCUSPI21ldKJCJwQ7_dOuhbGdPPuIpbZE2npMRQ/file# [follow
ing]
--2020-06-23 12:03:59--  https://uc479bf93fad3dd619bac4a1e5c4.dl.dropboxuserc
ontent.com/cd/0/inline/A6N3kYPWzojl9eqfSIilXJWfTrGEoTnruKV4Z-q_opONjjRTIV-Dzb
DLt8ZWZF47M3UQTSUFHGoeypxukTo-PrPTaKwcrCUSPI21ldKJCJwQ7_dOuhbGdPPuIpbZE2npMR
Q/file
Resolving uc479bf93fad3dd619bac4a1e5c4.dl.dropboxusercontent.com (uc479bf93fa
d3dd619bac4a1e5c4.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc479bf93fad3dd619bac4a1e5c4.dl.dropboxusercontent.com (uc479bf
93fad3dd619bac4a1e5c4.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 59175 (58K) [text/plain]
Saving to: ‘employees.csv’

employees.csv       100%[===================>]  57.79K  --.-KB/s    in 0.01s

2020-06-23 12:03:59 (4.22 MB/s) - ‘employees.csv’ saved [59175/59175]
```

In [ ]:
```
#Task 1:
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
# generating one row
rows = data.sample(frac =.25)
# checking if sample is 0.25 times data or not
if (0.25*(len(data))== len(rows)):
    print( "Cool")
    print(len(data), len(rows))
# display
rows
```

```
Cool
1000 250
```

Out[ ]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
552	Barbara	Female	9/2/1991	3:41 PM	127297	11.905	True	Product
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
476	Kathy	Female	10/25/1996	12:59 PM	143541	8.461	False	Human Resources
133	Lois	Female	12/25/1987	4:16 PM	106317	2.235	True	Client Services
807	Mary	Female	11/6/2011	8:32 AM	115057	2.089	False	Finance
...	...	...	...	...	...	...	...	...
457	Patricia	Female	1/9/2015	4:16 AM	121232	16.624	False	Legal
298	Patrick	NaN	8/30/2004	11:43 AM	137314	4.542	True	Marketing
614	Eric	Male	11/12/2004	9:16 PM	65168	11.513	False	Distribution
395	Kathy	Female	11/25/2001	9:55 PM	93753	7.094	True	Sales
839	Joyce	Female	7/25/2001	6:04 AM	51065	16.807	False	Human Resources

250 rows × 8 columns

In [ ]:
```
#Task 2:
import pandas as pd

# Creating the dataframe
df = pd.read_csv("employees.csv")
# Using regular expression to extract all
# columns which has letter 'a' or 'A' in its name.
regpat=input("Enter letter to filter out: ")
df.filter(regex = regpat)
```

```
Enter letter to filter out: A
```

Out[ ]:

0
1
2
3
4
...
995
996
997
998
999

1000 rows × 0 columns

In [ ]:
```
#Task 3:
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
df = pd.DataFrame(data)
# generating one row
rows = data.sample(frac =.25)
# checking if sample is 0.25 times data or not
if (0.25*(len(data))== len(rows)):
    print( "Cool")
    print(len(data), len(rows))
# display
buff=input("Enter a buffer value: ")
res = df.apply(lambda x: x.fillna(00) if x.dtype.kind in 'biufcOSUV' else x.fi
llna(buff))
#data=data['First Name'].fillna(buff,inplace = True)
print("After:\n")
rows
```

```
Cool
1000 250
Enter a buffer value: 999
After:
```

Out[ ]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
121	Kathleen	NaN	5/9/2016	8:55 AM	119735	18.740	False	Product
968	Louise	Female	3/27/1995	10:27 PM	43050	11.671	False	Distribution
542	Amanda	Female	8/1/2004	1:32 PM	80803	14.077	True	Distribution
188	Charles	Male	10/14/2000	9:40 PM	71749	15.931	False	Legal
191	Lois	Female	10/18/2013	4:51 PM	36946	6.652	False	Engineering
...	...	...	...	...	...	...	...	...
773	NaN	Male	10/24/1986	9:23 AM	47176	10.736	NaN	Finance
813	Evelyn	Female	2/10/2002	4:44 AM	123621	19.767	True	Marketing
26	Craig	Male	2/27/2000	7:45 AM	37598	7.757	True	Marketing
456	Deborah	NaN	2/3/1983	11:38 PM	101457	6.662	False	Engineering
701	Robin	NaN	9/16/2005	1:26 AM	93201	11.712	True	Legal

250 rows × 8 columns

Module 5 - Array Computations

NumPy

- NumPy is the foundation library for scientific computation in Python
- It contains, among other things:
  - A powerful n-dimensional array object
  - Sophisticated functions
  - Tools for integrating with other languages

Question 1 - Beginner

Task to be performed:

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 [ ]:

```
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]]
```

Question 2 - Beginner

Task to be performed:

1. Write a Python program to get the floor, ceiling and truncated values of the elements of a 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 [ ]:

```
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.]
```

Question 3 - Beginner

Task to be performed:

1. 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 [ ]:

```
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. ]]
```

Question 4 - Beginner (Bridging Question)

Tasks to be performed:

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 [ ]:

```
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.]))
```

Question 5 - Beginner

Tasks to be performed:

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 [ ]:

```
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: 120
Average of the array elements:
         0.07344368488146831
Standard deviation of the array elements:
         1.1163175344101983
Variance of the array elements:
         1.2461648376316645
```

Question 6 - Beginner

Task to be performed:

1. 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 [ ]:

```
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]
```

Question 7 - Beginner

Task to be performed:

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

Example:

```
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 [ ]:

```
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:
[0.54032553 0.47071709 0.78826623 0.18970793 0.06371784 0.49543187
 0.06259775 0.79558221 0.07498281 0.84177656]
Enter the number: 6

Sorted first N elements:
[0.06259775 0.06371784 0.07498281 0.18970793 0.47071709 0.49543187
 0.54032553 0.79558221 0.78826623 0.84177656]
```

Question 8 - Beginner

Task to be performed:

1. 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 -0.3160
8052
          1.36096266]
```

In [ ]:

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

```
Enter number: 7
[-1.43936728 -0.49836084  1.82676208 -0.16638007 -0.95862398 -0.73128906
 -0.04674693]
```

Question 9 - Beginner

Task to be performed:

1. Write a Python program to create a 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 [3]:

```
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 %d:" % (n))
print(x)
```

```
Enter value of N: 6
Original array:
[0.88003123 0.48125361 0.86569404 0.05787013 0.39058659 0.47565201]
Maximum value replaced by 6:
[6.         0.48125361 0.86569404 0.05787013 0.39058659 0.47565201]
```

Question 10 - Beginner

Task to be performed:

1. 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 [ ]:

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

Question 11 - Intermediate

Task to be performed:

1. Write a Python program that takes the number of rows and columns and finds the Cartesian and Polar products of a random MxN matrix representing Cartesian coordinates, i.e., (M,N).

Topics covered:

1. Array Manipulation

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 [ ]:

```
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:
 [0.86963009 0.45201153 0.84177515 1.13521499 1.10532844 0.4402121
 1.15569367 1.21808373 0.95491788 1.03128296]
Polar Product:
 [1.06025157 1.06448901 0.94646131 0.9933297  0.89499927 0.08510322
 0.9358328  0.69948309 0.31981508 1.24777052]
```

Question 12 - Intermediate

Task to be performed:

1. Take two numerical inputs X & Y, where (X<Y), and write a Python program to *shuffle* numbers between X and Y.
2. Also, find the result using *permutation*.

Topics covered:

1. Shuffle function
2. Permutation function

Example:

```
Input: Lower input value: 5
       Higher input value: 20

Output: [10.  15.  18.  17.5 13.   5.   9.  19.5 12.5 16.  10.5  7.5  5.5 1
7.
         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 [4]:

```
import numpy as np
x=int(input("Lower input value: "))
y=int(input("Higher input value: "))
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 input value: 7
Higher input value: 23
[19.5 14.  18.5  9.5  8.5 22.5 14.5 12.  17.5  8.  13.  15.5 18.  10.5
 17.  10.   7.  16.5 20.  19.   9.   7.5 21.  11.  12.5 21.5 11.5 22.
 15.  13.5 16.  20.5]
Same result using permutation():
[ 1  9 15  6  2  0  8  7 14 12 10  5 11 13  4  3]
```

Question 13 - Beginner

Task to be performed:

1. 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 [ ]:

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

```
[[[0.86801635 0.86542653 0.55187853]
  [0.66333702 0.29792621 0.08840663]
  [0.11599961 0.87908636 0.48385772]]

 [[0.3970789  0.83266533 0.26077771]
  [0.89997594 0.59389266 0.965916  ]
  [0.07682436 0.87923207 0.40864553]]

 [[0.1212504  0.97781444 0.83834454]
  [0.97746378 0.2201537  0.06047495]
  [0.13710041 0.14282104 0.52692216]]]
```

Question 14 - Intermediate (Bridging Question)

Write a Python program which takes two integer-NumPy arrays, P and Q of shape [3 * 3] and perform the following task:

Task to be performed:

Print the element-wise difference of the matrix P and Q (P-Q).

Topics covered:

1. Append function
2. Matrix subtraction

Example:

```
Input: 23 56 87 3 6 96 4 6 78
       12 34 54 7 2 54 6 2 78

Output: [[11 22 33]
         [-4  4 42]
         [-2  4  0]]
```

In [ ]:

```
import numpy as np

arr1 = list()
arr2 = list()
arr1.append(input().split(' '))
arr2.append(input().split(' '))

array_1 = np.array(arr1, int).reshape(3,3)
array_2 = np.array(arr2, int).reshape(3,3)

print(array_1 - array_2)
```

```
23 56 87 3 6 96 4 6 78
12 34 54 7 2 54 6 2 78
[[11 22 33]
 [-4  4 42]
 [-2  4  0]]
```

Question 15 - Intermediate

Task to be performed:

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

Topics covered:

1. atleast_2d() function

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 [ ]:

```
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.         0.16549908 0.37713704 0.10726593 0.68102777]
 [0.16549908 0.         0.50008381 0.25345639 0.7855579 ]
 [0.37713704 0.50008381 0.         0.27053905 0.30875658]
 [0.10726593 0.25345639 0.27053905 0.         0.57635195]
 [0.68102777 0.7855579  0.30875658 0.57635195 0.        ]]
```

Question 16 - Intermediate

Task to be performed:

1. 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.

Topics covered:

1. isdigit() function
2. islower() function
3. isupper() function

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 [ ]:

```
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]
```

Question 17 - Intermediate

Task to be performed:

1. 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
```

In [ ]:

```
import numpy as np

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

```
21.25271078649
```

Question 18 - Intermediate (Bridging Question)

Write a program which reads two space-separated positive integers X and Y as input and perform the following tasks:

Tasks to be performed:

1. Create a list (lst1) starting at one (1) with 16 elements at a step of X
2. Create a list (lst2) starting at one (1) with 16 elements at a step of Y
3. Create two NumPy arrays np1 and np2 using lst1 and lst2 respectively
4. Reshape both the NumPy arrays to (4,4)
5. Create a new np array (np3) with values obtained by subtracting both the arrays (np1 - np2)
6. Print all the elements of np3 in a single dimension list like format as shown below:

```
[n0 n1 n2 n3 n4 n5 n6 n7 n8]
```

Topics covered:

1. List and Array creation
2. Array subtraction
3. Changing array shape

Example:

```
Input: 7 9

Output: [  0  -2  -4  -6  -8 -10 -12 -14 -16 -18 -20 -22 -24 -26 -28 -30]
```

In [ ]:

```
import numpy as np
X,Y=map(int,input().split(' '))

lst1=list(range(1,X*16,X))
lst2=list(range(1,Y*16,Y))
np1=np.array(lst1)
np2=np.array(lst2)
np1=np1.reshape((4,4))
np2=np2.reshape((4,4))
np3=np.ravel(np1-np2)
print(np3)
```

```
7 9
[  0  -2  -4  -6  -8 -10 -12 -14 -16 -18 -20 -22 -24 -26 -28 -30]
```

Question 19 - Intermediate

Task to be performed:

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

Topics covered:

1. Matrix ranking

Example:

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

Output: 3
```

In [ ]:

```
import numpy

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

Out[ ]:

```
3
```

Question 20 - Advanced

Task to be performed:

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

(Hint: Use inbuilt attribute einsum)

Topics covered:

1. Matrix summation

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 [ ]:

```
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]]
```

Question 21 - Advanced (Bridging Question)

Write a Python program to read an integer on the first line of input for the number of elements (N) for a 1D NumPy array and read N comma-separated integers on the second line of input. Now perform the tasks given below:

Tasks to be performed:

1. Create a 1 X N NumPy array
2. Apply below function to each element in the NumPy array:

   f(xi) = (xi + u) / s^2

   where,

   - f(xi) represents the scaled/transformed value for the ith element in the array
   - xi represents an ith element in the given array
   - u represents the average of the given array
   - s represents the variance of the given array
3. Print the scaled/transformed array on a newline with precision up to 2 decimal places

Topics covered:

1. Array transformation
2. Array splitting

Example:

```
Input: 5
       2,3,5,7,9

Output: [-0.49 -0.34 -0.03  0.27  0.58]
```

In [ ]:

```
x=int(input())
l=map(int,input().split(','))
np1=np.array(list(l))
u=np1.mean()
s=np.std(np1)
v=s*s
np2=np1-u
np2=np2/v
print(np2.round(2),sep='\n')
```

```
5
2,3,5,7,9
[-0.49 -0.34 -0.03  0.27  0.58]
```

Question 22 - Advanced

Task to be performed:

1. Write a Python program to 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 [ ]:

```
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
```

Question 23 - Advanced

Task to be performed:

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

Topics covered:

1. Matrix multiplication

Example:

```
Input: 5  3  2

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 [ ]:

```
 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: 2
Enter value A: 3
First array:
[[0.4896915  0.82468463]
 [0.29311956 0.66149096]
 [0.90897446 0.0647436 ]
 [0.56893707 0.35792201]
 [0.30668585 0.69117229]]
Second array:
[[0.15528125 0.49299508 0.50408899]
 [0.76300888 0.99267579 0.2156281 ]]
Dot product of two arrays:
[[0.7052816  1.06005997 0.42467328]
 [0.55023945 0.80115256 0.29039438]
 [0.19054663 0.51238934 0.47216456]
 [0.36144293 0.63578369 0.36397296]
 [0.57499316 0.83730462 0.30363313]]
```

Question 24 - Advanced (Bridging Question)

You have been provided with two arrays:

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

Tasks to be performed:

Write a program to perform array shape manipulation.

1. Create a contiguous flattened array
2. Transpose the array
3. Flatten the transposed array
4. Perform re-shaping on the array (inverse operation to flattening)

Topics covered:

1. Transpose of an array
2. Array flattening
3. Array re-shaping

In [ ]:

```
#Task 1:
a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
a.ravel()
```

Out[ ]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [ ]:

```
#Task 2:
a.T
```

Out[ ]:

```
array([[ 1,  6],
       [ 2,  7],
       [ 3,  8],
       [ 4,  9],
       [ 5, 10]])
```

In [ ]:

```
#Task 3:
a.T.ravel()
```

Out[ ]:

```
array([ 1,  6,  2,  7,  3,  8,  4,  9,  5, 10])
```

In [ ]:

```
#Task 4:
a.shape
```

Out[ ]:

```
(2, 5)
```

In [ ]:

```
b = a.ravel()
b = b.reshape((2, 5))
b
```

Out[ ]:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

Module 6 - Data Manipulation

- Data Manipulation is the process of performing various operations on data to make it ready for further analysis
- Data Manipulation involves
  - Data Preparation: Load, combine, filter data
  - Data Cleaning: Remove inconsistencies from data
  - Data Transformation: Group, aggregate, bin data

pandas

- pandas is an open-source library which provides high-performance data structures to perform efficient data manipulation and analysis in Python

Question 1 - Beginner

Task to be performed:

1. 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
```

Output:

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

```
20 30
0     20
1     21
2     22
3     23
4     24
5     25
6     26
7     27
8     28
9     29
10    30
dtype: int64
```

Question 2 - Beginner

Task to be performed:

1. Write a Python program to add, subtract, multiply 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 [ ]:
```
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)
```

```
7, 5, 4, 9, 8
2, 8, 13, 6, 4
Addition:  0     9
1    13
2    17
3    15
4    12
dtype: int64
Substraction:  0    5
1   -3
2   -9
3    3
4    4
dtype: int64
Multiplication:  0    14
1    40
2    52
3    54
4    32
dtype: int64
Division:  0    3.500000
1    0.625000
2    0.307692
3    1.500000
4    2.000000
dtype: float64
```

Question 3 - Beginner

Task to be performed:

1. From the raw data below, create a Pandas Series.

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

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

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

```
Aron|Jackson|Ahree|Sam|

Aron|Jackson   |Ahree   |Sam|

  Aron|Jackson|   Ahree|Sam|
```

Question 4 - Beginner

Task to be performed:

1. Write a program to convert a dictionary into a 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 [ ]:
```
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
```

Question 5 - Beginner

Task to be performed:

1. Convert the given nested list into single series and print the output in sorted form.

Input:

```
[[1,23,12,31,14,12],[32,43,32,42],[65,75,65,57,41,33,68,52]]
```

Output:

```
0      1.0
2     12.0
5     12.0
4     14.0
1     23.0
3     31.0
8     32.0
6     32.0
15    33.0
14    41.0
9     42.0
7     43.0
17    52.0
13    57.0
10    65.0
12    65.0
16    68.0
11    75.0
dtype: float64
```

In [ ]:
```
import pandas as pd
series=pd.Series([[1,23,12,31,14,12],[32,43,32,42],[65,75,65,57,41,33,68,52]])
series=series.apply(pd.Series).stack()
series=series.reset_index(drop=True)
print(series.sort_values(axis=0))
```

```
0      1.0
2     12.0
5     12.0
4     14.0
1     23.0
3     31.0
8     32.0
6     32.0
15    33.0
14    41.0
9     42.0
7     43.0
17    52.0
13    57.0
10    65.0
12    65.0
16    68.0
11    75.0
dtype: float64
```

Question 6 - Beginner

Task to be performed:

1. Write a Python program to change the order of index of the Series.

   - The series is given as = [1,2,3,4,5]
   - The index is given as = ['A', 'B', 'C','D','E']
   - Change the index to = ['B','A','C','D','E']

In [ ]:
```
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
```

Question 7 - Beginner

Tasks to be performed:

1. Write a Python program to create the mean and standard deviation of the data from the given series.

   [134,257,323,464,523,668,795,810,969,531,300]

In [ ]:
```
import pandas as pd
s = pd.Series(data = [134,257,323,464,523,668,795,810,969,531,300])
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     134
1     257
2     323
3     464
4     523
5     668
6     795
7     810
8     969
9     531
10    300
dtype: int64

Mean of the said Data Series:
524.9090909090909

Standard deviation of the said Data Series:
262.98382252353645
```

Question 8 - Beginner

You have been given a series:

```
['100', '200', 'python', '300.12', '400']
```

Tasks to be performed:

1. Write a Python program to change the data type of the given series to numeric.
2. Write a Python program to convert the given series to an array.
3. Write a Python program to sort the given series.

In [ ]:
```
#Task 1:
import pandas as pd
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print()
print("Change the 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 data type to numeric:
0    100.00
1    200.00
2       NaN
3    300.12
4    400.00
dtype: float64
```

In [ ]:
```
#Task 2:
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']
```

In [ ]:
```
#Task 3:
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
```

Question 9 - Beginner

Task to be performed:

1. Create a series from 1 to 1000 and print only numbers divisble by 7 and 17 from the series.

Output:

```
118    119
237    238
356    357
475    476
594    595
713    714
832    833
951    952
dtype: int32
```

In [ ]:
```
import pandas as pd
series=pd.Series(np.arange(1,1001))
print(series[(series % 7==0) & (series%17==0)])
```

```
118    119
237    238
356    357
475    476
594    595
713    714
832    833
951    952
dtype: int64
```

Question 10 - Beginner

Task to be performed:

1. Create a Python program that reads the below dictionary as a DataFrame and then print it by iterating over columns and rows.

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

In [ ]:
```
import pandas as pd
my_dict={'Name':['Sameer','Leona','Samuel','Jackson','Gray','Sylphia'],
         'Class':[11,11,12,12,11,12], 'Age':[17,17,18,17,21,23] }
df=pd.DataFrame(my_dict)
for col in df.iteritems():
    print(col)
for row in df.iterrows():
    print(row)
```

```
('Name', 0     Sameer
1      Leona
2     Samuel
3    Jackson
4       Gray
5    Sylphia
Name: Name, dtype: object)
('Class', 0    11
1    11
2    12
3    12
4    11
5    12
Name: Class, dtype: int64)
('Age', 0    17
1    17
2    18
3    17
4    21
5    23
Name: Age, dtype: int64)
(0, Name     Sameer
Class        11
Age          17
Name: 0, dtype: object)
(1, Name     Leona
Class       11
Age         17
Name: 1, dtype: object)
(2, Name     Samuel
Class        12
Age          18
Name: 2, dtype: object)
(3, Name     Jackson
Class         12
Age           17
Name: 3, dtype: object)
(4, Name     Gray
Class      11
Age        21
Name: 4, dtype: object)
(5, Name     Sylphia
Class         12
Age           23
Name: 5, dtype: object)
```

Question 11 - Intermediate

You have been provided with the following data:

```
      employee_data = '{"employee_details":[{"employee_name": "James", "email": "ja
mes@gmail.com", "job_profile": "Sr. Developer"},{"employee_name": "Smith", "email":
"Smith@gmail.com", "job_profile": "Project Lead"}]}'
```

Write a Python program to import data from JSON file and convert this data into CSV format.

Tasks to be performed:

1. Import the data as a JSON file.
2. Open the file for writing.
3. Create the CSV writer object and convert the file into CSV format.
4. Print 'Done' after closing the file.

In [ ]:
```
import json
import csv

#Task 1:
employee_data = '{"employee_details":[{"employee_name": "James", "email": "jam
es@gmail.com", "job_profile": "Sr. Developer"},{"employee_name": "Smith", "ema
il": "Smith@gmail.com", "job_profile": "Project Lead"}]}'
employee_parsed = json.loads(employee_data)
emp_data = employee_parsed['employee_details']

#Task 2:
employ_data = open('EmployeeData.csv', 'w')

#Task 3:
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())

#Task 4:
employ_data.close()
print("DONE")
```

```
DONE
```

Question 12 - Intermediate

Task to be performed:

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

Example:

```
    Input: exam_data={'name':['Anastasia','Dima','Katherine','James','Emily','Micha
el','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', 'y
es']}
            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 [ ]:
```
import pandas as pd
import numpy as np
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Mic
hael', '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']}
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: 13
        name  score  attempts qualify
0  Anastasia   12.5         1     yes
1       Dima    9.0         3      no
2  Katherine   16.5         2     yes
3      James   13.0         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
```

Questions 13-16:

Samantha has created a dataset named 'top50spotify.csv' of her top 50 songs from spotify.

Dataset Description:

top50spotify.csv - The dataset contains 14 features. Here's a brief description of a few columns in the dataset:

- **SerialNo. - Serial number of songs**
- **Track.Name - Name of the track**
- **Artist.Name - Name of the artist**
- **Genre - Genre of the song**
- **Energy - Energy index of the song**
- **Length. - Length of the song**
- **Popularity - Popularity index of the song**

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/2hg67jin2n852mz/top50spotify.csv
```

```
--2020-06-30 06:10:09--  https://www.dropbox.com/s/2hg67jin2n852mz/top50spoti
fy.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/2hg67jin2n852mz/top50spotify.csv [following]
--2020-06-30 06:10:10--  https://www.dropbox.com/s/raw/2hg67jin2n852mz/top50s
potify.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucd4bf13c19b98c9aac39ef181ae.dl.dropboxusercontent.com/cd/
0/inline/A6nDNzKgzyBo5A8sbsIApXvgI7iGTEaEtvyggJiNasFrmr5-DNu73o0prQCIHgnQFTPr
Brv8lYcc_nUYzOUNW99uGJP4Lo09fJQpUY16gnrzGmFuag55u-_oQsFVswR4RJ4/file# [follow
ing]
--2020-06-30 06:10:10--  https://ucd4bf13c19b98c9aac39ef181ae.dl.dropboxuserc
ontent.com/cd/0/inline/A6nDNzKgzyBo5A8sbsIApXvgI7iGTEaEtvyggJiNasFrmr5-DNu73o
0prQCIHgnQFTPrBrv8lYcc_nUYzOUNW99uGJP4Lo09fJQpUY16gnrzGmFuag55u-_oQsFVswR4RJ
4/file
Resolving ucd4bf13c19b98c9aac39ef181ae.dl.dropboxusercontent.com (ucd4bf13c19
b98c9aac39ef181ae.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to ucd4bf13c19b98c9aac39ef181ae.dl.dropboxusercontent.com (ucd4bf1
3c19b98c9aac39ef181ae.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 3882 (3.8K) [text/plain]
Saving to: ‘top50spotify.csv’

top50spotify.csv    100%[===================>]   3.79K  --.-KB/s    in 0s

2020-06-30 06:10:11 (558 MB/s) - ‘top50spotify.csv’ saved [3882/3882]
```

Question 13 - Intermediate

Tasks to be performed:

1. **Import the dataset as a DataFrame and drop the first column.**
2. **Save it as 'top50.csv'.**

In [ ]:
```
#Task 1:
import pandas as pd
top50=pd.read_csv('top50spotify.csv')
col=list(top50.columns)
col[0]='temp'
top50.columns=col
top50=top50.drop('temp',axis=1)


#Task 2:
top50.to_csv('top50.csv')
top50.head()
```

Out[ ]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..	L
0	Señorita	Shawn Mendes	canadian pop	117	55	76	-6	
1	China	Anuel AA	reggaeton flow	105	81	79	-4	
2	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	40	-4	
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8	
4	Goodbyes (Feat. Young Thug)	Post Malone	dfw rap	150	65	58	-4	

Question 14 - Intermediate

Tasks to be performed:

1. **Find the average *Energy* and *Length* of first 10 songs.**
2. **Find the total length of songs, group by genre from top to bottom.**

In [ ]:
```
#Task 1:
first_10=top50[:10]
first_10[['Energy','Length.']].sum()/10
```

Out[ ]:
```
Energy      65.1
Length.    195.6
dtype: float64
```

In [ ]:
```
#Task 2:
print(top50.groupby('Genre')['Length.'].sum().sort_values(axis=0,ascending=False))
```

```
Genre
dance pop           1621
pop                 1368
latin               1126
edm                  656
reggaeton flow       611
canadian hip hop     579
panamanian pop       514
reggaeton            427
brostep              396
electropop           389
canadian pop         382
dfw rap              333
country rap          272
australian pop       210
atl hip hop          200
boy band             181
escape room          173
big room             164
r&b en espanol       162
pop house            153
trap music           131
Name: Length., dtype: int64
```

Question 15 - Intermediate (Bridging Question)

Tasks to be performed:

1. **Print the artist name with the most number of tracks in one genre.**

(Hint: Group by artist name and genre)

1. **Print the data of the tracks created by the artist from the previous question.**

In [ ]:
```
#Task 1:
import pandas as pd
keys=['Genre','Artist','N_Tracks']
new_df=pd.DataFrame(columns=keys)
i=0
for x,y in top50.groupby(['Genre','Artist.Name']):
    new_df.loc[i]=[x[0],x[1],y['Track.Name'].count()]
    i=i+1

new_df[new_df.N_Tracks==new_df.N_Tracks.max()]
```

Out[ ]:

	Genre	Artist	N_Tracks
27	pop	Ed Sheeran	4

In [ ]:
```
#Task 2:
top50[top50['Artist.Name']=='Ed Sheeran']
```

Out[ ]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..	Live
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8	
5	I Don't Care (with Justin Bieber)	Ed Sheeran	pop	102	68	80	-5	
37	Antisocial (with Travis Scott)	Ed Sheeran	pop	152	82	72	-5	
49	Cross Me (feat. Chance the Rapper & PnB Rock)	Ed Sheeran	pop	95	79	75	-6	

Question 16 - Intermediate

Task to be performed:

1. **Create a new column called Rating and input data based on the following:**
   - **If popularity is greater than average set 'Good'**
   - **If popularity is less than average set 'Bad'**

In [ ]:
```
avg=top50.Popularity.mean()
rating=[]
for i in top50.Popularity:
    if i <= avg:
        rating.append('Bad')
    else:
        rating.append('Good')

top50['Rating']=rating

top50.head()
```

Out[ ]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..	L
0	Señorita	Shawn Mendes	canadian pop	117	55	76	-6	
1	China	Anuel AA	reggaeton flow	105	81	79	-4	
2	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	40	-4	
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8	
4	Goodbyes (Feat. Young Thug)	Post Malone	dfw rap	150	65	58	-4	

Question 17 - Intermediate

Tasks to be performed:

1. **Create a pandas series from below dictionary where indices are subjects:**

```
{'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}
 }
```

2. **Convert the created series into DataFrame and replace the null values with zeroes.**
3. **Transpose the DataFrame and create a new column 'Average' and fill the values in it by calculating the average of all subjects.**

In [1]:
```
#Task 1:
import pandas as pd
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} }
series=pd.Series(tests)
series=series.apply(pd.Series)
series
```

Out[1]:

	Sam	Jackson	Ahree	Gloria	Isla	Aron	Gray
English	60.0	74.0	85.0	NaN	NaN	NaN	NaN
History	65.0	NaN	NaN	83.0	78.0	72.0	61.0
Geography	NaN	92.0	76.0	95.0	82.0	75.0	NaN
Mathematics	99.0	89.0	85.0	74.0	NaN	NaN	95.0
Science	89.0	NaN	87.0	NaN	93.0	82.0	78.0

In [ ]:
```
#Task 2:
df=pd.DataFrame(series)
df=df.fillna(0)
df
```

Out[ ]:

	Sam	Jackson	Ahree	Gloria	Isla	Aron	Gray
English	60.0	74.0	85.0	0.0	0.0	0.0	0.0
History	65.0	0.0	0.0	83.0	78.0	72.0	61.0
Geography	0.0	92.0	76.0	95.0	82.0	75.0	0.0
Mathematics	99.0	89.0	85.0	74.0	0.0	0.0	95.0
Science	89.0	0.0	87.0	0.0	93.0	82.0	78.0

In [ ]:
```
#Task 3:
df=df.transpose()
df
```

Out[ ]:

	English	History	Geography	Mathematics	Science
Sam	60.0	65.0	0.0	99.0	89.0
Jackson	74.0	0.0	92.0	89.0	0.0
Ahree	85.0	0.0	76.0	85.0	87.0
Gloria	0.0	83.0	95.0	74.0	0.0
Isla	0.0	78.0	82.0	0.0	93.0
Aron	0.0	72.0	75.0	0.0	82.0
Gray	0.0	61.0	0.0	95.0	78.0

In [ ]:
```
average=[]
for i in df.iterrows():
    average.append(i[1].mean())
df['Average']=average
df
```

Out[ ]:

	English	History	Geography	Mathematics	Science	Average
Sam	60.0	65.0	0.0	99.0	89.0	62.6
Jackson	74.0	0.0	92.0	89.0	0.0	51.0
Ahree	85.0	0.0	76.0	85.0	87.0	66.6
Gloria	0.0	83.0	95.0	74.0	0.0	50.4
Isla	0.0	78.0	82.0	0.0	93.0	50.6
Aron	0.0	72.0	75.0	0.0	82.0	45.8
Gray	0.0	61.0	0.0	95.0	78.0	46.8

Questions 18-22:

Gloria is planning to purchase a new car for herself. She inquired at various sources and was left with 2 different datasets managed by different sources for her research.

