Creating Python Functions

1. Create functions with different numbers of parameters and return types.
2. Explore function scope and variable accessibility.
3. Implement functions with default argument values.
4. Write recursive functions.
5. Demonstrate how to use docstrings to document functions.

In [44]:
```
'''1. Create functions with different numbers of parameters and return types.'''
#Creating a multiply function that accepts 3 parameters as input
def mutliply(a,b,c):
    p=a*b*c
    return float(p)

#Creating a sum function that accepts a variable number of inputs
def sum(*args):
    s=0
    for number in args:
        s+=number
    return s

a=4
b=54
c=3
d=5
e=54
f=53

print(mutliply(a,b,c))
print(sum(a,b,c))
print(sum(a,b,c,d,e,f))
```

```
648.0
61
173
```

In [6]:
```
'''2. Explore function scope and variable accessibility.'''
def testFunction():
    localvar = 5    #Defining local variable
    global var      #Declaring global variable
    var = 10
    print(localvar+5)
    print(var + 5)

try:
    print(testFunction())
    print(var+5)
    print(localvar+5)       #Will throw an exception since it is a local variable
except:
    print('localvar not accessible as it is a local variable to testFunction')
```

```
10
15
None
15
localvar not accessible as it is a local variable to testFunction
```

In [32]:
```python
'''3. Implement functions with default argument values.'''
def function(*args, a = 20):
    s =0
    for i in args:
        s += i
    s += a
    return s

a=3
b=5
c=10
print(function(a,b,c))
```

```
38
```

In [ ]:
```python
'''3. Implement functions with default argument values.'''
def function(b,c, a = 20):
    s =0
    s += a+b+c
    return s

a=3
b=5
c=10
print(function(a,b,c)) #Giving 3 values overrides the constant parameter in the fun
```

```
18
```

In [ ]:
```python
'''4. Write recursive functions.'''
def apNterm(a,d,n):
    if n == 1:
        return a
    else:
        return(d+apNterm(a,d,n-1)) #Recursively calls the apNterm function with low

print(apNterm(2,7,11))
```

```
72
```

In [ ]:
```python
'''4. Write docstrings'''
def apNterm(a,d,n):
    """
    This fucntion takes 3 arguments and returns the Nth term of the AP
    Args:
    a (int) : Initial term
    d (int) : Common Difference
    n (int) : Nth term

    Returns:
    int : The Nth term of the given AP
```

```
    """
    if n == 1:
        return a
    else:
        return(d+apNterm(a,d,n-1)) #Recursively calls the apNterm function with low

print(apNterm(2,7,11))
```

Lambda Functions

1. Create simple lambda functions for various operations.
2. Use lambda functions with built-in functions like map, filter, and reduce.
3. Compare lambda functions with regular functions in terms of syntax and use cases.

In [61]:
```
'''Creating a simple Lambda functions'''

sum = lambda a, b, c : a+b+c   #function to calculate sum
print(sum(1,2,3))

interest = lambda p,r,t : (p*r*t)/100   #function to calculate simple interest
print(interest(1150000,12.1,6))
```

```
6
834900.0
```

In [ ]:
```
'''Lambda functions with builtin functions Map, Filter, Reduce'''

'''interest = lambda p,r,t : (p*r*t)/100   #Not able to figure out multiple argumen
values = [1150000,12.1,6]
result = map(interest,values)
print(list(result))'''

cube = lambda n: n**3
numbers = [1,2,3,4,5,6]
print(list(map(cube,numbers)))


import functools
values = [1150000,12.1,6]
interest = (functools.reduce(lambda x, y: x * y, values))/100
print(interest)

even_numbers = filter(lambda n: n%2==0, numbers)     # Use filter to filter out eve
print(list(even_numbers))

'''Regular function vs lambda function to check isEven or not for Numbers'''

def isEven(n):
    return n%2 == 0
isEvenResult = list(map(isEven,numbers))
print(numbers,isEvenResult)

isEvenlambda = lambda n: n%2==0
isEvenResultlambda = list(map(isEvenlambda,numbers))
print(numbers,isEvenResultlambda)
```

```
[1, 8, 27, 64, 125, 216]
834900.0
[2, 4, 6]
[1, 2, 3, 4, 5, 6] [False, True, False, True, False, True]
[1, 2, 3, 4, 5, 6] [False, True, False, True, False, True]
```

If Statements

1. Demonstrate conditional logic using if, else, and elif statements.
2. Create complex conditional expressions.
3. Implement nested if statements.

In [115…

```python
'''1. Demonstrate conditional logic using if, else, and elif statements.'''
a=43355464523
if(a%2==0):
    print('{} is divisible by 2'.format(a))
elif(a%3 == 0):
    print('{} is divisble by 3'.format(a))
else:
    print('{} is not divisible by either 2 or 3'.format(a))


'''2. Create complex conditional statements'''
age = 19
income = 45000
credit_score = 700

if age > 18 and income > 30000 and credit_score > 650:
    print("Eligible for premium loan")
elif (age > 18) & (income > 30000 | credit_score > 650):
    print("Eligible for standard loan")
else:
    print("Not eligible for any loan")

'''3. Implement nested if statements.'''
age = 19
student_status = True
income = 20000

if age >= 18:
    if student_status:
        if income < 25000:
            print("Eligible for a student loan")
        else:
            print("Income too high for a student loan")
    elif income > 30000:
        print("Eligible for a general loan")
    else:
        print("Not eligible for any loan due to low income")
else:
    print("Not eligible due to age restriction")
```

```
43355464523 is not divisible by either 2 or 3
Eligible for premium loan
Eligible for a student loan
```

Loops

1. Use for loops to iterate over sequences.
2. Employ while loops for indefinite iteration.
3. Implement nested loops.
4. Utilize break and continue statements.

In [135…

```
'''1. Use for loops to iterate over sequences.'''
sequence = [2,23e2,324,45,6,325,4456,54,324,'3sedf',342,'34523',34,'gfher']
for i in sequence:
    print(i)

'''2. Employ while loops for indefinite iteration.'''
while True:
    a=input('Enter a character')
    if(a=='t'):                  #indefinite while loop, terminated when 't' is en
        break
    else:
        pass

'''3. Implement nested loops.'''
for i in range(4):
    for j in range(4):
        print(i)

'''4. Utilize break and continue statements.'''
while True:
    a=input('Enter a character')
    if(a=='t'):                  #indefinite while loop, terminated when 't' is en
        break
    else:
        continue
```

```
2
2300.0
324
45
6
325
4456
54
324
3sedf
342
34523
34
gfher
0
0
0
0
1
1
1
1
2
2
2
2
3
3
3
3
```

Lists, Tuples, Sets, Dictionaries

1. Create and manipulate lists, tuples, sets, and dictionaries.
2. Understand the differences between these data structures.
3. Perform operations like indexing, slicing, adding, removing elements.
4. Explore built-in methods for each data structure.