- **Insurance_Car_data.csv**
- **Sales_Car_data.csv**

Datasets Description:

Insurance_Car_data.csv - The dataset contains 6 features. Here's a brief description of the columns in the dataset:

- **Manufacturer - Name of the manufacturer**
- **Model - Name of the model**
- **Fuel capacity - Fuel capacity of the car**
- **Fuel efficiency - Fuel efficiency of the car**
- **Price in thousands - Price of the car**
- **Wheelbase - Wheelbase of the car**

Sales_Car_data.csv - The dataset contains 6 features. Here's a brief description of the columns in the dataset:

- **Manufacturer - Name of the manufacturer**
- **Model - Name of the model**
- **Sales in thousands - Sales of the cars**
- **4-year resale value - 4-year resale value of the car**
- **Latest Launch - Latest launch of the car**
- **Price in thousands - Price of the car**

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/8hn7xwu188lbohv/Insurance_Car_data.csv
```

```
--2020-06-30 06:10:56--  https://www.dropbox.com/s/8hn7xwu188lbohv/Insurance_
Car_data.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/8hn7xwu188lbohv/Insurance_Car_data.csv [following]
--2020-06-30 06:10:56--  https://www.dropbox.com/s/raw/8hn7xwu188lbohv/Insura
nce_Car_data.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc8eb9d6120473a7d483e7d1d0a0.dl.dropboxusercontent.com/cd/
0/inline/A6lGbjWjxpseHV_fVlYf-r8lw6EWCFPS4u0hb48UuL-LdKgXAKG8CK3OTedtwWoDfda-
0Tg4nPpxjHNo1t2ORBPPeQz-YT8rAw3DMtOuRJwvSxlldPLsfQFjTuUYWbdz4Vs/file# [follow
ing]
--2020-06-30 06:10:57--  https://uc8eb9d6120473a7d483e7d1d0a0.dl.dropboxuserc
ontent.com/cd/0/inline/A6lGbjWjxpseHV_fVlYf-r8lw6EWCFPS4u0hb48UuL-LdKgXAKG8CK
3OTedtwWoDfda-0Tg4nPpxjHNo1t2ORBPPeQz-YT8rAw3DMtOuRJwvSxlldPLsfQFjTuUYWbdz4V
s/file
Resolving uc8eb9d6120473a7d483e7d1d0a0.dl.dropboxusercontent.com (uc8eb9d6120
473a7d483e7d1d0a0.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc8eb9d6120473a7d483e7d1d0a0.dl.dropboxusercontent.com (uc8eb9d
6120473a7d483e7d1d0a0.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 8180 (8.0K) [text/plain]
Saving to: ‘Insurance_Car_data.csv’

Insurance_Car_data. 100%[===================>]   7.99K  --.-KB/s    in 0s

2020-06-30 06:10:57 (998 MB/s) - ‘Insurance_Car_data.csv’ saved [8180/8180]
```

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/9v3bwvl7k0q046o/Sales_Car_data.csv
```

```
--2020-06-30 06:10:59--  https://www.dropbox.com/s/9v3bwvl7k0q046o/Sales_Car_
data.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/9v3bwvl7k0q046o/Sales_Car_data.csv [following]
--2020-06-30 06:10:59--  https://www.dropbox.com/s/raw/9v3bwvl7k0q046o/Sales_
Car_data.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc76920c0f4391802168677d8dfa.dl.dropboxusercontent.com/cd/
0/inline/A6m700jG-dhJw66A0CXjDzbxCeLyJc3c4KBzmXrbdU1GzZ2MkzbDOm6aTTRS-FtxxdeB
EmOTQjK9LkO-WzU94HQdUpDvpED-mjv6euxydts3iGXZZKW0C5bktaJ_9YGca1Y/file# [follow
ing]
--2020-06-30 06:11:00--  https://uc76920c0f4391802168677d8dfa.dl.dropboxuserc
ontent.com/cd/0/inline/A6m700jG-dhJw66A0CXjDzbxCeLyJc3c4KBzmXrbdU1GzZ2MkzbDOm
6aTTRS-FtxxdeBEmOTQjK9LkO-WzU94HQdUpDvpED-mjv6euxydts3iGXZZKW0C5bktaJ_9YGca1
Y/file
Resolving uc76920c0f4391802168677d8dfa.dl.dropboxusercontent.com (uc76920c0f4
391802168677d8dfa.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc76920c0f4391802168677d8dfa.dl.dropboxusercontent.com (uc76920
c0f4391802168677d8dfa.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 9652 (9.4K) [text/plain]
Saving to: ‘Sales_Car_data.csv’

Sales_Car_data.csv  100%[===================>]   9.43K  --.-KB/s    in 0s

2020-06-30 06:11:00 (215 MB/s) - ‘Sales_Car_data.csv’ saved [9652/9652]
```

Question 18 - Intermediate

Task to be performed:

1. **Gloria requires both Insurance and Sales data before making the purchase, so merge both the datasets.**

In [ ]:
```
import pandas as pd
i_car=pd.read_csv('Insurance_Car_data.csv')
s_car=pd.read_csv('Sales_Car_data.csv')
car=pd.merge(s_car,i_car,how='outer')
car.head()
```

Out[ ]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelbase
0	Acura	Integra	16.919	16.360	02-Feb-14	21.50	13.2	28.0	101.2
1	Acura	TL	39.384	19.875	06-Mar-15	28.40	17.2	25.0	108.1
2	Acura	CL	14.114	18.225	01-Apr-14	NaN	17.2	26.0	106.9
3	Acura	RL	8.588	29.725	03-Oct-15	42.00	18.0	22.0	114.6
4	Audi	A4	20.397	22.255	10-Aug-15	23.99	16.4	27.0	102.6

Question 19 - Intermediate (Bridging Questions)

Task to be performed:

1. **Remove any null values with mode if data is categorical else with average.**

In [ ]:
```
car.isna().any()
```

Out[ ]:
```
Manufacturer           False
Model                  False
Sales in thousands     False
4-year resale value     True
Latest Launch          False
Price in thousands      True
Fuel capacity           True
Fuel efficiency         True
Wheelbase               True
dtype: bool
```

In [ ]:
```
import numpy as np
for i in car:
    if car[i].dtype==np.float64:

        car[i]=car[i].fillna(car[i].mean())
    else:
        car[i]=car[i].fillna(car[i].mode())
```

In [ ]:
```
car.isna().any()
```

Out[ ]:
```
Manufacturer           False
Model                  False
Sales in thousands     False
4-year resale value    False
Latest Launch          False
Price in thousands     False
Fuel capacity          False
Fuel efficiency        False
Wheelbase              False
dtype: bool
```

In [ ]:
```
car.head()
```

Out[ ]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelbase
0	Acura	Integra	16.919	16.360	02-Feb-14	21.500000	13.2	28.0	101.2
1	Acura	TL	39.384	19.875	06-Mar-15	28.400000	17.2	25.0	108.1
2	Acura	CL	14.114	18.225	01-Apr-14	27.390755	17.2	26.0	106.9
3	Acura	RL	8.588	29.725	03-Oct-15	42.000000	18.0	22.0	114.6
4	Audi	A4	20.397	22.255	10-Aug-15	23.990000	16.4	27.0	102.6

Question 20 - Intermediate (Bridging Question)

Task to be performed:

1. **Group by the manufacturer and print the average 4-year resale value in descending order.**

In [ ]:
```
car.groupby('Manufacturer')['4-year resale value'].mean().sort_values(ascending=False)
```

Out[ ]:
```
Manufacturer
Porsche          56.475000
Mercedes-Benz    29.648875
Audi             28.270000
BMW              27.624325
Lexus            25.607321
Cadillac         22.329595
Acura            21.046250
Lincoln          20.107658
Infiniti         19.690000
Saab             18.072975
Subaru           18.072975
Jaguar           18.072975
Volvo            18.072975
Oldsmobile       17.073492
Dodge            16.961634
Toyota           16.814775
Honda            15.557000
Chrysler         15.406139
Jeep             15.353333
Volkswagen       14.966329
Buick            14.941250
Nissan           14.886564
Pontiac          14.532163
Mitsubishi       14.262143
Mercury          13.970000
Chevrolet        13.768664
Ford             13.424816
Saturn           13.345190
Plymouth         11.911994
Hyundai           7.531667
Name: 4-year resale value, dtype: float64
```

Question 21 - Intermediate

Tasks to be performed:

1. **Find the best fuel-efficient *Model* and *Manufacturer*.**
2. **Print details of models made by Audi using groupby method.**

(Hint: String should be cleaned before operation)

In [ ]:
```
#Task 1:
car[car['Fuel efficiency']==car['Fuel efficiency'].max()][['Manufacturer','Model']]
```

Out[ ]:

	Manufacturer	Model
26	Chevrolet	Metro

In [ ]:
```
#Task 2:
car.Manufacturer=car.Manufacturer.str.strip()
group=car.groupby('Manufacturer')
group.get_group('Audi')
```

Out[ ]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelbase
4	Audi	A4	20.397	22.255	10-Aug-15	23.99	16.4	27.0	102.6
5	Audi	A6	18.780	23.555	08-Sep-15	33.95	18.5	22.0	108.7
6	Audi	A8	1.380	39.000	27-Feb-14	62.00	23.7	21.0	113.0

Question 22 - Intermediate

Tasks to be performed:

1. **Print the data of car where *Sales in thousands* is between 200 to 300.**
2. **Sample 20 rows from the dataset randomly.**

In [ ]:
```
#Task 1:
car[car['Sales in thousands'].between(200,300,inclusive=True)]
```

Out[ ]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheell
40	Dodge	Ram Pickup	227.061	15.060	03-Jun-14	19.460	26.0	17.0	1
49	Ford	Taurus	245.815	10.055	20-Dec-15	17.885	16.0	24.0	1
52	Ford	Explorer	276.747	16.640	25-Apr-14	31.930	21.0	19.0	1
55	Ford	Ranger	220.650	7.850	14-Jan-14	12.050	20.0	23.0	1
58	Honda	Accord	230.902	13.210	20-May-14	15.350	17.1	27.0	1
137	Toyota	Camry	247.994	13.245	02-Oct-15	17.518	18.5	27.0	1

In [ ]:
```
#Task 2:
car=car.sample(n=20)
car
```

Out[ ]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	W
97	Mercedes-Benz	SLK230	1.526	18.072975	08-Jun-15	41.000	14.0	27.000000	
27	Chevrolet	Impala	107.995	18.072975	18-Jun-15	18.890	17.0	27.000000	
112	Oldsmobile	Bravada	20.017	19.925000	21-Sep-15	31.598	17.5	19.000000	
78	Lincoln	Navigator	22.925	18.072975	23-Dec-14	42.660	30.0	15.000000	
39	Dodge	Viper	0.916	58.470000	08-Jul-15	69.725	19.0	16.000000	
69	Jeep	Grand Cherokee	157.040	18.810000	12-Oct-15	26.895	20.5	19.000000	
38	Dodge	Intrepid	88.028	12.275000	06-Feb-14	22.505	17.0	23.844156	
85	Mitsubishi	Montero Sport	39.348	13.880000	18-May-14	22.527	19.5	20.000000	
119	Pontiac	Grand Am	131.097	10.290000	26-Nov-14	19.720	15.2	25.000000	
109	Oldsmobile	Intrigue	38.554	18.072975	04-Jan-15	24.150	18.0	23.844156	
134	Subaru	Outback	47.107	18.072975	07-Jul-15	22.695	16.9	25.000000	
58	Honda	Accord	230.902	13.210000	20-May-14	15.350	17.1	27.000000	
31	Chrysler	Cirrus	32.306	12.640000	10-Jun-15	16.480	16.0	27.000000	
131	Saturn	SW	5.223	10.790000	15-Jan-15	14.290	12.1	31.000000	
45	Dodge	Caravan	181.749	12.025000	09-Jan-15	19.565	20.0	24.000000	
28	Chrysler	Sebring Coupe	7.854	12.360000	16-Jan-14	19.840	15.9	24.000000	
114	Plymouth	Neon	32.734	7.750000	26-Apr-15	12.640	12.5	29.000000	
11	Buick	Regal	39.350	13.740000	09-Mar-15	25.300	17.5	23.000000	
111	Oldsmobile	Aurora	14.690	19.890000	18-Feb-15	36.229	18.5	22.000000	
147	Volkswagen	Passat	51.102	16.725000	30-Oct-14	21.200	16.4	27.000000	

Question 23 - Beginner

Task to be performed:

1. **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 [ ]:
```
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: int64
```

Question 24 - Intermediate

Given below are two DataFrames:

```
df1 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Lee','Gloria'] ,
                    'Age': [18, 27, 18, 27, 17] ,
                    'Hobby': ['music','binge-watching','dancing','music','music']})


df2 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Janna','Cohle'] ,
                    'Age': [23, 21, 21, 25, 18],
                    'Hobby': ['painting','dancing','dancing','gaming','binge-watchi
ng']})
```

Task to be performed:

1. **Perform left and right joins respectively.**

In [ ]:
```
import pandas as pd
df1 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Lee','Gloria'] ,
                    'Age': [18, 27, 18, 27, 17] ,
                    'Hobby': ['music','binge-watching','dancing','music','music'
]})
df2 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Janna','Cohle'] ,
                        'Age': [23, 21, 21, 25, 18],
                        'Hobby': ['painting','dancing','dancing','gaming','binge-w
atching']})
print(pd.merge(df1,df2,how='left',left_on=['Name'], right_on=['Name']))
print()
print(pd.merge(df1,df2,how='right',left_on=['Name'], right_on=['Name']))
```

```
     Name  Age_x         Hobby_x  Age_y   Hobby_y
0     Sam     18           music   23.0  painting
1    Aron     27  binge-watching   21.0   dancing
2   Smith     18         dancing   21.0   dancing
3     Lee     27           music    NaN       NaN
4  Gloria     17           music    NaN       NaN

    Name  Age_x         Hobby_x  Age_y         Hobby_y
0    Sam   18.0           music     23        painting
1   Aron   27.0  binge-watching     21         dancing
2  Smith   18.0         dancing     21         dancing
3  Janna    NaN             NaN     25          gaming
4  Cohle    NaN             NaN     18  binge-watching
```

Question 25 - Advanced

Task to be performed:

1. **Create a Pandas DataFrame having keys and ltable and rtable as below -**

```
'key': ['One', 'Two'], 'ltable': [1, 2]
'key': ['One', 'Two'], 'rtable': [4, 5]
```

Merge both the tables based on key.

In [ ]:
```
import pandas as pd
sj=pd.DataFrame({'key': ['One', 'Two'], 'ltable': [1, 2]})
sk=pd.DataFrame({'key': ['One', 'Two'], 'rtable': [4, 5]})
c=pd.concat([sj,sk],axis=1,keys=['One','Two'])
c
```

Out[ ]:

	One		Two	
	key	ltable	key	rtable
0	One	1	One	4
1	Two	2	Two	5

Question 26 - Intermediate (Bridging Questions)

Task to be performed:

1. **Create a Python DataFrame from the below data:**

Food	PortionSize	Calories	K joules
Blueberries	100g	30	128
Cranberries	100g	15	65
Cherries	120g	39	168
Coconut	110g	351	1446

Rearrange columns as : Food, Calories, K joules, PortionSize.

In [ ]:
```
import pandas as pd
fruits=pd.DataFrame({'Food':['Blueberries','Cranberries','Cherrries','Coconut'
],
                     'PortionSize':['100g','100g','120g','110g'],
                     'Calories': [30,15,39,351],
                     'K joules':[128,65,168,1446]})
fruits
```

Out[ ]:

	Food	PortionSize	Calories	K joules
0	Blueberries	100g	30	128
1	Cranberries	100g	15	65
2	Cherrries	120g	39	168
3	Coconut	110g	351	1446

In [ ]:
```
fruits.reindex(columns=['Food','Calories','K joules','PortionSize'])
```

Out[ ]:

	Food	Calories	K joules	PortionSize
0	Blueberries	30	128	100g
1	Cranberries	15	65	100g
2	Cherrries	39	168	120g
3	Coconut	351	1446	110g

Question 27 - Advanced (Bridging Question)

Datasets Description:

laliga_player_stats.csv - The dataset contains 14 features. Here's a brief description of a few columns in the dataset:

- **Team - Team of the player**
- **Position - Position of the player**
- **Name - Name of the player**
- **Minutes played - Minutes played by the player**
- **Games played - Games played by the player**
- **Yellow Cards - Yellow cards given to the player**
- **Red Cards - Red cards given to the player**
- **Goals scored - Goals scored by the player**
- **Penalties scored - Penalties scored by the player**

Tasks to be performed:

1. **Group the dataset by position and print the average timing of each position.**
2. **Print the name and position of the player who scored the most number of goals.**
3. **Print the team name that scored the maximum number of goals.**
4. **Find the team with the least number of yellow cards.**

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/vkhzdt1f98oyyv4/laliga_player_stats.csv
```

```
--2020-06-30 06:12:18--  https://www.dropbox.com/s/vkhzdt1f98oyyv4/laliga_pla
yer_stats.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/vkhzdt1f98oyyv4/laliga_player_stats.csv [following]
--2020-06-30 06:12:19--  https://www.dropbox.com/s/raw/vkhzdt1f98oyyv4/laliga
_player_stats.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc739247e31b7213d7e431d4eb68.dl.dropboxusercontent.com/cd/
0/inline/A6lLqxwndQSKub6cmGybcCtVvgFkP9O4TMkteyArMr20nQVuseMxlOeHHVNmaPEemX_u
l9Cj7R9r4yvHt_gZ_CZ0Zxz0E7Jk4HyRA2rL1Oo15DGCF45O4nHZ-RL8EIvU6lY/file# [follow
ing]
--2020-06-30 06:12:19--  https://uc739247e31b7213d7e431d4eb68.dl.dropboxuserc
ontent.com/cd/0/inline/A6lLqxwndQSKub6cmGybcCtVvgFkP9O4TMkteyArMr20nQVuseMxlO
eHHVNmaPEemX_ul9Cj7R9r4yvHt_gZ_CZ0Zxz0E7Jk4HyRA2rL1Oo15DGCF45O4nHZ-RL8EIvU6l
Y/file
Resolving uc739247e31b7213d7e431d4eb68.dl.dropboxusercontent.com (uc739247e31
b7213d7e431d4eb68.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc739247e31b7213d7e431d4eb68.dl.dropboxusercontent.com (uc73924
7e31b7213d7e431d4eb68.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 36400 (36K) [text/plain]
Saving to: ‘laliga_player_stats.csv’

laliga_player_stats 100%[===================>]  35.55K  --.-KB/s    in 0.02s

2020-06-30 06:12:19 (1.85 MB/s) - ‘laliga_player_stats.csv’ saved [36400/3640
0]
```

In [ ]:
```
#Task 1:
import pandas as pd
stats=pd.read_csv('laliga_player_stats.csv')

stats.groupby('Position')['Minutes played'].mean()
```

Out[ ]:
```
Position
Defender      121.626973
Forward       181.430862
Goalkeeper    113.551719
Midfielder    108.873855
Name: Minutes played, dtype: float64
```

In [ ]:
```
#Task 2:
stats[stats['Goals scored']==max(stats['Goals scored'])][['Name','Position']]
```

Out[ ]:

	Name	Position
138	Messi	Forward

In [ ]:
```
#Task 3:
stats.groupby('Team')['Goals scored'].sum().sort_values(ascending=False).index
[0]
```

Out[ ]: 'FC Barcelona'

In [ ]:
```
#Task 4:
stats.groupby('Team')['Yellow Cards'].sum().sort_values().index[0]
```

Out[ ]: 'RC Celta'

Question 28 - Advanced

Tasks to be performed:

1. **Write a Python program to show different methods of converting Python structures into DataFrames.**

In [ ]:
```
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.S
eries([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
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
```

Module 7 - Visualizing Data

- Using easy to understand visualizations, information can be quickly shared
- Visualizations help to find hidden patterns & relationships in data
- It is easy to track trends like sales, shares & so on
- Storytelling with data creates impactful results

Matplotlib

- Matplotlib is a Python library that is specially designed for the development of graphical elements such as plots and charts for interactive data visualization

Seaborn

- Seaborn is a Python data visualization library based on Matplotlib
- It has high-level interface for drawing attractive and informative statistical graphics

Questions 1-4:

Dataset Description:

Manufacturer_Car_data.csv - The dataset contains 8 features. Here's a brief description of the columns in the dataset:

- **Manufacturer** - Name of the manufacturer
- **Model** - Name of the model
- **Width** - Width of the model
- **Length** - Length of the model
- **Fuel capacity** - Fuel capacity of the car
- **Fuel efficiency** - Fuel efficiency of the car
- **Horsepower** - Horsepower of the car

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/pk46bdflmvt3hib/Manufacturer_Car_data.csv
```

```
--2020-06-30 05:52:00--  https://www.dropbox.com/s/pk46bdflmvt3hib/Manufactur
er_Car_data.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.66.1, 2620:100:6022:
1::a27d:4201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.66.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/pk46bdflmvt3hib/Manufacturer_Car_data.csv [following]
--2020-06-30 05:52:00--  https://www.dropbox.com/s/raw/pk46bdflmvt3hib/Manufa
cturer_Car_data.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc754362a70c22a36efdc919b27f.dl.dropboxusercontent.com/cd/
0/inline/A6mym3TE9fpXMhEm-UKXnB-ajZ_9Pj6cuY35V39JrTYZ8RYN0hviomYeU3QuZwNK5yw6
10qUhBE9TPhhPS7Leby41unIeId9b9frpYpbJiUipUktsl6X2ahsdUDdIPOQU-4/file# [follow
ing]
--2020-06-30 05:52:01--  https://uc754362a70c22a36efdc919b27f.dl.dropboxuserc
ontent.com/cd/0/inline/A6mym3TE9fpXMhEm-UKXnB-ajZ_9Pj6cuY35V39JrTYZ8RYN0hviom
YeU3QuZwNK5yw610qUhBE9TPhhPS7Leby41unIeId9b9frpYpbJiUipUktsl6X2ahsdUDdIPOQU-
4/file
Resolving uc754362a70c22a36efdc919b27f.dl.dropboxusercontent.com (uc754362a70
c22a36efdc919b27f.dl.dropboxusercontent.com)... 162.125.66.15, 2620:100:6022:
15::a27d:420f
Connecting to uc754362a70c22a36efdc919b27f.dl.dropboxusercontent.com (uc75436
2a70c22a36efdc919b27f.dl.dropboxusercontent.com)|162.125.66.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 9548 (9.3K) [text/plain]
Saving to: ‘Manufacturer_Car_data.csv’

Manufacturer_Car_da 100%[===================>]   9.32K  --.-KB/s    in 0s

2020-06-30 05:52:01 (264 MB/s) - ‘Manufacturer_Car_data.csv’ saved [9548/954
8]
```

Question 1 - Beginner

Tasks to be performed:

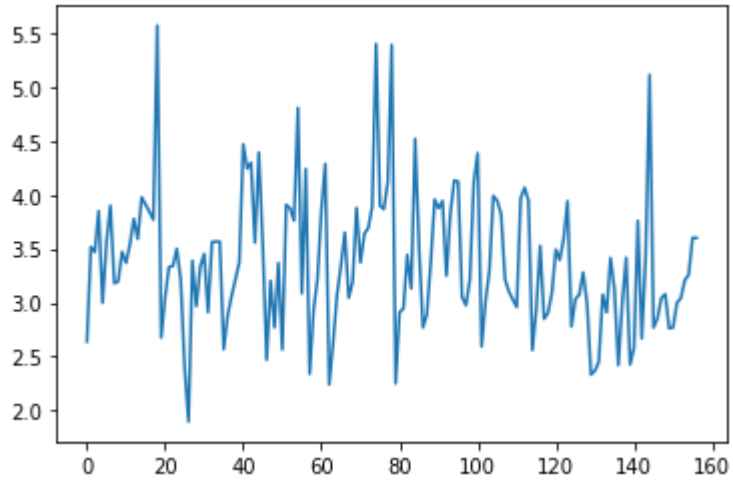
1. Import the dataset and remove any null values.
2. Plot Curb weight on simple plot.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
m_car=pd.read_csv('Manufacturer_Car_data.csv')
m_car.isna().any()
```

Out[ ]:
```
Manufacturer       False
Model              False
Width               True
Length              True
Curb weight         True
Fuel capacity       True
Fuel efficiency     True
Horsepower          True
dtype: bool
```

In [ ]:
```
m_car=m_car.dropna()
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.plot(m_car['Curb weight'])
plt.show()
```



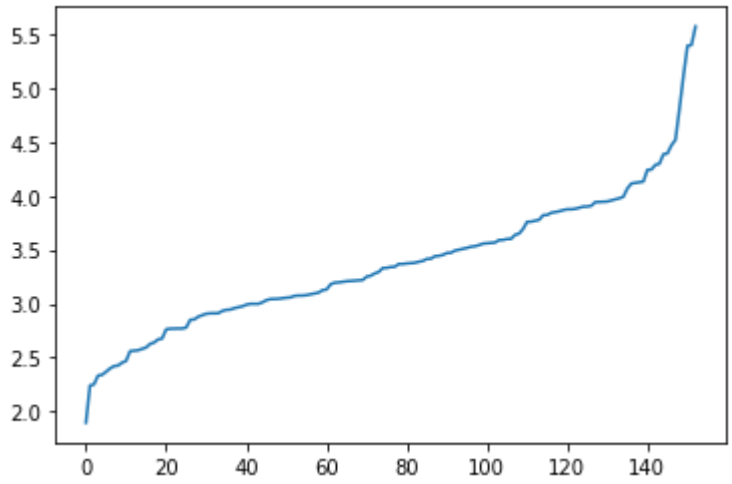
Question 2 - Beginner

Tasks to be performed:

1. Sort the dataset by *Curb weight*.
2. Plot the simple plot again.

In [ ]:
```
m_car=m_car.sort_values('Curb weight')
m_car=m_car.reset_index(drop=True)
plt.plot(m_car['Curb weight'])
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fdbed8f7ac8>]
```



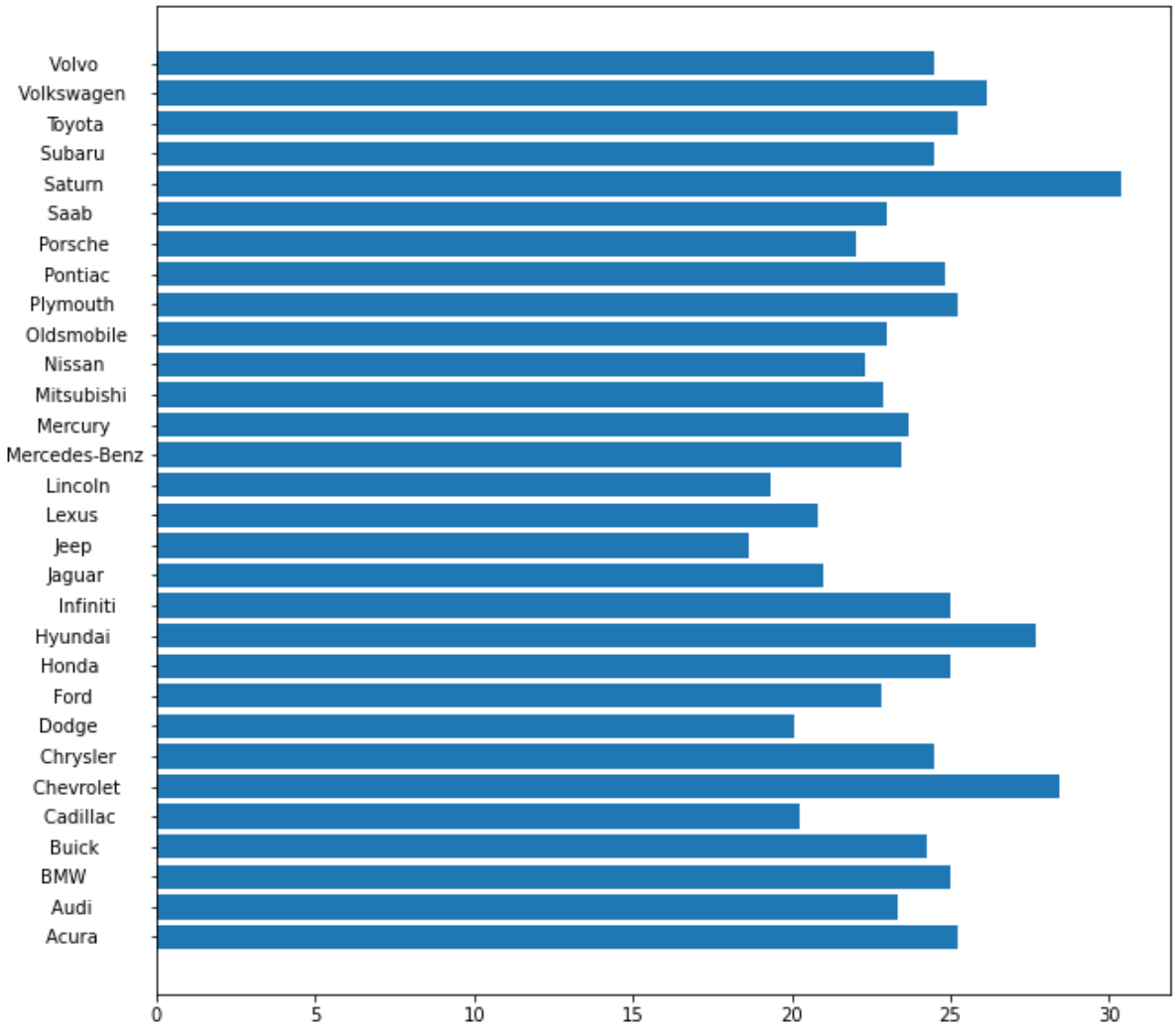
Question 3 - Beginner

Which manufacturers provide maximum fuel efficiency and fuel capacity?

Tasks to be performed:

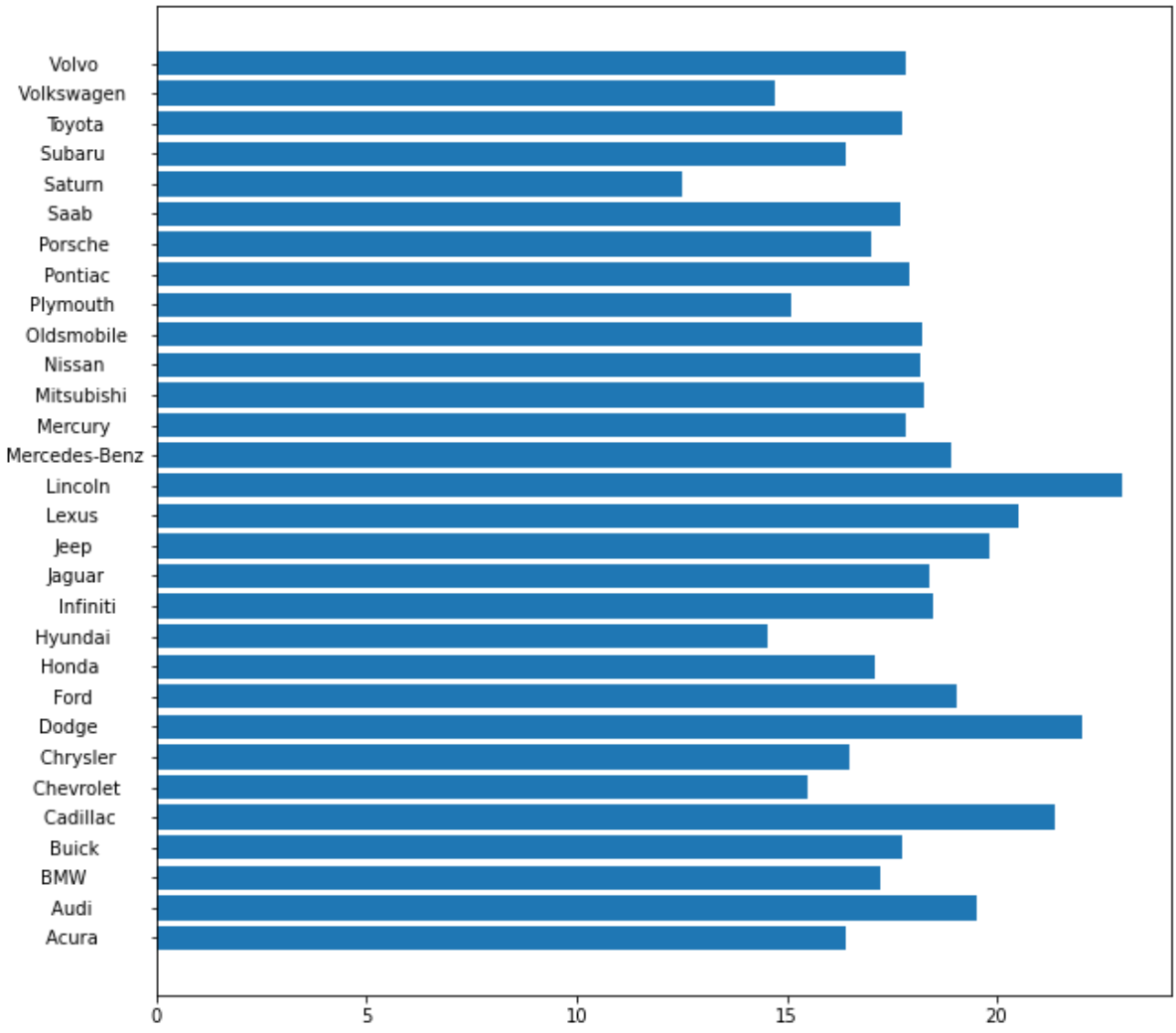
1. Group by *Manufacturer*.
2. Use barplot to plot *Fuel efficiency* and *Fuel capacity* of each Manufacturer.

In [ ]:
```
group=m_car.groupby('Manufacturer')
fuel_eff=[]
fuel_cap=[]
grp=[]
for gp,data in group:
    grp.append(gp)
    fuel_eff.append(data['Fuel efficiency'].mean())
    fuel_cap.append(data['Fuel capacity'].mean())
plt.figure(figsize=(10,10))
plt.barh(grp,fuel_eff)
plt.show()
```



From the graph above, we can see that **Saturn** provides maximum fuel efficiency amongst all manufacturers.

In [ ]:
```
plt.figure(figsize=(10,10))
plt.barh(grp,fuel_cap)
plt.show()
```



From the graph above, we can see that **Lincoln** provides maximum fuel capacity amongst all manufacturers.

Question 4 - Beginner

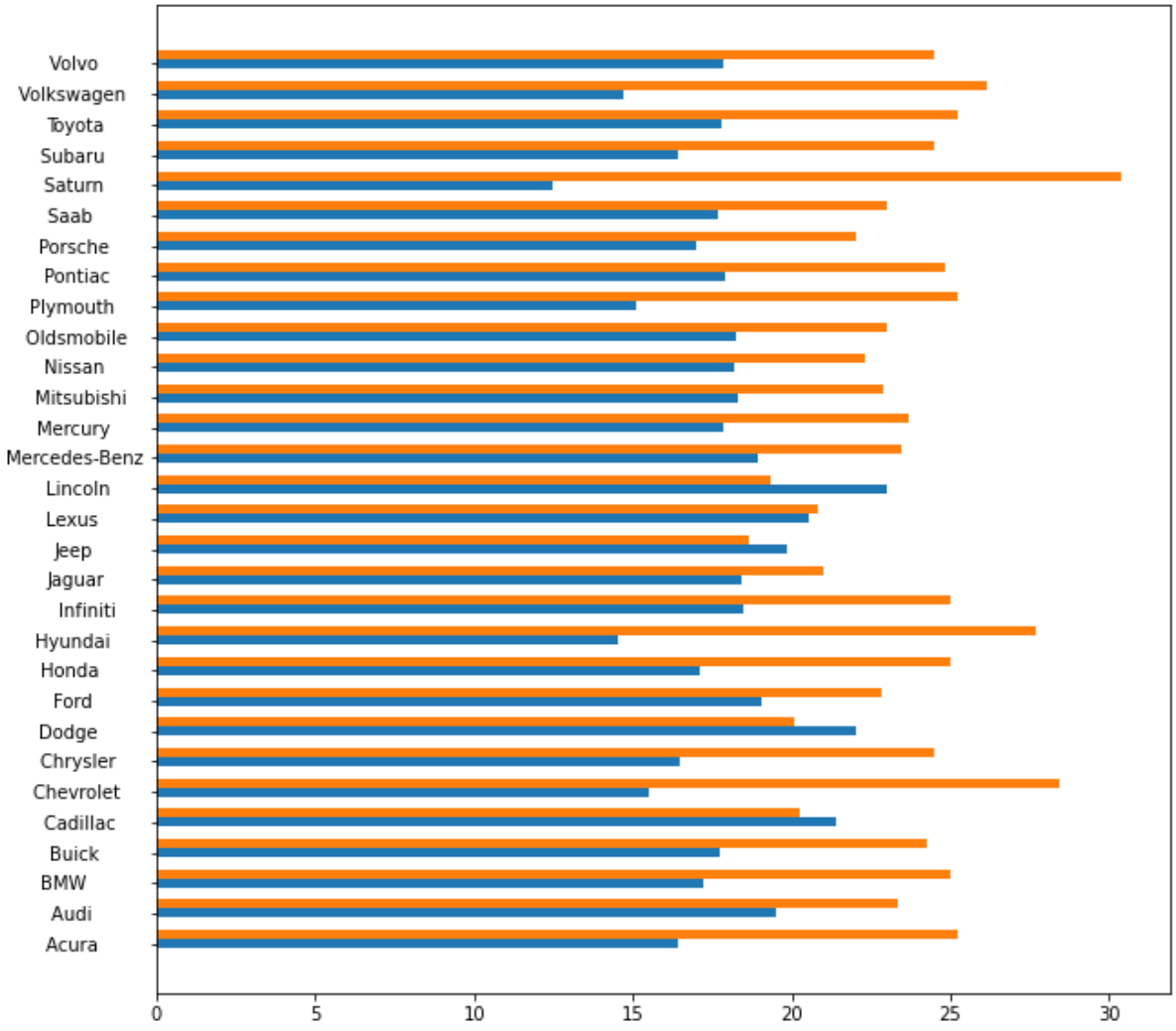
Which manufacturer has the most closely comparable fuel efficiency and fuel capacity?

Task to be performed:

1. Using a grouped bar chart, plot the same in a single horizontal barplot.

In [ ]:
```
grp_height=0.3
plt.figure(figsize=(10,10))
plt.barh(np.arange(len(grp))-grp_height,fuel_cap,height=grp_height,tick_label=
grp)
plt.barh(np.arange(len(grp)),fuel_eff,height=grp_height)
```

Out[ ]:
```
<BarContainer object of 30 artists>
```



From the graph above, we can see that **Lexus** has the most closely comparable fuel efficiency and fuel capacity.

Questions 5-9:

Dataset Description:

supermarket_sales.csv - The dataset contains 17 features. Here's a brief description of 10 columns in the dataset:

- **City** - City name
- **Customer type** - Customer type (Member/Normal)
- **Gender** - Gender of the customer
- **Product line** - Product line
- **Unit price** - Price of each unit of the product
- **Total** - Total purchase
- **Payment** - Mode of payment
- **gross margin percentage** - Gross margin percentage
- **gross income** - Gross income
- **Rating** - Rating of the product

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/jm09bhnctkd7ft5/supermarket_sales.csv
```

```
--2020-07-01 13:47:18--  https://www.dropbox.com/s/jm09bhnctkd7ft5/supermarke
t_sales.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/jm09bhnctkd7ft5/supermarket_sales.csv [following]
--2020-07-01 13:47:18--  https://www.dropbox.com/s/raw/jm09bhnctkd7ft5/superm
arket_sales.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc11977747b6f21ed3e7affbd476.dl.dropboxusercontent.com/cd/
0/inline/A6snu9CUGUuLR96XqwK2NEGerBETJ932cBW-IM82b7COl8_mxxkZtEtmVPAZVYEmt2QX
a_8XdARGa6sW_ASAoVomzhxF4rZJeIQ9Q25BaZkGSvBOOebkX5jG1Lv5f8ufBbY/file# [follow
ing]
--2020-07-01 13:47:19--  https://uc11977747b6f21ed3e7affbd476.dl.dropboxuserc
ontent.com/cd/0/inline/A6snu9CUGUuLR96XqwK2NEGerBETJ932cBW-IM82b7COl8_mxxkZtE
tmVPAZVYEmt2QXa_8XdARGa6sW_ASAoVomzhxF4rZJeIQ9Q25BaZkGSvBOOebkX5jG1Lv5f8ufBb
Y/file
Resolving uc11977747b6f21ed3e7affbd476.dl.dropboxusercontent.com (uc11977747b
6f21ed3e7affbd476.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc11977747b6f21ed3e7affbd476.dl.dropboxusercontent.com (uc11977
747b6f21ed3e7affbd476.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 131528 (128K) [text/plain]
Saving to: ‘supermarket_sales.csv’

supermarket_sales.c 100%[===================>] 128.45K  --.-KB/s    in 0.04s

2020-07-01 13:47:19 (3.38 MB/s) - ‘supermarket_sales.csv’ saved [131528/13152
8]
```

Question 5 - Beginner

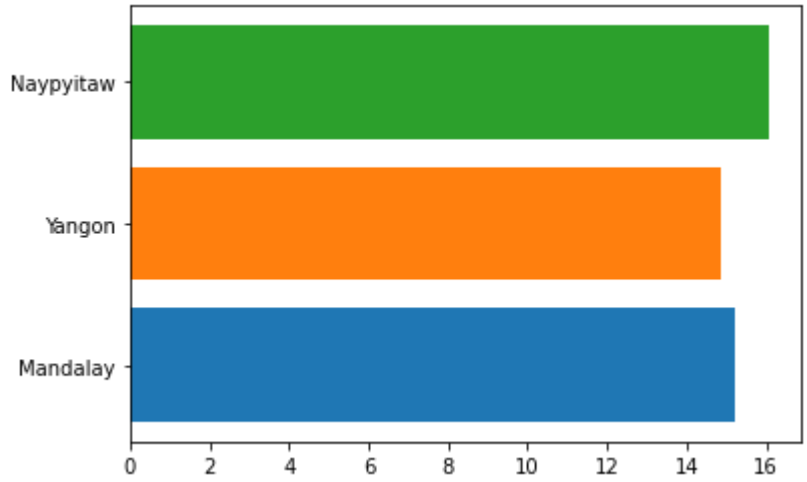
Which city has the highest average gross income?

Tasks to be performed:

1. Import the dataset.
2. Plot a simple bar chart of the average gross income of all cities.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
sales=pd.read_csv('supermarket_sales.csv')

cities=set(sales.City)
for city in cities:
    plt.barh(city,sales[sales.City==city]['gross income'].mean())
```



From the graph above, we can see that **Naypyitaw** has the highest average gross income.

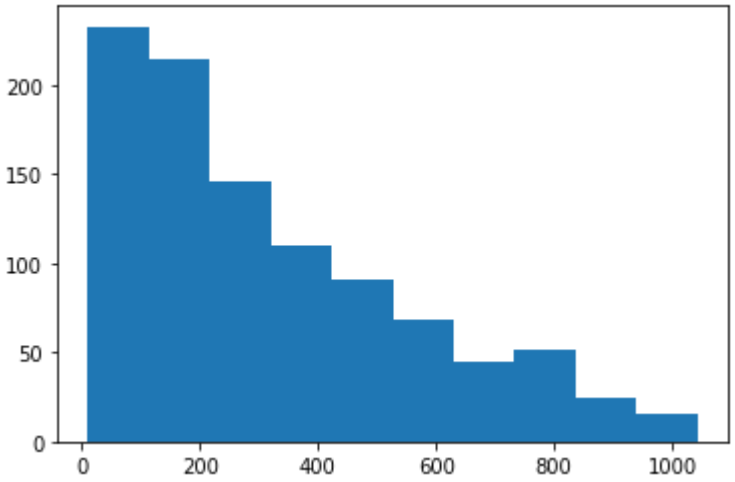
Question 6 - Beginner

Around how many customers made purchases of more than $200?

Task to be performed:

1. Create a histogram for Total purchases.

In [ ]:
```
plt.hist(sales.Total)
plt.show()
```



From the above graph, we can see that around 200 customers made purchases of more than $200.

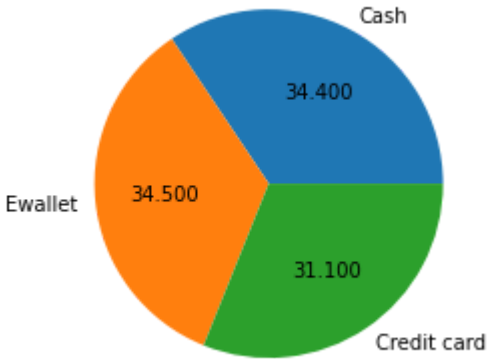
Question 7 - Beginner

Find out which mode of payment is the most used.

Task to be performed:

1. Create a pie chart to represent which mode of payment is the most used.

In [ ]:
```
mop=list(set(sales.Payment))
val=[]
for mode in mop:
    val.append(sales[sales.Payment==mode].Payment.count())
plt.pie(val,labels=mop,autopct='%0.3f')
plt.show()
```



From the above graph, we can see that **e-wallets** are the most used modes of payment.

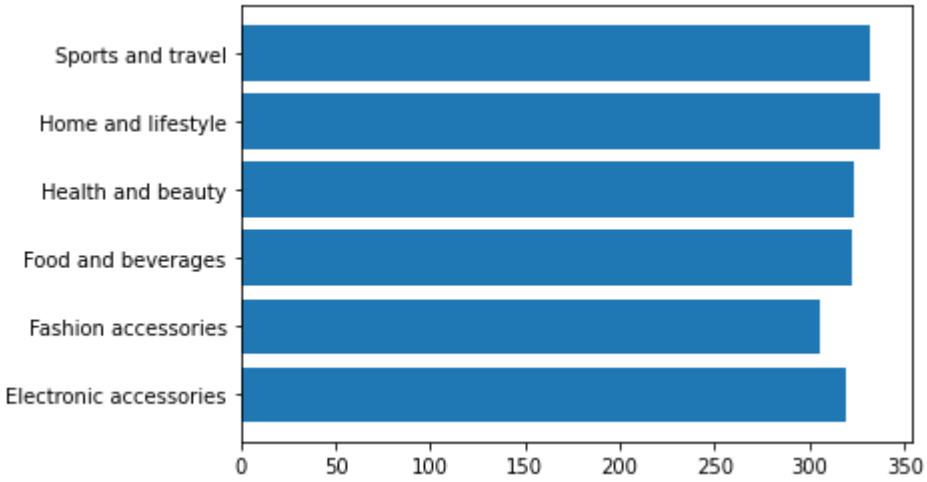
Question 8 - Beginner

Which product line has the highest numbe of purchases?

Tasks to be performed:

1. Group by *Product Line*.
2. Then create a horizontal bar chart to represent mean total purchase in a particular product line.

In [ ]:
```
my_dict=dict()
for grp,data in sales.groupby('Product line'):
    my_dict[grp]=data.Total.mean()
plt.barh(list(my_dict.keys()),list(my_dict.values()))
plt.show()
```



From the above graph, we can see that **Home and lifestyle** product line has the highest number of purchases.

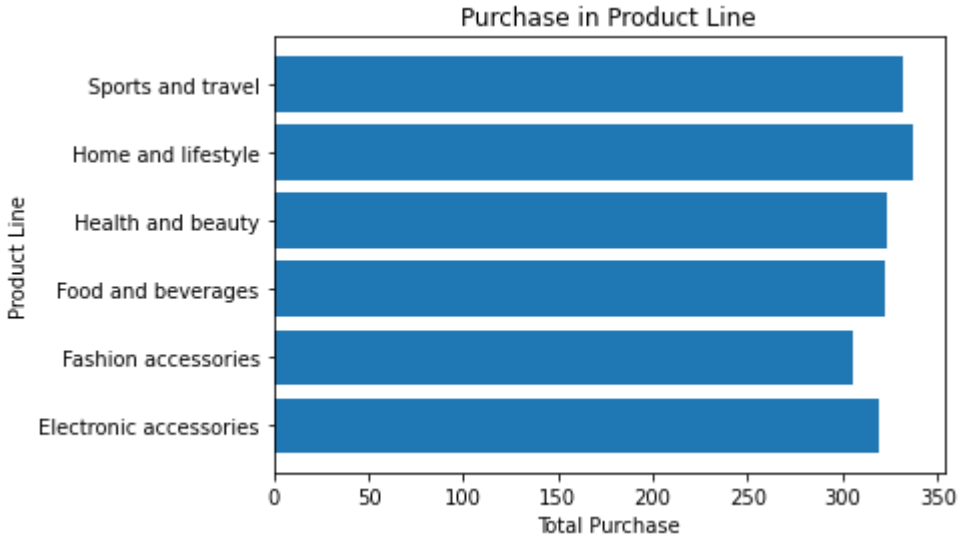
Question 9 - Beginner

Make the above graph more readable.

Tasks to be performed:

1. Label X and Y axes along with the title of the graph and save it as **'graph.png'**.

In [ ]:
```
plt.barh(list(my_dict.keys()),list(my_dict.values()))
plt.title('Purchase in Product Line')
plt.xlabel('Total Purchase')
plt.ylabel('Product Line',rotation=90)
plt.show()
plt.savefig('graph.png')
```