In [96]:
```
fruits = list(('apple', 'orange', 'banana', 'pineapple'))
items = ['orange', 'kiwi', 'grape', 'mango', 'chickoo']
print(fruits)
print(items)
new_fruits = []
new_fruits.append(items)   #Appends entire list into the existing empty list
print(new_fruits)
print(new_fruits[0][0])     #Indexing the element in the list
for i in items:
    fruits.append(i)       #Appends items list to the end of fruits list
print(fruits)
sorted_fruits = fruits.sort()  #Sorts the fruits list
print(fruits)
fruits.insert(2,'jackfruit')  #Inserts new element
print(fruits)
selected_fruit = fruits.pop(2) #Removes element from the list
print(selected_fruit)
```

```
print(fruits)

citiesA = tuple(['Windsor', 'Toronto', 'Montreal', 'Chicago', 'Detroit'])
citiesB = ('Detroit', 'London', 'Ottawa', 'Hamilton')
print(citiesA[:])
print(citiesA[1:3])                #Indexing and slicing operation
print(citiesA[-4:-2])
print(citiesB.index('Detroit'))

set1 = {213,433.3,'sfs',"ynt",True}
print(set1)
set1.add('tejus')
print(set1)
set2 = set1.copy()   #Create a set copy
print(set2)
print(set1.difference(fruits))   #gets difference btw set and an iterable
print(set1.difference(set2))
print(set1.pop())           #removes a random element
print(set1)
for i in 'cnchj', False, 'trhjk', 654, 6557:
    set2.add(i)
print(set2)
print(set1.intersection(set2))      #Gets the common element in 2 sets
print(set1.difference(set2))        #Gets elements present in set1 but not set2
print(set2.difference(set1))
print(set1.symmetric_difference(set2))      #Gets the difference in elements betwee
print((set1.union(set2)).difference(set1))
set1.clear()
print(set1)

cars = { 'brand' : 'Ford',
         'Model' : 'Mustang',
         'Color' : 'Black' }
print(cars)
print(cars.values())
print(cars.keys())
cars['year'] = 1969
print(cars)
print(cars.values())
print(cars.keys())
cars.popitem()
cars.pop('Color')
print(cars)
```

```
['apple', 'orange', 'banana', 'pineapple']
['orange', 'kiwi', 'grape', 'mango', 'chickoo']
[['orange', 'kiwi', 'grape', 'mango', 'chickoo']]
orange
['apple', 'orange', 'banana', 'pineapple', 'orange', 'kiwi', 'grape', 'mango', 'chic
koo']
['apple', 'banana', 'chickoo', 'grape', 'kiwi', 'mango', 'orange', 'orange', 'pineap
ple']
['apple', 'banana', 'jackfruit', 'chickoo', 'grape', 'kiwi', 'mango', 'orange', 'ora
nge', 'pineapple']
jackfruit
['apple', 'banana', 'chickoo', 'grape', 'kiwi', 'mango', 'orange', 'orange', 'pineap
ple']
('Windsor', 'Toronto', 'Montreal', 'Chicago', 'Detroit')
('Toronto', 'Montreal')
('Toronto', 'Montreal')
0
{True, 433.3, 213, 'sfs', 'ynt'}
{True, 433.3, 213, 'sfs', 'ynt', 'tejus'}
{True, 433.3, 213, 'sfs', 'ynt', 'tejus'}
{True, 433.3, 213, 'sfs', 'ynt', 'tejus'}
set()
True
{433.3, 213, 'sfs', 'ynt', 'tejus'}
{False, True, 'ynt', 654, 213, 'sfs', 6557, 'cnchj', 'tejus', 433.3, 'trhjk'}
{'ynt', 'tejus', 433.3, 213, 'sfs'}
set()
{False, True, 'cnchj', 654, 'trhjk', 6557}
{False, True, 'cnchj', 654, 'trhjk', 6557}
{False, True, 'cnchj', 654, 'trhjk', 6557}
set()
{'brand': 'Ford', 'Model': 'Mustang', 'Color': 'Black'}
dict_values(['Ford', 'Mustang', 'Black'])
dict_keys(['brand', 'Model', 'Color'])
{'brand': 'Ford', 'Model': 'Mustang', 'Color': 'Black', 'year': 1969}
dict_values(['Ford', 'Mustang', 'Black', 1969])
dict_keys(['brand', 'Model', 'Color', 'year'])
{'brand': 'Ford', 'Model': 'Mustang'}
```

Operators

1. Use arithmetic, comparison, logical, and assignment operators.
2. Understand operator precedence.
3. Apply operators in expressions and calculations.

In [9]:
```python
'''1. Usinf arithmetic, comparison, logical and assignment operators'''
a = 32412
b = 9834.43
c = True
d = False
e = 5444.2
f = 23

arithmetic = {'sum' : a+b,
              'difference' : a-b,
```

```
              'product' : a*b,
              'quotient' : a/b,
              'remainder' : a%b,
              }
for i in arithmetic.keys():
    print('{} : {}'.format(i,arithmetic[i]))
print('\n')

comparison = { 'a=b' : a==b,
               'a!=b' : a!=b,
               'a<b' : a<b,
               'a>b' : a>b,
               'a>=b' : a>=b,
               'a<=b' : a<=b,
               }
for i in comparison.keys():
    print('{} : {}'.format(i,comparison[i]))
print('\n')

logical = { 'c and d' : c and d,
               'c or d' : c or d,
               'not c' : not c,
               'not d' : not d
               }
for i in logical.keys():
    print('{} : {}'.format(i,logical[i]))
print('\n')

print((a+b+e-f))                #Operator Precedence examples
print(((a+b*e)-f/e)/3)

if(a>b):
    if(a>e):
        if(a>f):
            print('{} is the largest'.format(a))
        else:
            print('{} is the largest'.format(f))
    elif(e>f):
        print('{} is the largest'.format(e))
    else:
        print('{} is the largest'.format(f))
elif(b>e):
    if(b>f):
        print('{} is the largest'.format(b))
    else:
        print('{} is the largest'.format(f))
elif(e>f):
    print('{} is the largest'.format(e))
else:
    print('{} is the largest'.format(f))
```

```
sum : 42246.43
difference : 22577.57
product : 318753545.16
quotient : 3.295768031294137
remainder : 2908.709999999999


a=b : False
a!=b : True
a<b : False
a>b : True
a>=b : True
a<=b : False


c and d : False
c or d : True
not c : False
not d : True


47667.63
17857671.933925107
32412 is the largest
```

Reading CSV files

1. Read CSV files into Pandas DataFrames.
2. Explore different CSV reading options and parameters.
3. Handle missing values and data cleaning.

In [60]:
```
import pandas as pd
reactionsDf = pd.read_csv(r"C:\Users\tejus\Downloads\Reactions.csv")
carsDf = pd.read_csv(r"C:\Users\tejus\Downloads\cars.csv", delimiter=';', header=0)
#print(reactionsDf)
#print(carsDf)
#reactionsDf.info()
#carsDf.info()
reactionsDf.head(10)        #reactionsDf contains Null values
#carsDf.head(10)             #carsDf has no Null values
reactionsDf = reactionsDf.dropna()        #removing all null rows
reactionsDf.head(10)
print(reactionsDf.columns)        #Getting name of first unnamed column
reactionsDf = reactionsDf.rename(columns={'Unnamed: 0': 'ID'})  #Renaming unknown c
reactionsDf.head(10)
```

```
Index(['Unnamed: 0', 'Content ID', 'User ID', 'Type', 'Datetime'], dtype='object')
```

Out[60]:

	ID	Content ID	User ID	Type	Datetime
1	1	97522e57-d9ab-4bd6-97bf-c24d952602d2	5d454588-283d-459d-915d-c48a2cb4c27f	disgust	2020-11-07 09:43:50
2	2	97522e57-d9ab-4bd6-97bf-c24d952602d2	92b87fa5-f271-43e0-af66-84fac21052e6	dislike	2021-06-17 12:22:51
3	3	97522e57-d9ab-4bd6-97bf-c24d952602d2	163daa38-8b77-48c9-9af6-37a6c1447ac2	scared	2021-04-18 05:13:58
4	4	97522e57-d9ab-4bd6-97bf-c24d952602d2	34e8add9-0206-47fd-a501-037b994650a2	disgust	2021-01-06 19:13:01
5	5	97522e57-d9ab-4bd6-97bf-c24d952602d2	9b6d35f9-5e15-4cd0-a8d7-b1f3340e02c4	interested	2020-08-23 12:25:58
6	6	97522e57-d9ab-4bd6-97bf-c24d952602d2	7918d465-0953-4f20-9e28-539e74c82e2f	peeking	2020-12-07 06:27:54
7	7	97522e57-d9ab-4bd6-97bf-c24d952602d2	fa3e42f0-71d8-455f-b024-e52d5c27a145	cherish	2021-04-11 17:35:49
8	8	97522e57-d9ab-4bd6-97bf-c24d952602d2	b0c22f82-b882-4394-bf27-6dfadf26e5c2	hate	2021-01-27 08:32:09
9	9	97522e57-d9ab-4bd6-97bf-c24d952602d2	1932a904-86ba-4438-bb52-b7e6516a4019	peeking	2021-04-01 22:54:23
11	11	97522e57-d9ab-4bd6-97bf-c24d952602d2	f50ac030-3af8-4e07-aacf-dccff353b8f6	indifferent	2020-11-07 08:36:27

Python String Methods

1. Manipulate strings using various built-in methods.
2. Perform operations like concatenation, slicing, finding substrings.
3. Convert strings to uppercase, lowercase, and title case.
4. Remove whitespace and split strings.

In [90]:

```
text = "My name is Tejus. I code on Python"

length = len(text)          #Get length of string
print(length)

replaced_text = text.replace("Python", "R")
print(replaced_text)        #Replacing a word in sentence

reversed_text = text[::-1]      #first index : last index : step
print(reversed_text)

str1 = "I do programming."

concatenated = text + ". " + str1
print(concatenated)    #concat two strings
```

```
sliced = concatenated[11:16]         # Get "Tejus"
print(sliced)

#substrings
index = concatenated.find("Python")
substring = concatenated[index:34]
print(substring)


uppercase = text.upper()            # Convert to uppercase
print(uppercase)

lowercase = text.lower()          # Convert to lowercase
print(lowercase)

title_case = text.title()         # Convert to title case
print(title_case)

string = "   Hello! How are YOU DOing??   "

cleaned_string = string.strip()             #remove all whitespaces
print(cleaned_string)

# Split string into a list of words
split_string = cleaned_string.split('e')
split_string1 = cleaned_string.split('?')
print(split_string)
print(split_string1)
```

```
34
My name is Tejus. I code on R
nohtyP no edoc I .sujeT si eman yM
My name is Tejus. I code on Python. I do programming.
Tejus
Python
MY NAME IS TEJUS. I CODE ON PYTHON
my name is tejus. i code on python
My Name Is Tejus. I Code On Python
Hello! How are YOU DOing??
['H', 'llo! How ar', ' YOU DOing??']
['Hello! How are YOU DOing', '', '']
```

NumPy

1. Create different types of NumPy arrays (1D, 2D, 3D).
2. Perform basic arithmetic operations on arrays.
3. Use indexing and slicing to access elements.
4. Explore array manipulation functions (reshape, transpose, concatenate).
5. Create and use NumPy random number generators.

In [ ]:
```
import numpy as np

# 1D array
array_1d = np.array([1, 2, 3, 4, 5])
print(array_1d)
```

```
# 2D array
array_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(array_2d)

# 3D array
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(array_3d)

print(array_1d.ndim)        #verify dimension of array
print(array_2d.ndim)
print(array_3d.ndim)
```

```
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
1
2
3
```

In [101…

```
a = np.array([[1, 2, 3],[4,6,4],[23,64,68],[76,43,11]])
b = np.array([32, 57, 69])

# Addition
sum = a + b
print(sum)
print('\n')

# Subtraction
difference = a - b
print(difference)
print('\n')

# Multiplication
product = a * b
print(product)
print('\n')

# Division
division = b / a
print(division)
```

```
[[ 33  59  72]
 [ 36  63  73]
 [ 55 121 137]
 [108 100  80]]


[[-31 -55 -66]
 [-28 -51 -65]
 [ -9   7  -1]
 [ 44 -14 -58]]


[[  32  114  207]
 [ 128  342  276]
 [ 736 3648 4692]
 [2432 2451  759]]


[[32.         28.5        23.        ]
 [ 8.          9.5        17.25      ]
 [ 1.39130435  0.890625    1.01470588]
 [ 0.42105263  1.3255814   6.27272727]]
```

In [105…

```
a = np.array([[1, 2, 3],[4,6,5],[23,64,68],[76,43,11]])

print(a)
# Access element (row 1, column 2)
element = a[0, 1]
print("Element at [1,2]:", element)

# Slice a subarray
subarray = a[0:2, 1:3]  # First two rows, last two columns
print("Sliced Subarray:\n", subarray)
```

```
[[ 1  2  3]
 [ 4  6  5]
 [23 64 68]
 [76 43 11]]
Element at [1,2]: 2
Sliced Subarray:
 [[2 3]
 [6 5]]
```

In [ ]:

```
# Reshaping
reshaped = a.reshape(3, 4)  # Convert 4x3 to 3x4
print("Reshaped Array:\n", reshaped)

#Transposing
transposed = a.T
print("Transposed Array:\n", transposed)

# Concatenating
array1 = np.array([[100, 110, 120]])
array2 = np.array([[100],[110],[120],[130], [150]])
concatenated = np.concatenate((a, array1), axis=0)          #concat across rows
```

```
concatenated = np.concatenate((concatenated, array2), axis=1)   #concat across colu
print("Concatenated Array:\n", concatenated)
```

```
Reshaped Array:
 [[ 1  2  3  4]
 [ 6  5 23 64]
 [68 76 43 11]]
Transposed Array:
 [[ 1  4 23 76]
 [ 2  6 64 43]
 [ 3  5 68 11]]
Concatenated Array:
 [[  1   2   3 100]
 [  4   6   5 110]
 [ 23  64  68 120]
 [ 76  43  11 130]
 [100 110 120 150]]
```

In [ ]:
```
# Generate random integers
rand_integers = np.random.randint(1, 10, size=(3, 3))
print("Random Integers:\n", rand_integers)

# Generate random floats between 0 and 1
rand_floats = np.random.rand(3, 3)
print("Random Floats:\n", rand_floats)

# Set a random seed for reproducibility
np.random.seed(42)
seeded_random = np.random.rand(3, 3)
print("Seeded Random Array:\n", seeded_random)
```

Pandas

1. Create Pandas Series and DataFrames.
2. Load data from various file formats (CSV, Excel, etc.).
3. Perform data cleaning and manipulation tasks.
4. Explore data analysis and visualization using Pandas.
5. Create pivot tables and group data for analysis.

In [125…
```
import pandas as pd

# Pandas Series
series = pd.Series([10, 20, 30, 40, 50], name="Numbers")
print(series)

# DataFrame from dictionary
data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "City": ["New York", "San Francisco", "Los Angeles"]
}
df = pd.DataFrame(data)
df.head()
```

```
0    10
1    20
2    30
3    40
4    50
Name: Numbers, dtype: int64
```

Out[125…

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles

In [143…

```
# Load a CSV file
csv_df = pd.read_csv(r'C:\Users\tejus\Downloads\Reactions.csv')
print(csv_df.head(5))

# Load an Excel file
excel_df = pd.read_excel(r'C:\Users\tejus\Downloads\SuperStoreUS-2015.xlsx')
print(excel_df.head(5))
```

```
   Unnamed: 0                            Content ID  \
0           0  97522e57-d9ab-4bd6-97bf-c24d952602d2
1           1  97522e57-d9ab-4bd6-97bf-c24d952602d2
2           2  97522e57-d9ab-4bd6-97bf-c24d952602d2
3           3  97522e57-d9ab-4bd6-97bf-c24d952602d2
4           4  97522e57-d9ab-4bd6-97bf-c24d952602d2

                                User ID     Type             Datetime
0                                   NaN      NaN  2021-04-22 15:17:15
1  5d454588-283d-459d-915d-c48a2cb4c27f  disgust  2020-11-07 09:43:50
2  92b87fa5-f271-43e0-af66-84fac21052e6  dislike  2021-06-17 12:22:51
3  163daa38-8b77-48c9-9af6-37a6c1447ac2   scared  2021-04-18 05:13:58
4  34e8add9-0206-47fd-a501-037b994650a2  disgust  2021-01-06 19:13:01
   Row ID Order Priority  Discount  Unit Price  Shipping Cost  Customer ID  \
0   20847           High      0.01        2.84           0.93            3
1   20228  Not Specified      0.02      500.98          26.00            5
2   21776       Critical      0.06        9.48           7.29           11
3   24844         Medium      0.09       78.69          19.99           14
4   24846         Medium      0.08        3.28           2.31           14

       Customer Name       Ship Mode Customer Segment Product Category  ...  \
0      Bonnie Potter     Express Air        Corporate  Office Supplies  ...
1     Ronnie Proctor  Delivery Truck      Home Office        Furniture  ...
2      Marcus Dunlap     Regular Air      Home Office        Furniture  ...
3  Gwendolyn F Tyson     Regular Air   Small Business        Furniture  ...
4  Gwendolyn F Tyson     Regular Air   Small Business  Office Supplies  ...

    Region State or Province         City  Postal Code Order Date  Ship Date  \
0     West        Washington    Anacortes        98221 2015-01-07 2015-01-08
1     West        California  San Gabriel        91776 2015-06-13 2015-06-15
2     East        New Jersey      Roselle         7203 2015-02-15 2015-02-17
3  Central         Minnesota   Prior Lake        55372 2015-05-12 2015-05-14
4  Central         Minnesota   Prior Lake        55372 2015-05-12 2015-05-13

      Profit Quantity ordered new    Sales Order ID
0     4.5600                    4    13.01    88522
1  4390.3665                   12  6362.85    90193
2   -53.8096                   22   211.15    90192
3   803.4705                   16  1164.45    86838
4   -24.0300                    7    22.23    86838

[5 rows x 25 columns]
```

In [ ]:
```
csv_df.head(10)        #reactions dataset contains Null values
csv_df = csv_df.dropna()        #removing all null rows
csv_df.head(10)
print(csv_df.columns)        #Getting name of first unnamed column
csv_df = csv_df.rename(columns={'Unnamed: 0': 'ID'})  #Renaming unknown column to I
csv_df.head(10)
```

```
Index(['Unnamed: 0', 'Content ID', 'User ID', 'Type', 'Datetime'], dtype='object')
```

Out[ ]:

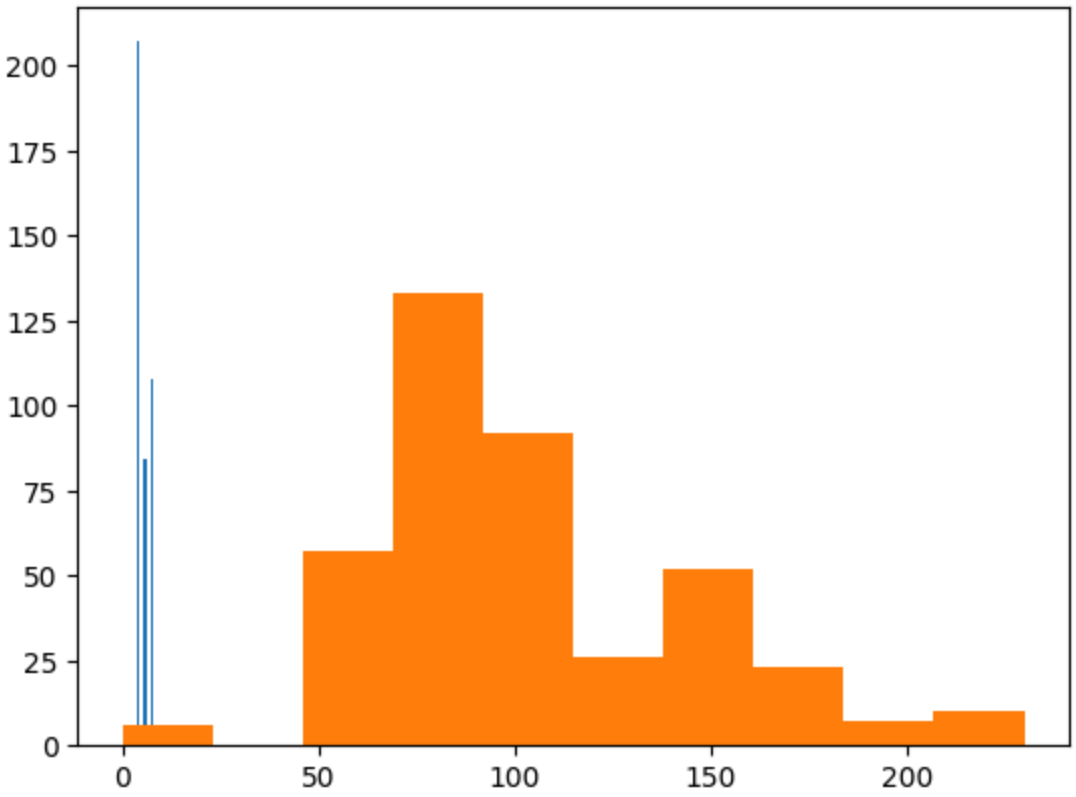
	ID	Content ID	User ID	Type	Datetime
1	1	97522e57-d9ab-4bd6-97bf-c24d952602d2	5d454588-283d-459d-915d-c48a2cb4c27f	disgust	2020-11-07 09:43:50
2	2	97522e57-d9ab-4bd6-97bf-c24d952602d2	92b87fa5-f271-43e0-af66-84fac21052e6	dislike	2021-06-17 12:22:51
3	3	97522e57-d9ab-4bd6-97bf-c24d952602d2	163daa38-8b77-48c9-9af6-37a6c1447ac2	scared	2021-04-18 05:13:58
4	4	97522e57-d9ab-4bd6-97bf-c24d952602d2	34e8add9-0206-47fd-a501-037b994650a2	disgust	2021-01-06 19:13:01
5	5	97522e57-d9ab-4bd6-97bf-c24d952602d2	9b6d35f9-5e15-4cd0-a8d7-b1f3340e02c4	interested	2020-08-23 12:25:58
6	6	97522e57-d9ab-4bd6-97bf-c24d952602d2	7918d465-0953-4f20-9e28-539e74c82e2f	peeking	2020-12-07 06:27:54
7	7	97522e57-d9ab-4bd6-97bf-c24d952602d2	fa3e42f0-71d8-455f-b024-e52d5c27a145	cherish	2021-04-11 17:35:49
8	8	97522e57-d9ab-4bd6-97bf-c24d952602d2	b0c22f82-b882-4394-bf27-6dfadf26e5c2	hate	2021-01-27 08:32:09
9	9	97522e57-d9ab-4bd6-97bf-c24d952602d2	1932a904-86ba-4438-bb52-b7e6516a4019	peeking	2021-04-01 22:54:23
11	11	97522e57-d9ab-4bd6-97bf-c24d952602d2	f50ac030-3af8-4e07-aacf-dccff353b8f6	indifferent	2020-11-07 08:36:27

In [ ]:

```
print(carsDf.head())    #list top 5 rows

print(carsDf.Cylinders.describe())
print(carsDf.MPG.describe())

import matplotlib.pyplot as plt

plt.hist(carsDf.Cylinders, label='Cylinders', bins=8 )
plt.hist(carsDf.Horsepower)
```

```
                          Car   MPG  Cylinders  Displacement  Horsepower  \
0  Chevrolet Chevelle Malibu  18.0          8         307.0       130.0
1          Buick Skylark 320  15.0          8         350.0       165.0
2         Plymouth Satellite  18.0          8         318.0       150.0
3              AMC Rebel SST  16.0          8         304.0       150.0
4                Ford Torino  17.0          8         302.0       140.0

   Weight  Acceleration  Model Origin
0  3504.0          12.0     70     US
1  3693.0          11.5     70     US
2  3436.0          11.0     70     US
3  3433.0          12.0     70     US
4  3449.0          10.5     70     US
count    406.000000
mean       5.475369
std        1.712160
min        3.000000
25%        4.000000
50%        4.000000
75%        8.000000
max        8.000000
Name: Cylinders, dtype: float64
count    406.000000
mean      23.051232
std        8.401777
min        0.000000
25%       17.000000
50%       22.350000
75%       29.000000
max       46.600000
Name: MPG, dtype: float64
```

Out[ ]:
```
(array([  6.,   0.,  57., 133.,  92.,  26.,  52.,  23.,   7.,  10.]),
 array([  0.,  23.,  46.,  69.,  92., 115., 138., 161., 184., 207., 230.]),
 <BarContainer object of 10 artists>)
```



In [222…

```
carsDf.head()
carDf_new = carsDf[['Cylinders','Displacement', 'Horsepower']]
print(carDf_new)
pivotTable = pd.pivot_table(carDf_new, values='Horsepower', aggfunc='mean', index=[
print('pivot_table')
print(pivotTable.head())

groupedData = carDf_new.groupby('Cylinders')['Horsepower'].mean()
print('grouping')
groupedData.head(5)
```

```
     Cylinders  Displacement  Horsepower
0            8         307.0       130.0
1            8         350.0       165.0
2            8         318.0       150.0
3            8         304.0       150.0
4            8         302.0       140.0
..         ...           ...         ...
401          4         140.0        86.0
402          4          97.0        52.0
403          4         135.0        84.0
404          4         120.0        79.0
405          4         119.0        82.0

[406 rows x 3 columns]
pivot_table
                         Horsepower
Cylinders Displacement
3         70.0           95.666667
          80.0          110.000000
4         68.0           49.000000
          71.0           65.000000
          72.0           69.000000
grouping
```

Out[222…

```
Cylinders
3     99.250000
4     76.574879
5     82.333333
6    100.297619
8    158.453704
Name: Horsepower, dtype: float64
```