```
<Figure size 432x288 with 0 Axes>
```

Question 10 - Beginner

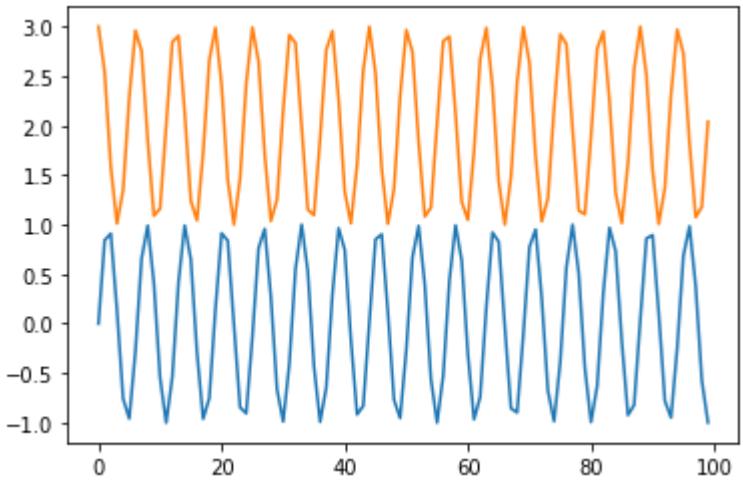
Create a line plot to represent the following equations:

- y=cos(x)+2
- y=sin(x)

Task to be performed:

1. Use NumPy to generate the data and create a line plot to represent the above equations.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
x=np.arange(100)
y=np.arange(100)
e1=np.sin(x)
e2=np.cos(x)+2
plt.plot(y,e1)
plt.plot(y,e2)
plt.show()
```

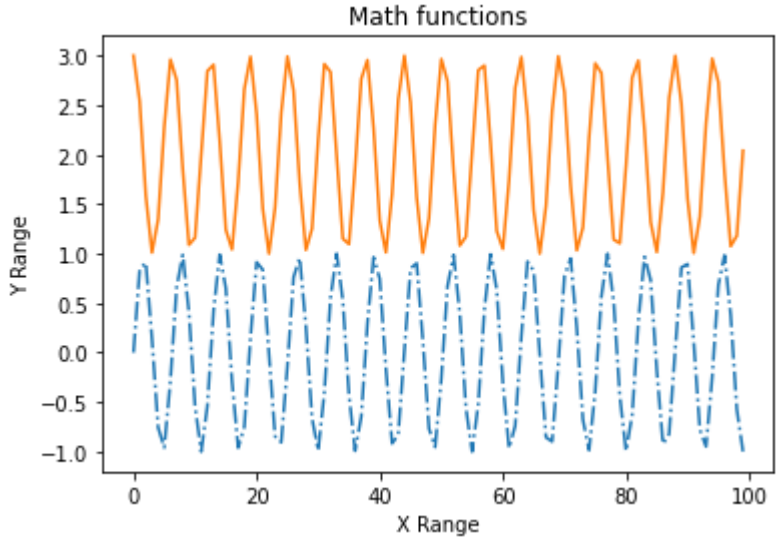


Question 11 - Beginner

Task to be performed:

1. Add labels and title to the graph in Question 10 and use different line styles.

In [2]:
```
plt.plot(y,e1,'-.')
plt.plot(y,e2,'-')
plt.xlabel('X Range')
plt.ylabel('Y Range')
plt.title('Math functions')
plt.show()
```

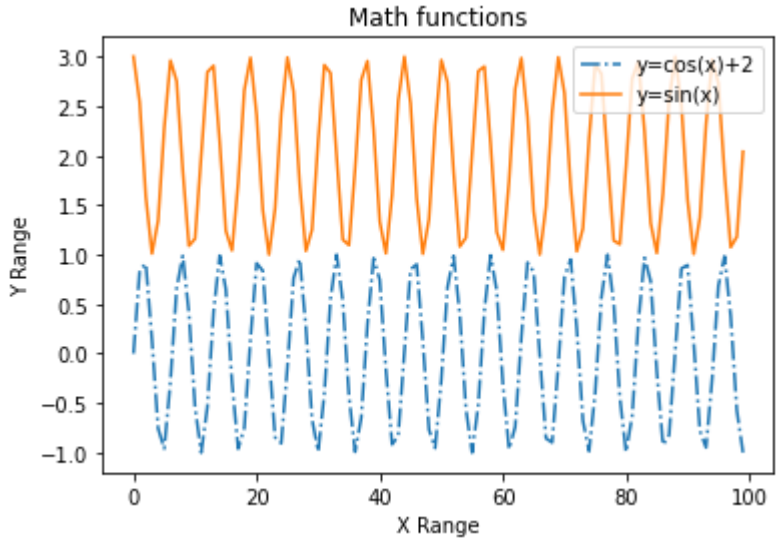


Question 12 - Beginner

Task to be performed:

1. Create legends for the graph in Question 11.

In [3]:
```
plt.plot(y,e1,'-.')
plt.plot(y,e2,'-')
plt.xlabel('X Range')
plt.ylabel('Y Range')
plt.title('Math functions')
plt.legend(['y=cos(x)+2','y=sin(x)'])
plt.show()
```



Questions 13-14:

Dataset Description:

Mall_Customers.csv - The dataset contains 5 features. Here's a brief description of the columns in the dataset:

- **CustomerID** - CustomerID in serial order
- **Gender** - Gender of the customer
- **Age** - Age of the customer
- **Annual Income** - Annual income (in $) of the customer
- **Spending Score** - Spending score (1-100) of the customer

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/ojuoca138sokhu6/Mall_Customers.csv
```

```
--2020-06-30 05:55:03--  https://www.dropbox.com/s/ojuoca138sokhu6/Mall_Custo
mers.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.66.1, 2620:100:6022:
1::a27d:4201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.66.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/ojuoca138sokhu6/Mall_Customers.csv [following]
--2020-06-30 05:55:03--  https://www.dropbox.com/s/raw/ojuoca138sokhu6/Mall_C
ustomers.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucdb5af9aa256416b8ac77e0aa13.dl.dropboxusercontent.com/cd/
0/inline/A6l3aLj5LpZq09vICvlQsOv9J_Qe6T1y3-cyDq1uF-nws8JuQYZGmxPaLvY5EP7qaV-o
9KpoTNm1C1kj5rTOlIuw1bv6e9iz2Nph0ZUFhF-my1GVlMlLhoI6zzE_KupSbT0/file# [follow
ing]
--2020-06-30 05:55:03--  https://ucdb5af9aa256416b8ac77e0aa13.dl.dropboxuserc
ontent.com/cd/0/inline/A6l3aLj5LpZq09vICvlQsOv9J_Qe6T1y3-cyDq1uF-nws8JuQYZGmx
PaLvY5EP7qaV-o9KpoTNm1C1kj5rTOlIuw1bv6e9iz2Nph0ZUFhF-my1GVlMlLhoI6zzE_KupSbT
0/file
Resolving ucdb5af9aa256416b8ac77e0aa13.dl.dropboxusercontent.com (ucdb5af9aa2
56416b8ac77e0aa13.dl.dropboxusercontent.com)... 162.125.66.15, 2620:100:6022:
15::a27d:420f
Connecting to ucdb5af9aa256416b8ac77e0aa13.dl.dropboxusercontent.com (ucdb5af
9aa256416b8ac77e0aa13.dl.dropboxusercontent.com)|162.125.66.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 3968 (3.9K) [text/plain]
Saving to: ‘Mall_Customers.csv’

Mall_Customers.csv  100%[===================>]   3.88K  --.-KB/s    in 0s

2020-06-30 05:55:04 (567 MB/s) - ‘Mall_Customers.csv’ saved [3968/3968]
```

Question 13 - Intermediate

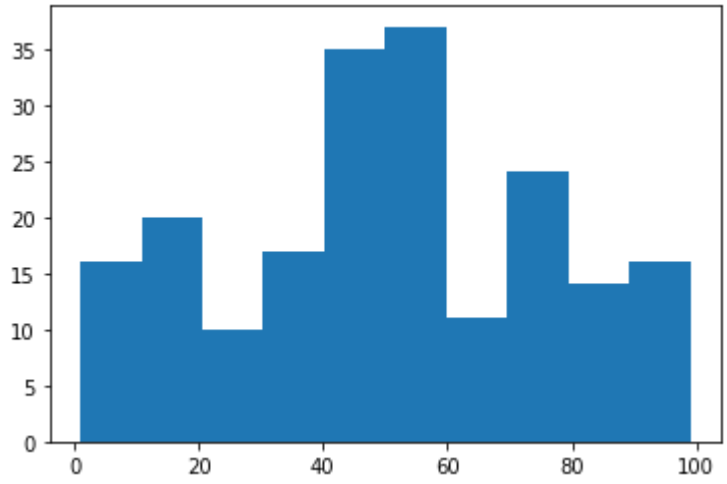
What is the most common spending score?

Tasks to be performed:

1. Import the dataset.
2. Plot a histogram on *Spending Score* with 10 bins.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
mall=pd.read_csv('Mall_Customers.csv')

plt.hist(mall['Spending Score'],bins=10)
plt.show()
```



From the above graph, we can see that most people have a credit score between 50-60.

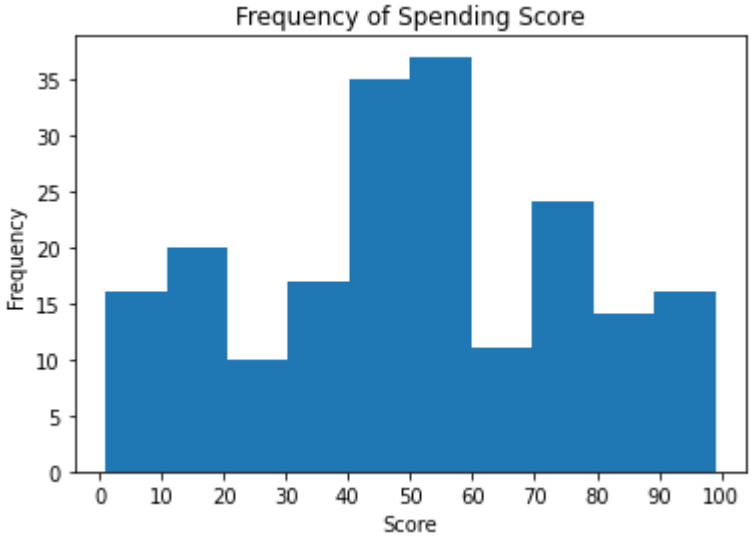
Question 14 - Intermediate

Make the above graph more readable.

Tasks to be performed:

1. Add labels and title along with the proper xticks.
2. Conclude what the histogram tells you.

In [ ]:
```
plt.hist(mall['Spending Score'],bins=10)
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.title('Frequency of Spending Score')
plt.xticks(np.arange(0,101,10))
plt.show()
```



Questions 15-16:

Sylphia has a dataset of various cereals sold in the supermarket.

Dataset Description:

cereal.csv - The dataset contains 16 features. Here's a brief description of 3 columns in the dataset:

- **name** - Brand name of the cereals
- **MFR** - Manufacturer of the brands
- **rating** - Rating of the cereals

In [ ]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/idnul34dfo5cnke/cereal.csv
```

```
--2020-07-01 13:57:11--  https://www.dropbox.com/s/idnul34dfo5cnke/cereal.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:
1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/idnul34dfo5cnke/cereal.csv [following]
--2020-07-01 13:57:12--  https://www.dropbox.com/s/raw/idnul34dfo5cnke/cerea
l.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc1d8e6fe76d040891c90da1819c.dl.dropboxusercontent.com/cd/
0/inline/A6uQeVdvi-DUTnZ-B27senR0_wape7SgruUGJ5zqFoY3-7zNdS5CoMOXFtGCelrPreFC
JN2gv-CM3rHOyMBDkoMkNggx-h2GjrMBhYzsXiH-y5NopZkbtSVSQDIrTLho5DU/file# [follow
ing]
--2020-07-01 13:57:12--  https://uc1d8e6fe76d040891c90da1819c.dl.dropboxuserc
ontent.com/cd/0/inline/A6uQeVdvi-DUTnZ-B27senR0_wape7SgruUGJ5zqFoY3-7zNdS5CoM
OXFtGCelrPreFCJN2gv-CM3rHOyMBDkoMkNggx-h2GjrMBhYzsXiH-y5NopZkbtSVSQDIrTLho5D
U/file
Resolving uc1d8e6fe76d040891c90da1819c.dl.dropboxusercontent.com (uc1d8e6fe76
d040891c90da1819c.dl.dropboxusercontent.com)... 162.125.82.15, 2620:100:6032:
15::a27d:520f
Connecting to uc1d8e6fe76d040891c90da1819c.dl.dropboxusercontent.com (uc1d8e6
fe76d040891c90da1819c.dl.dropboxusercontent.com)|162.125.82.15|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 5052 (4.9K) [text/plain]
Saving to: ‘cereal.csv’

cereal.csv          100%[===================>]   4.93K  --.-KB/s    in 0s

2020-07-01 13:57:13 (493 MB/s) - ‘cereal.csv’ saved [5052/5052]
```

Question 15 - Intermediate

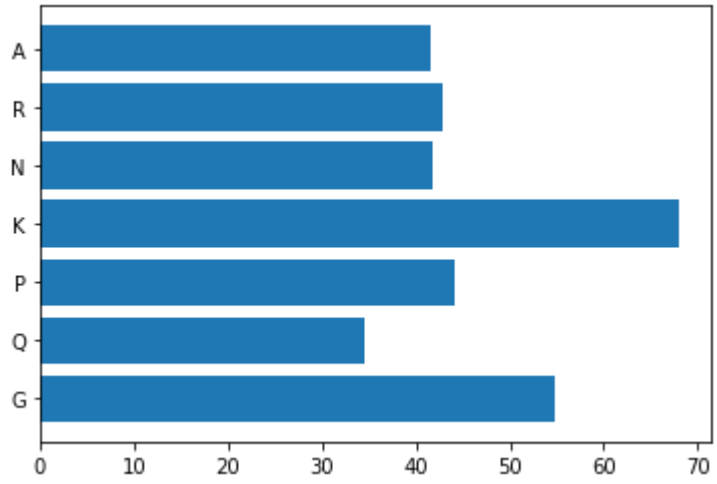
Sylphia wants to visualize the quality of cereals and determine which manufacturer delivers the best quality.

Tasks to be performed:

1. Import the dataset.
2. Plot ratings of different types of manufacturers.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
cereal=pd.read_csv('cereal.csv')

plt.barh(list(set(cereal.mfr)),cereal.groupby('mfr').rating.mean())
plt.show()
```



From the above graph, we can see that manufacturer **K** delivers the best quality of cereal.

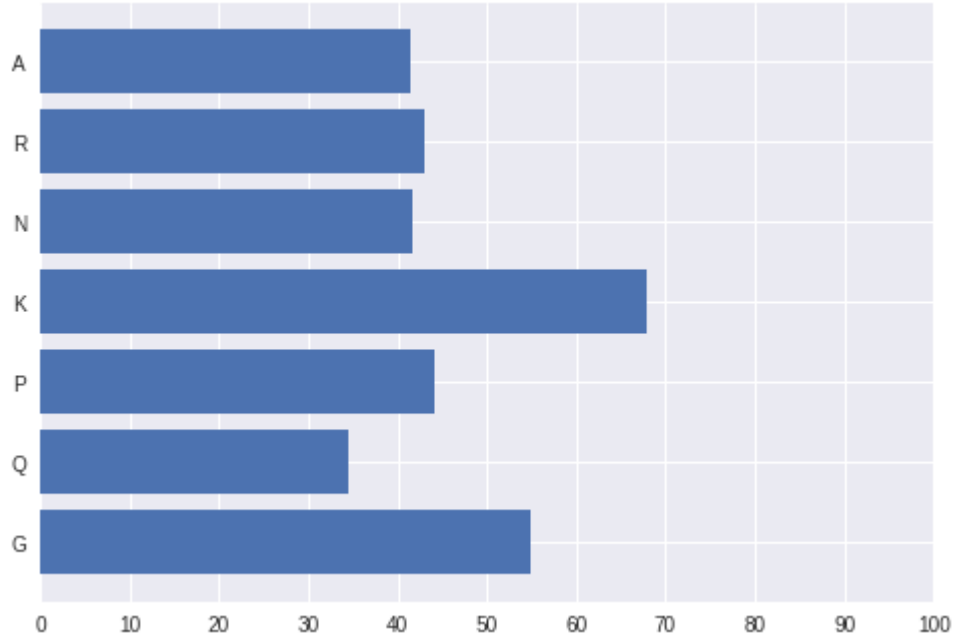
Question 16 - Intermediate

Change the style of the above graph.

Tasks to be performed:

1. Use xticks range form 0-100.
2. Change style of the graph to seaborn.

In [ ]:
```
plt.style.use('seaborn')
plt.barh(list(set(cereal.mfr)),cereal.groupby('mfr').rating.mean())
plt.xticks(np.arange(0,101,10))
plt.show()
```



Questions 17-20:

Dataset Description:

rainfall.csv - The dataset contains 19 features. Here's a brief description of a few columns in the dataset:

- **SUBDIVISION** - Geographic sub-division
- **YEAR** - Year
- **12 columns - ((JAN to DEC** - Rainfall in cms each month))
- **ANNUAL** - Annual rainfall in cms

In [8]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/rmiumhvw54ybl8s/rainfall.csv
```

```
--2020-08-03 10:27:39--  https://www.dropbox.com/s/rmiumhvw54ybl8s/rainfall.c
sv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::
a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/rmiumhvw54ybl8s/rainfall.csv [following]
--2020-08-03 10:27:40--  https://www.dropbox.com/s/raw/rmiumhvw54ybl8s/rainfa
ll.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc0ca395f0e47d71efbe09842ad1.dl.dropboxusercontent.com/cd/
0/inline/A8siilZsMxpL0Y5JLQiWEu2yBGei_EcGcMjxpvopJ3CPzABDKLRmDui3NAPkLpnDHNpn
VKxPes0roSOgnzQ6qgR_TN_eqTO-eOLwu5yhTqZcju3n-z5LIV-zUxIT44nXTtQ/file# [follow
ing]
--2020-08-03 10:27:40--  https://uc0ca395f0e47d71efbe09842ad1.dl.dropboxuserc
ontent.com/cd/0/inline/A8siilZsMxpL0Y5JLQiWEu2yBGei_EcGcMjxpvopJ3CPzABDKLRmDu
i3NAPkLpnDHNpnVKxPes0roSOgnzQ6qgR_TN_eqTO-eOLwu5yhTqZcju3n-z5LIV-zUxIT44nXTt
Q/file
Resolving uc0ca395f0e47d71efbe09842ad1.dl.dropboxusercontent.com (uc0ca395f0e
47d71efbe09842ad1.dl.dropboxusercontent.com)... 162.125.5.15, 2620:100:601d:1
5::a27d:50f
Connecting to uc0ca395f0e47d71efbe09842ad1.dl.dropboxusercontent.com (uc0ca39
5f0e47d71efbe09842ad1.dl.dropboxusercontent.com)|162.125.5.15|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 528115 (516K) [text/plain]
Saving to: ‘rainfall.csv’

rainfall.csv        100%[===================>] 515.74K  --.-KB/s    in 0.1s

2020-08-03 10:27:41 (4.44 MB/s) - ‘rainfall.csv’ saved [528115/528115]
```

Question 17 - Intermediate

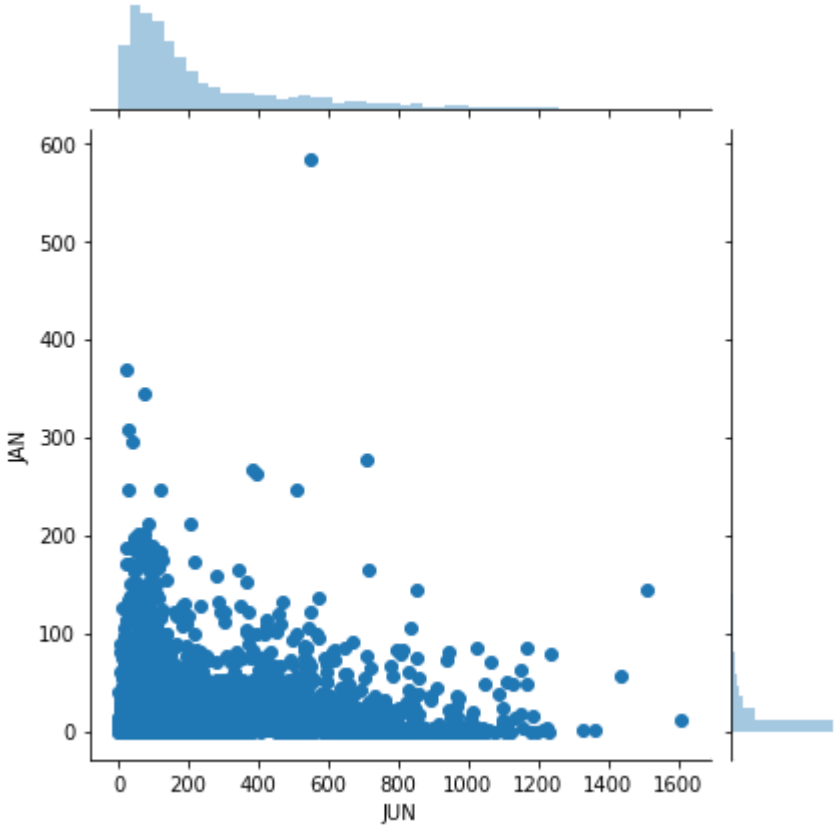
Is there any evident relationship between rainfall in the month of January and June?

Tasks to be performed:

1. Import the dataset.
2. Create a jointplot for *JUN* and *JAN*.

In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
rain=pd.read_csv('rainfall.csv')

import seaborn as sns
sns.jointplot(rain.JUN,rain.JAN)
plt.show()
```



From the above plot, we can see that there is no relationship as such between January and June in terms of rainfall.

Question 18 - Advanced

Which sub-division has the highest rainfall and which state has the lowest?

Task to be performed:

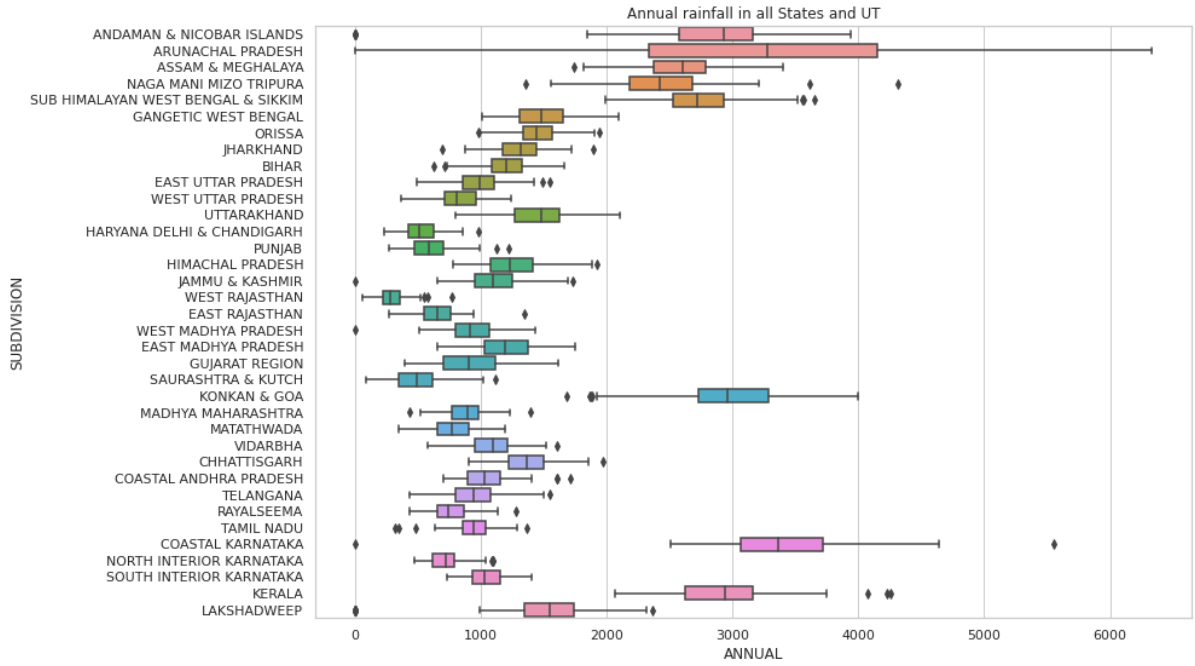
1. Generate a boxplot for every sub-division.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
rain=rain.fillna(0)

sns.set(rc={'figure.figsize':(13,9)}, font_scale=1, style='whitegrid')

sns.boxplot(y='SUBDIVISION', x='ANNUAL', data=rain)
plt.title('Annual rainfall in all States and UT')
```

Out[ ]:
```
Text(0.5, 1.0, 'Annual rainfall in all States and UT')
```



From the above graph, we can see that **Arunachal Pradesh** has the highest rainfall and **West Rajasthan** has the lowest rainfall.

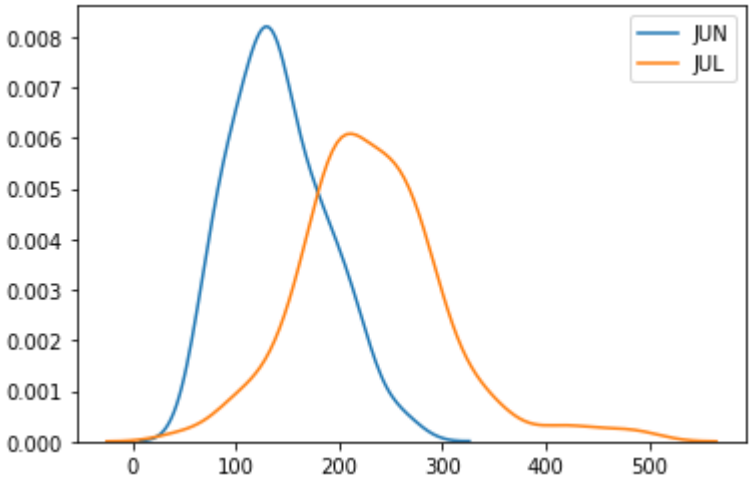
Question 19 - Advanced

Between June and July, which of the two months has more rainfall?

Tasks to be performed:

1. Create a KDEplot for the month of *JUN* & *JAN*. Take *SOUTH INTERIOR KARNATAKA* as the sub-division.
2. Draw a relevant conclusion from the plot.

In [ ]:
```
sns.kdeplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUN)
sns.kdeplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUL)
plt.show()
```



There is more rainfall in the month of June.

Question 20 - Advanced

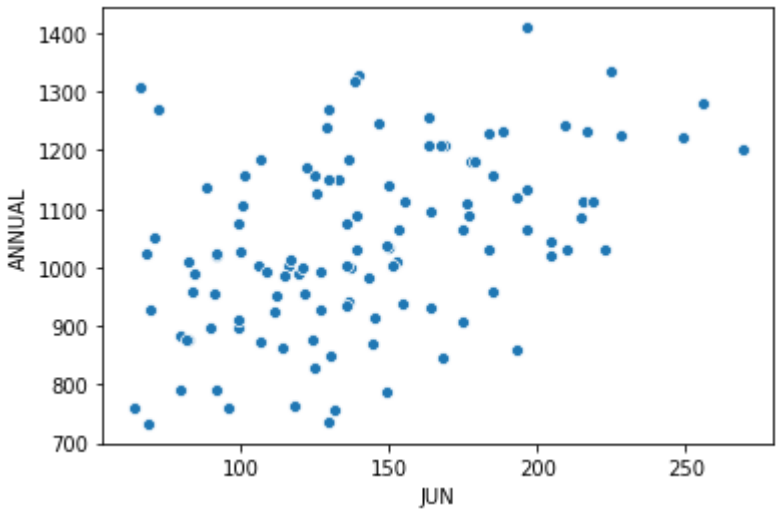
Between January and June, which of the two is more closely related to Annual Rainfall?

Tasks to be performed:

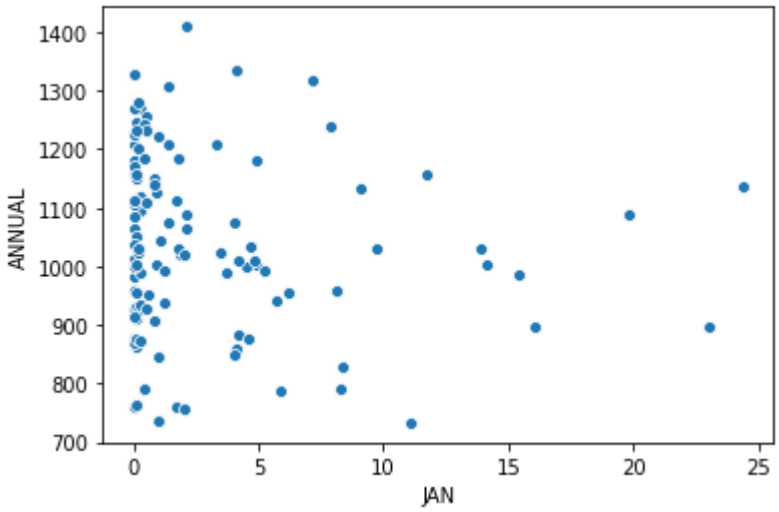
Using seaborn, create a scatter plot for *SUBDIVISION=SOUTH INTERIOR KARNATAKA*

1. For the month JUN and ANNUAL rainfall
2. For the month JAN and ANNUAL rainfall

In [ ]:
```
sns.scatterplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUN,rain[ra
in.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].ANNUAL)
plt.show()
```



In [ ]:
```
sns.scatterplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JAN,rain[ra
in.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].ANNUAL)
plt.show()
```



The month of June has a low positive correlation with the Annual Rainfall. The month of January seems mostly uncorrelated.

Questions 21-27:

Dataset Description:

Car_sales.csv - The dataset contains 15 features. Here's a brief description of 12 columns in the dataset:

- **Manufacturer** - Manufacturer of the car
- **Model** - Model of the car
- **Sales in thousands** - Sales in thousands
- **4-year resale value** - 4-year resale value of the car
- **Vehicle type** - Vehicle type
- **Price** - Price in thousands
- **Engine size** - Engine size of the car
- **Horsepower** - Horsepower of the car
- **Width** - Width of the car
- **Length** - Length of the car
- **Fuel capacity** - Fuel capacity of the car
- **Fuel efficiency** - Fuel efficiency of the car

In [11]:
```
#fetch and download the datset from dropbox
!wget https://www.dropbox.com/s/w1ijf6gy8h7n5tc/Car_sales.csv
```

```
--2020-08-03 10:28:07--  https://www.dropbox.com/s/w1ijf6gy8h7n5tc/Car_sales.
csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.1, 2620:100:601d:1::
a27d:501
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.1|:443... connecte
d.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/w1ijf6gy8h7n5tc/Car_sales.csv [following]
--2020-08-03 10:28:07--  https://www.dropbox.com/s/raw/w1ijf6gy8h7n5tc/Car_sa
les.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc4db72f6e09dc5526cf0dd5618a.dl.dropboxusercontent.com/cd/
0/inline/A8thay-3-VB3GIHQjB38UUYGPAsZd81IHoJKQW1M4SucTIOptcBokisnEMCPPrEWkvYY
eo7Xk4XY-AIn0hJAMFnenj0TBKt0Yt2kq528HsZzdlngN4ZmyrrcrgbMQ6w1NSQ/file# [follow
ing]
--2020-08-03 10:28:07--  https://uc4db72f6e09dc5526cf0dd5618a.dl.dropboxuserc
ontent.com/cd/0/inline/A8thay-3-VB3GIHQjB38UUYGPAsZd81IHoJKQW1M4SucTIOptcBoki
snEMCPPrEWkvYYeo7Xk4XY-AIn0hJAMFnenj0TBKt0Yt2kq528HsZzdlngN4ZmyrrcrgbMQ6w1NS
Q/file
Resolving uc4db72f6e09dc5526cf0dd5618a.dl.dropboxusercontent.com (uc4db72f6e0
9dc5526cf0dd5618a.dl.dropboxusercontent.com)... 162.125.5.15, 2620:100:601d:1
5::a27d:50f
Connecting to uc4db72f6e09dc5526cf0dd5618a.dl.dropboxusercontent.com (uc4db72
f6e09dc5526cf0dd5618a.dl.dropboxusercontent.com)|162.125.5.15|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 16760 (16K) [text/plain]
Saving to: ‘Car_sales.csv’

Car_sales.csv       100%[===================>]  16.37K  --.-KB/s    in 0s

2020-08-03 10:28:08 (197 MB/s) - ‘Car_sales.csv’ saved [16760/16760]
```

Question 21 - Intermediate

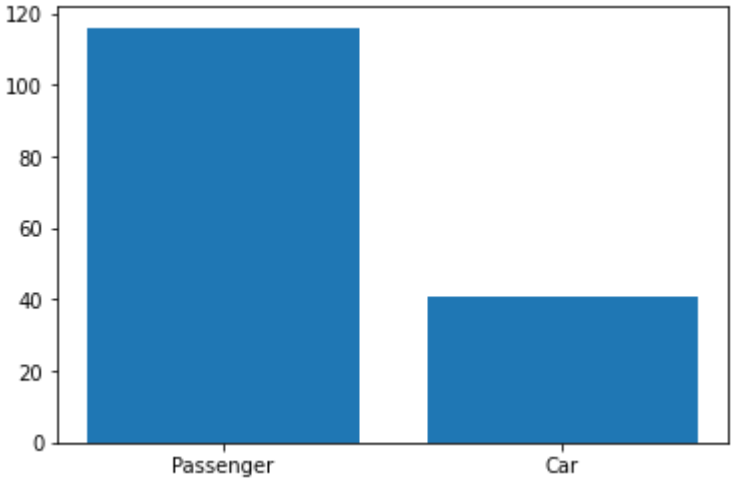
Draw visual conclusion on vehical types.

Tasks to be performed:

1. Import the dataset.
2. Generate a barchart for *Vehicle type* to see the count of vehicle types.

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
car=pd.read_csv('Car_sales.csv')

label=list(set(car['Vehicle type']))
count=[]
for i in label:
    count.append(car[car['Vehicle type']==i].Manufacturer.count())
plt.bar(label,count)
plt.show()
```



From the above graph, we can see more than twice the number of passenger vehicles than cars.

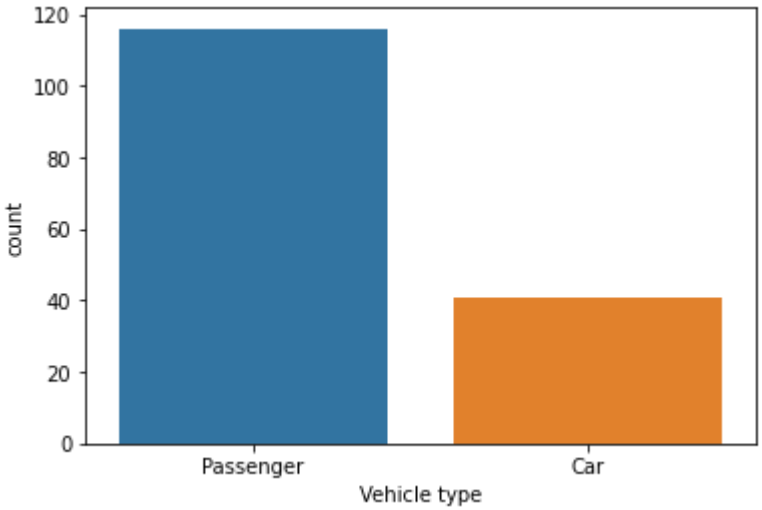
Question 22 - Intermediate

Estimate the number of passenger vehicles and cars.

Task to be performed:

1. Generate a countplot for *Vehicle type* to see the count of vehicle types.

In [13]:
```
sns.countplot(car['Vehicle type'])
plt.show()
```



From the above graph, we can infer that there are close to 115 passenger vehicles and about 40 cars.

Question 23 - Intermediate

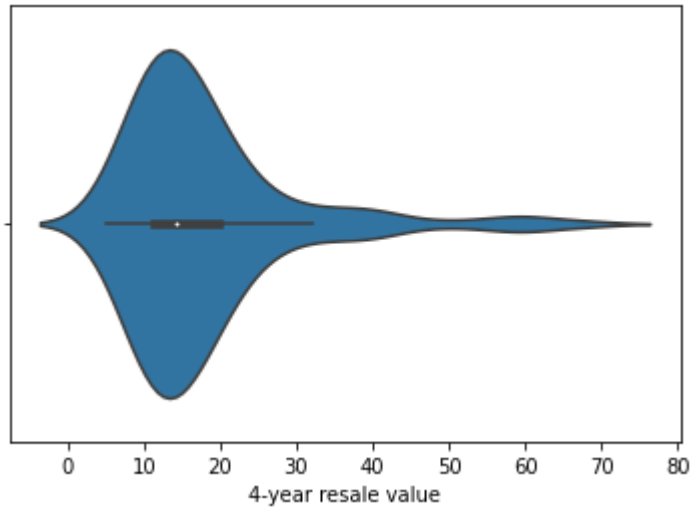
Visualize the 4-year resale value of the vehicles.

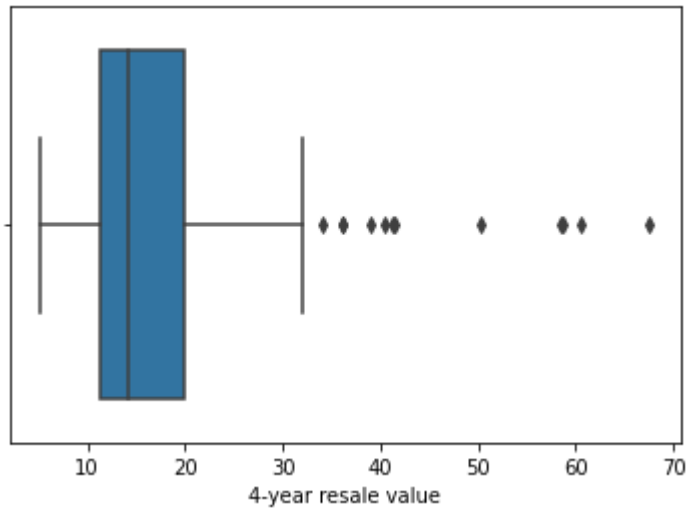
Tasks to be performed:

For *4-year resale value*,

1. Create a violin plot.
2. Create a boxplot.

In [14]:
```
sns.violinplot(car['4-year resale value'])
plt.show()
sns.boxplot(car['4-year resale value'])
plt.show()
```





Question 24 - Advanced

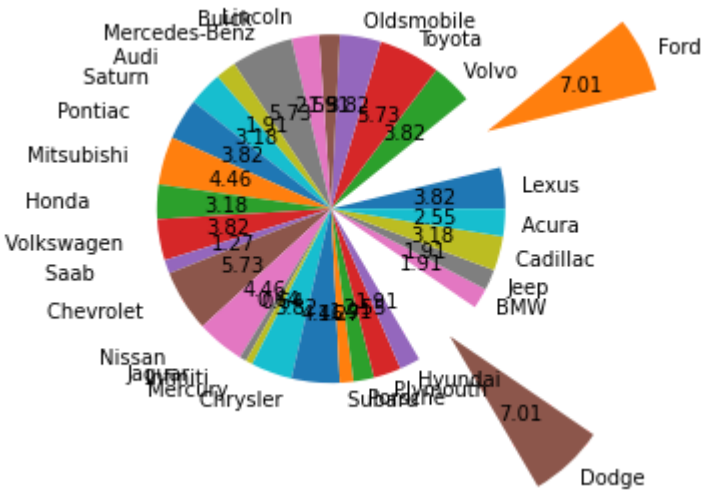
Find out which manufacturer made more than 10 models.

Task to be performed:

1. Create a pie chart and highlight (explode) the manufacturer who made more than 10 models.

In [15]:
```
val=[]
for i in set(car.Manufacturer):
    val.append(car[car.Manufacturer==i].Manufacturer.count())
temp_df=pd.DataFrame({'Manufacturer':list(set(car.Manufacturer)),'Count':val})


explode_max=np.zeros(len(val))
indx=list(temp_df[temp_df.Count>10].index)
explode_max[indx]=1
plt.pie(val,labels=list(set(car.Manufacturer)),autopct='%0.2f',explode=explode
_max)
plt.show()
```



From the above chart, we can see **Volkswagen** and **Dodge** made more than 10 models.

Question 25 - Advanced

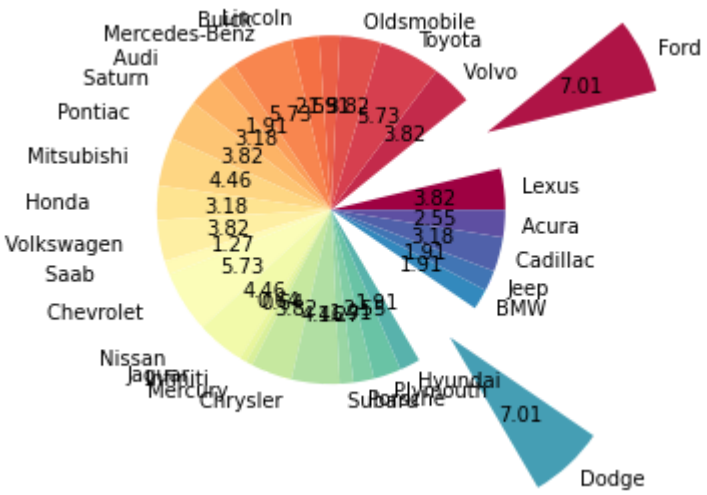
Make the above graph more appealing using colors.

Task to be performed:

1. Create a custom color list and generate the pie chart again using new colors.

(Hint: Use linspace to generate a custom color list)

In [16]:
```
cmap = plt.get_cmap('Spectral')
colors = [cmap(i) for i in np.linspace(0,1,len(set(car.Manufacturer)))]
plt.pie(val,labels=list(set(car.Manufacturer)),autopct='%0.2f',explode=explode
_max,colors=colors)
plt.show()
```



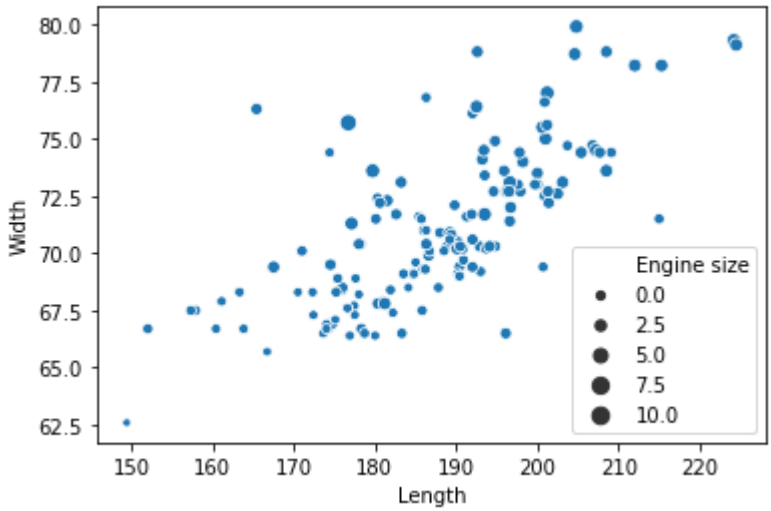
Question 26 - Advanced

Is there any relationship between the length and width of vehicles?

Task to be performed:

1. Generate scatterplot comparing *Length* and *Width* of the vehicles and use the *Engine size* to determine to the size of points.

In [17]:
```
sns.scatterplot(car.Length,car.Width,size=car['Engine size'])
plt.show()
```



From the above scatterplot, we can see that the length and width of vehicles have a positive correlation.

Question 27 - Advanced

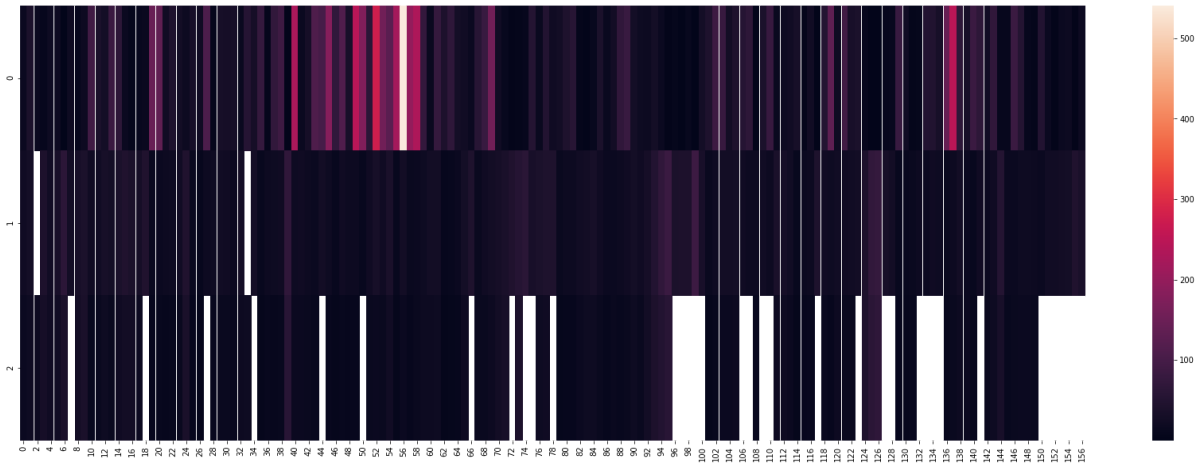
Task to be performed:

1. Generate a heatmap using the following columns:

   - *Sales in thousands*
   - *Price*
   - *4-year resale value*

In [30]:
```
fig, ax = plt.subplots(figsize=(30,10))
sns.heatmap([car['Sales in thousands'],car['Price'],car['4-year resale value']], linewidths=.0005, ax=ax)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7fac9ed61080>



In [ ]: