Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 6

In [ ]:
```
Name :- Vora Yagnik
Enrollment No :- 23010101661
```

Tuples, dictionary, set

A

01) WAP to sort python dictionary by key or value.

In [27]:
```
d = {3: 'x', 5: 'a', 1: 'z', 4: 'b', 2: 'y'}
print("Original dictionary: ",d)

l = sorted(d.items(),key=lambda v: v[0])

print("Sorted by Keys : ",dict(l))

l = sorted(d.items(),key=lambda v: v[1])
print("Sorted by Keys : ",dict(l))
```

```
Original dictionary:  {3: 'x', 5: 'a', 1: 'z', 4: 'b', 2: 'y'}
Sorted by Keys :  {1: 'z', 2: 'y', 3: 'x', 4: 'b', 5: 'a'}
Sorted by Keys :  {5: 'a', 4: 'b', 3: 'x', 2: 'y', 1: 'z'}
```

02) WAP to merge two dictionaries given by user.

In [35]:
```
a = {'a': 1, 'b': 2, 'c': 3}
b = {'d': 4, 'e': 5, 'f': 6}

print("First dictionary : ",a)
print("Second dictionary : ",b)
a.update(b)
print("Merged dictionary : ",a)
```

```
First dictionary :  {'a': 1, 'b': 2, 'c': 3}
Second dictionary :  {'d': 4, 'e': 5, 'f': 6}
Merged dictionary :  {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
```

03) WAP to find tuples that have all elements divisible by K from a list of tuples.

In [5]:
```
a = [(1,2,3),(3,6,9),(9,8,7),(12,15,18),(8,5,2)]

b = [i for i in a if all(y%3 ==0 for y in i)]

print(b)
```

```
[(3, 6, 9), (12, 15, 18)]
```

04) WAP to find Tuples with positive elements in List of tuples.

In [7]:
```
a = [(1,2,-3),(3,6,9),(-9,8,-7),(12,15,18),(-8,5,2)]

b = [i for i in a if all(y > 0 for y in i)]

print(b)
```

```
[(3, 6, 9), (12, 15, 18)]
```

05) WAP which perform union of two sets.

In [16]:
```
a = {1,2,3,'a','b','c'}
b = {3,4,5,'c','d','e'}

c = a|b

print(c)
```

```
{1, 2, 3, 'd', 4, 5, 'e', 'a', 'b', 'c'}
```

B

01) WAP to convert binary tuple into integer.

In [20]:
```
a = (1,1,1,1,1)

int_value = int(''.join(map(str, a)), 2)

print(int_value)
```

```
31
```

02) WAP to count frequency in list by dictionary.

In [30]:
```
a = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
d = {}

for i in a:
    d[i] = 0

for i in a:
    d[i]+=1

print(d)
```

```
{1: 2, 2: 2, 3: 2, 4: 2, 5: 2}
```

03) WAP to remove all the duplicate words from the list using dictionary.

In [31]:
```
a = ['red', 'orange', 'yellow', 'green', 'blue', 'violet', 'red', 'orange', 'yellow
b = []
d = {}
for i in a:
    if i not in b:
        b.append(i)
    d[i] = a.count(i)

print(b)
print(d)
```

```
['red', 'orange', 'yellow', 'green', 'blue', 'violet']
{'red': 2, 'orange': 2, 'yellow': 2, 'green': 2, 'blue': 2, 'violet': 2}
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 7

Functions

```
In [ ]: Name :- Vora Yagnik
        Enrollment No :- 23010101661
```

01) WAP to count simple interest using function.

SI =(principle(amount) * rate of interest * time)/ 100

```
In [3]: def interest(p,r,n):
            return (p*r*n)/100

        interest(100000,2,2)
```

```
Out[3]: 4000.0
```

02) WAP that defines a function to add first n numbers.

```
In [7]: def sumN(n):
            sum = 0
            for i in range(1,n+1):
                sum += i
            return sum

        n = int(input("Enter N : "))
        print("Sum of first ",n," = ",sumN(n))
```

```
Sum of first  10  =  55
```

03) WAP to find maximum number from given two numbers using function.

In [12]:
```
def maxOfTwo(a,b):
    return a if a>b else b

a = int(input("Enter first number : "))
b = int(input("Enter second number : "))

print("Max from",a,"and",b,"is",maxOfTwo(a,b))
```

```
Max from 10 and 20 is 20
```

04) WAP that defines a function which returns 1 if the number is prime otherwise return 0.

In [43]:
```
def isPrime(n):
    c = 0
    for i in range(2,int(n**0.5)):
        c += 1
        if n%i == 0:
            return 0,c
    return 1,c
n = int(input("Enter number : "))
prime,iteration = isPrime(n)
print("No. of iterations : ",iteration,"\n",prime)
```

```
No. of iterations :  4
 0
```

In [1]:
```
# Example on how iterations are calculated
# Only iteration in loops are counted
iteration_count= 0
n = 5
for i in range(n):
    iteration_count += 1

    for j in range(n-i-1):
        iteration_count += 1
        print(" ", end = " ")
    for j in range(i+1):
        iteration_count += 1
        print("*", end = " ")
    print()
print("Total Iterations : ", iteration_count)
```

```
        * 
      * * 
    * * * 
  * * * * 
* * * * * 
Total Iterations :  30
```

In [20]:
```

```

```
Enter Number :  9973
No. of iteration :  16
1
```

05) Write a function called primes that takes an integer value as an argument and returns a list of all prime numbers up to that number.

In [83]:
```python
def primes(n):
    p = []
    t,c = 0,0
    for i in range(2,n+1):
        t += c
        c = 0
        f = True
        for j in range (2,i):
            c += 1
            if i%j == 0:
                f = False
        if f==True:
            p.append(i)
    return p,t
i,c = primes(1000)
print(c)
```

```
497503
```

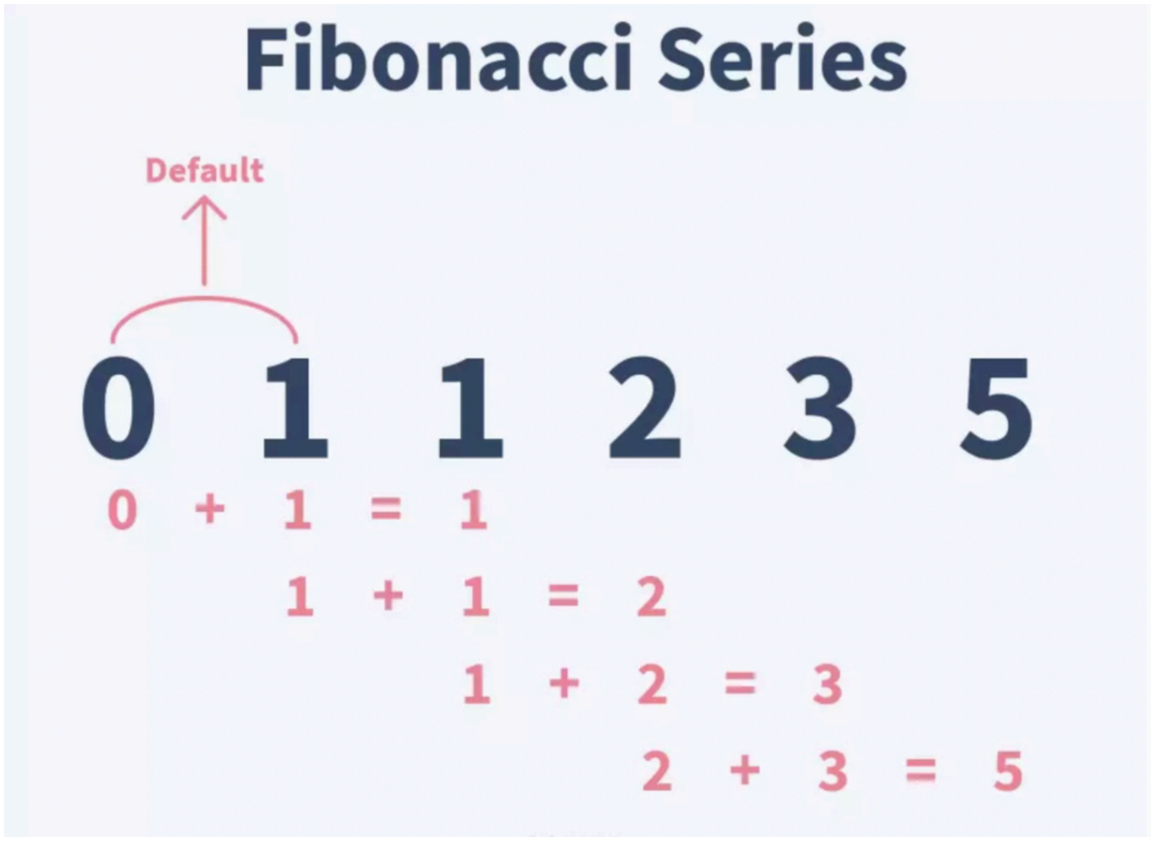
In [93]:
```python
c = 0
def primes(n):
    global c
    for i in range(2,n):
        for j in range (2,int(i**0.5)):
            c += 1
            if i%j == 0:
                break
        else:
            print(i,end=" ")
print(primes(1000))
print(c)
```

```
2 3 4 5 6 7 8 9 11 13 15 17 19 23 25 29 31 35 37 41 43 47 49 53 59 61 67 71 73 79 83
89 97 101 103 107 109 113 121 127 131 137 139 143 149 151 157 163 167 169 173 179 18
1 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 289 29
3 307 311 313 317 323 331 337 347 349 353 359 361 367 373 379 383 389 397 401 409 41
9 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 529 541 54
7 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 67
3 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 82
3 827 829 839 841 853 857 859 863 877 881 883 887 899 907 911 919 929 937 941 947 95
3 961 967 971 977 983 991 997 None
5103
```

optional task : optimization

if basic task completed then try to optimize

06) WAP to generate Fibonacci series of N given number using function name fibbo. (e.g. 0 1 1 2 3 5 8...)



In [10]:
```
def fibo(n):
    a = 0
    b = 1
    print(a,b,end=",",sep=",")
    for i in range(2,n):
        c = a+b
        print(c,end=",")
        a = b
        b = c

n = int(input("Enter no of Steps : "))
fibo(n)
```

```
0,1,1,2,3,5,8,13,21,34,
```

07) WAP to find the factorial of a given number using recursion.

In [19]:
```
def fact(n):
    if n == 0:
        return 1
    else:
        return n*fact(n-1)
```

```
n = int(input("Enter number to find factorial : "))
fact(n)
```

```
Out[19]:  5040
```

08) WAP to implement simple calculator using lamda function.

```
In [25]:
a = int(input("Enter first no :- "))
b = int(input("Enter second no :- "))
c = input("Enter Operation [+,-,*,/] :- ")

if c == '+':
    print("Ans :- ",(lambda a,b:a+b)(a,b))
elif c == '-':
    print("Ans :- ",(lambda a,b:a-b)(a,b))
elif c == '*':
    print("Ans :- ",(lambda a,b:a*b)(a,b))
elif c == '/':
    print("Ans :- ",(lambda a,b:a/b)(a,b))
```

```
Ans :-  10
```

09)Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically

Sample Items : green-red-yellow-black-white

Expected Result : black-green-red-white-yellow

```
In [31]:
s = "green-red-yellow-black-white"
l = s.split("-")
l = sorted(l)
print("-".join(l))
```

```
black-green-red-white-yellow
```

01) WAP to calculate power of a number using recursion.

```
In [34]:
def powerD(base,power):
    if power == 0:
        return 1
    else:
        return base*powerD(base,power-1)

powerD(3,3)
```

```
Out[34]:  27
```

02) WAP to count digits of a number using recursion.

```
In [42]: def countD(n):
             if n//10 == 0:
                 return 1
             else:
                 return 1+countD(n//10)
         countD(1418150)
```

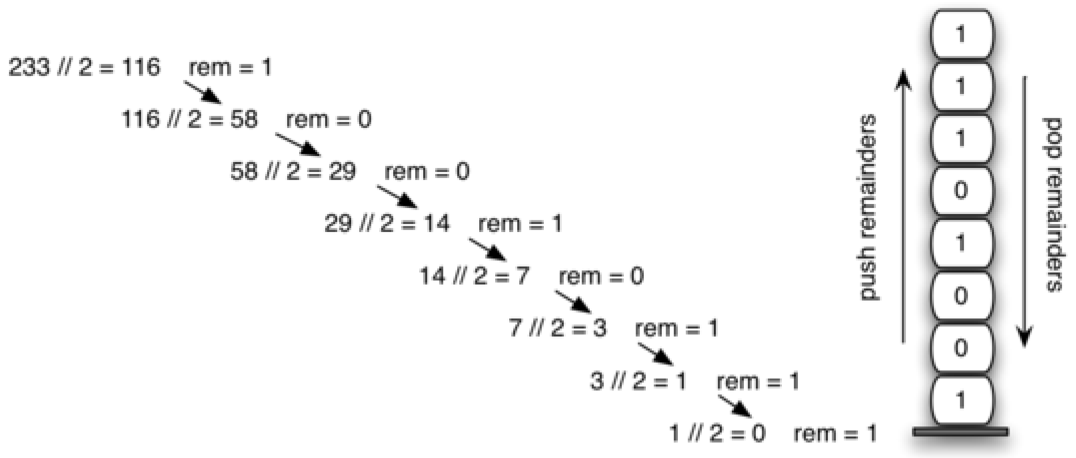
```
Out[42]: 7
```

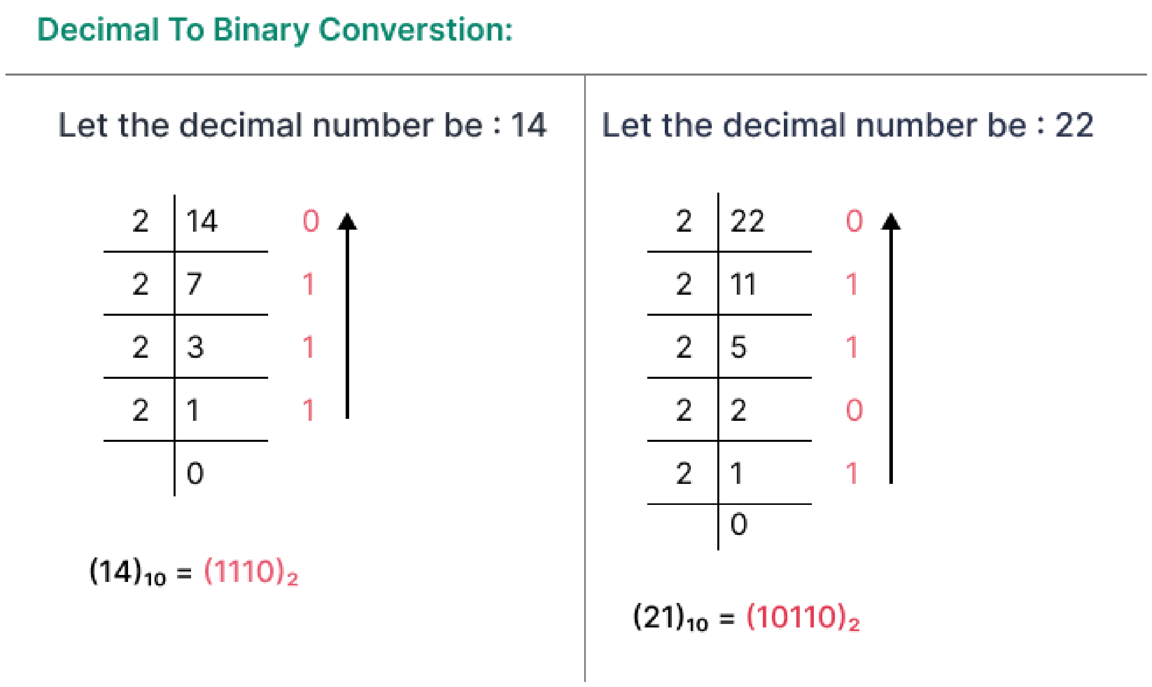
03) WAP to reverse an integer number using recursion.

```
In [44]: def reverseN(n,rev_n=0):
             if n ==0:
                 return rev_n
             else:
                 lDigit = n%10
                 rev_n = rev_n*10 + lDigit
                 return reverseN(n//10,rev_n)

         reverseN(2151515)
```

```
Out[44]: 5151512
```

04) WAP to convert decimal number into binary using recursion.




Decimal To Binary Converstion:
Let the decimal number be : 14
Let the decimal number be : 22
(14)10 = (1110)2
(21)10 = (10110)2

```
In [55]: def dtob(n):
             str = ""
             if n == 0:
                 return 0
             else:
                 return (n%2 + 10*dtob(int(n//2)))
         dtob(10)
```

```
Out[55]: 1010
```

Map , Filter , Reduce

map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

```
In [3]: numbers = [1, 2, 3, 4, 5]

        squared_numbers = list(map(lambda x: x * x, numbers))

        print(squared_numbers)
```

```
[1, 4, 9, 16, 25]
```

The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

```
In [1]: seq = [0, 1, 2, 3, 5, 8, 13]
```

```python
result = filter(lambda x: x % 2 != 0, seq)
print(list(result))

result = filter(lambda x: x % 2 == 0, seq)
print(list(result))
```

```
[1, 3, 5, 13]
[0, 2, 8]
```

The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

In [2]:
```python
import functools

lis = [1, 3, 5, 6, 2]

print("The sum of the list elements is : ", end="")
print(functools.reduce(lambda a, b: a+b, lis))

print("The maximum element of the list is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b, lis))
```

```
The sum of the list elements is : 17
The maximum element of the list is : 6
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 8

File handling

In [ ]:
```
Name :- VORA YAGNIK
Enrollment no :- 23010101661
```

A

01) WAP to read entire file named abc.txt

In [2]:
```python
file = open("abc.txt","r")
print(file.read())
```

```
Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type
specimen book. It has survived not only five centuries, but also the leap into
electronic typesetting, remaining essentially unchanged. It was popularised in
the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,
and more recently with desktop publishing software like Aldus PageMaker including
versions of Lorem Ipsum.Lorem Ipsum is simply dummy text of the printing and
typesetting industry. Lorem Ipsum has been the industry's standard dummy text
ever since the 1500s, when an unknown printer took a galley of type and scrambled
it to make a type specimen book. It has survived not only five centuries, but also
the leap into electronic typesetting, remaining essentially unchanged. It was
popularised in the 1960s with the release of Letraset sheets containing Lorem
Ipsum passages, and more recently with desktop publishing software like Aldus
PageMaker including versions of Lorem Ipsum.Lorem Ipsum is simply dummy text
of the printing and typesetting industry. Lorem Ipsum has been the industry's
standard dummy text ever since the 1500s, when an unknown printer took a galley
of type and scrambled it to make a type specimen book. It has survived not only
five centuries, but also the leap into electronic typesetting, remaining
essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with desktop
publishing software like Aldus PageMaker including versions of Lorem Ipsum.
```

02) WAP to print program it self on console.

In [3]:
```python
file = open("demo.py","r")
print(file.read())
```

```
def demo(a):
    print(a)

demo("Hello World");
```

03) WAP to read first 5 lines from the file named abc.txt

In [4]:
```python
file = open("abc.txt","r")
for _ in range(5):
    c = file.readline()
    if c:
        print(c)
    else:
        break
```

```
Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

when an unknown printer took a galley of type and scrambled it to make a type

specimen book. It has survived not only five centuries, but also the leap into

electronic typesetting, remaining essentially unchanged. It was popularised in
```

04) WAP to find the longest word in a file named abc.txt

```
In [5]:
file = open("abc.txt","r")
words = file.read().split()
print(max(words,key=len))
```

```
typesetting,
```

05) WAP to find the size of the file named abc.txt

```
In [10]:
file = open("abc.txt","r")
file.seek(0)
print("Size of file is :",len(file.read()))
```

```
Size of file is : 1729
```

06) WAP to implement search function to search specific occurance of word in a given text file.

```
In [20]:
file = open("abc.txt","r")
file.seek(0)
list = file.read().lower().split(" ")
word = input("Enter a Word to search : ")
word = word.lower()
wc = 0
wc = list.count(word)
print(wc)
```

```
8
```

B

01) WAP to write first 100 prime numbers to a file named primenumbers.txt

(Note: each number should be in new line)

```
In [31]:
file = open("primenumbers.txt","w")
for i in range(1,100):
    flag = False
    for j in range(2,i):
        if(i%j != 0):
            flag = True
    if(flag):
        file.write(str(i)+"\n")
```

02) WAP to merge two files and write it in a new file.

In [12]:
```
f1 = open("abc.txt","r")
f2 = open("xyz.txt","r")
f3 = open("pqr.txt","w+")

f3.writelines(f1.readlines())
f3.writelines(f2.readlines())
```

03) WAP to encrypt a text file.

In [7]:
```
import string
s = string.ascii_letters
b = "URXjqYOguKkbANeSiZECMQHPIvGzncBVpmwDhadTsxLWJfrloytF"

def encode(word):
    x = ""
    for i in word:
        temp = s.find(i)
        x += b[temp]
    return x
text = input("Enter a string to encrypt : ")
print(encode(text))
```

```
IUONukFQeZU
```

04) WAP to decrypt a previously encrypted file.

In [10]:
```
import string
s = string.ascii_letters
b = "URXjqYOguKkbANeSiZECMQHPIvGzncBVpmwDhadTsxLWJfrloytF"

def decode(word):
    x = ""
    for i in word:
        temp = b.find(i)
        x += s[temp]
    return x

text = input("Enter a string to decrypt : ")
print(decode(text))
```

```
yagnikZvora
```

05) WAP to remove a word from text file.

In [43]:
```
import shutil
word = input("Enter Word remove : ")
file = open("abc.txt","r+")
new_file = open("temp.txt","w+")
for line in file:
    new_line = line.replace(word,"")
    new_file.write(new_line)

shutil.copyfile("temp.txt","abc.txt")
```

```
Out[43]:  'abc.txt'
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 9

Exception Handling

```
In [ ]: Name : VORA YAGNIK
        Enrollment no: 23010101661
```

A

01) WAP to handle divide by zero exception.

```
In [1]: try:
            a = 5/0
        except ZeroDivisionError:
            print("Divide By Zero Error is occured")
```

```
Divide By Zero Error is occured
```

02) Write a Python program that inputs a number and generates an error message if it is not a number.

```
In [3]: try:
            a = int(input("Enter a number : "))
        except ValueError:
            print("Please Enter a Number")
```

```
Please Enter a Number
```

03) WAP to handle file not found Exception

```
In [6]: try:
            file = open("demo.txt","r")
        except FileNotFoundError:
            print("Please open existing file")
```

```
Please open existing file
```

04) WAP to handle type Exception.

```
In [7]: try:
            a = "a" + 5
        except TypeError:
            print("Type Error is Occured")
```

```
Type Error is Occured
```

05) WAP to demonstrate valueError and indexError with example.

```
In [8]: def valueErrorDemo():
            try:
                a = int("a")
            except ValueError:
                print("ValueError is occured")

        def indexErrorDemo():
            try:
                a = [1,2,3]
                b = a[5]
            except IndexError:
                print("IndexError is occured")

        valueErrorDemo()
        indexErrorDemo()
```

```
ValueError is occured
IndexError is occured
```

06) WAP to domonstrate else and finally block.

```
In [13]: def demo(a, b):
             try:
                 result = a / b
             except ZeroDivisionError:
                 print("Division by zero")
             else:
                 print("else :::: Division successful. Result:", result)
             finally:
                 print("finally :::: Finally block executed")

         demo(10, 2)
         demo(10, 0)
```

```
else :::: Division successful. Result: 5.0
finally :::: Finally block executed
Error: Division by zero
finally :::: Finally block executed
```

07) Create a short program that prompts the user for a list of grades separated by commas. Split the string into individual grades and use a list comprehension to convert each string to an integer. You should use a try statement to inform the user when the values they entered cannot be converted.

In [23]:
```
try:
    grades = input("Enter the list of grades separated by commas: ")
    grade_list = grades.split(',')
    gradesToInt = [int(grade) for grade in grade_list]
    print("Grades:", grades)
except ValueError:
    print("Please Enter Valid format")
```

```
Please Enter Valid format
```

B

01) WAP to Raising User Generated Exception.

In [29]:
```
class MyException(Exception):
    def __init__(self,arg):
        self.arg = arg

try:
    a = int(input("Enter a positive number"))
    if a < 0:
        raise MyException("Enter positive number")
except MyException as e:
    print(e.arg)
```

```
Enter positive number
```

02) WAP to raise your custom Exception.

In [31]:
```
class MyException(Exception):
    def __init__(self,arg):
        self.arg = arg

try:
    a = int(input("Enter a positive number"))
    if a%2 == 0:
        raise MyException("Enter number is even")
    else:
        raise MyException("Entered number is odd")
```

```python
except MyException as e:
    print(e.arg)
```

```
Entered number is odd
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 10

```
In [ ]: Name :- Vora Yagnik
        Enrollment No :- 23010101661
```

Modules

A

01) WAP to create Calculator module which defines functions like add, sub,mul and div. create another file that uses the Calculator module.

```
In [2]: import Calculator as cl

        a = int(input("Enter First number : "))
        b = int(input("Enter Second number : "))
        op = input("Enter operation : ")

        cl.calc(a,b,op)
```

```
Enter First number :  20
Enter Second number :  10
Enter operation :  -
```

```
Out[2]: 10
```

02) WAP to Pick a random character from a given String.

```
In [14]: import random as Random
         str = "yagnik"
         # str = input("Enter a string : ")
```

```
r = Random.random()
print(str[int(r*10)%len(str)])
```

```
n
```

03) WAP to Pick a random element from a given list.

In [30]:
```
import random as Random
list = [1,2,3,4,5,6,7,8,9,10]
r = Random.random()
print(list[int(r*10)%len(list)])
```

```
4
```

04) WAP to demonstrate the use of the math module.

In [3]:
```
import math
print("Value of pi: ",math.pi)
print("Value of e: ",math.e)
print("Square root of 10: ",math.sqrt(10))
print("Sine of 30 degrees: ",math.sin(math.radians(30)))
print("Cosine of 45 degrees: ",math.cos(math.radians(45)))
print("Tangent of 60 degrees: ",math.tan(math.radians(60)))
print("Floor of 2.5: ",math.floor(2.5))
print("Ceiling of 3.2: ",math.ceil(3.2))
print("Factorial of 5: ",math.factorial(5))
print("Absolute value of -5: ",math.fabs(-5))
print("2 raised to the power of 3: ",math.pow(2,3))
print("Logarithm of 10 to the base 2: ",math.log(10,2))
print("Natural logarithm of 10 (log of 10 to the base e): ",math.log(10))
print("Hyperbolic sine of 1: ",math.sinh(1))
print("Hyperbolic cosine of 1: ",math.cosh(1))
print("Hyperbolic tangent of 1: ",math.tanh(1))
print("Inverse hyperbolic sine of 1: ",math.asinh(1))
print("Inverse hyperbolic cosine of 1: ",math.acosh(1))
print("Inverse hyperbolic tangent of 0: ",math.atanh(0))
```

```
Value of pi:  3.141592653589793
Value of e:  2.718281828459045
Square root of 10:  3.1622776601683795
Sine of 30 degrees:  0.49999999999999994
Cosine of 45 degrees:  0.7071067811865476
Tangent of 60 degrees:  1.7320508075688767
Floor of 2.5:  2
Ceiling of 3.2:  4
Factorial of 5:  120
Absolute value of -5:  5.0
2 raised to the power of 3:  8.0
Logarithm of 10 to the base 2:  3.3219280948873626
Natural logarithm of 10 (log of 10 to the base e):  2.302585092994046
Hyperbolic sine of 1:  1.1752011936438014
Hyperbolic cosine of 1:  1.5430806348152437
Hyperbolic tangent of 1:  0.7615941559557649
Inverse hyperbolic sine of 1:  0.881373587019543
Inverse hyperbolic cosine of 1:  0.0
Inverse hyperbolic tangent of 0:  0.0
```

05) WAP to demonstrate the use of date time module.

In [8]:
```
import datetime as dt
print("Current date and time: ",dt.datetime.now())
print("Current date: ",dt.date.today())
print("Current time: ",dt.datetime.now())
print("Current year: ",dt.date.today().year)
print("Current month: ",dt.date.today().month)
print("Current day: ",dt.date.today().day)
print("Current hour: ",dt.datetime.now().hour)
print("Current minute: ",dt.datetime.now().minute)
print("Current second: ",dt.datetime.now().second)
print("Current microsecond: ",dt.datetime.now().microsecond)
```

```
Current date and time:  2024-04-10 21:06:26.454087
Current date:  2024-04-10
Current time:  2024-04-10 21:06:26.458084
Current year:  2024
Current month:  4
Current day:  10
Current hour:  21
Current minute:  6
Current second:  26
Current microsecond:  460095
```

B

01) WAP to Roll dice in such a way that every time you get the same number.

In [19]:
```
import random
```

```
random.seed(0)
result = random.randint(1, 6)
print("Rolling the dice...")
print("Dice result:",result)
```

```
Rolling the dice...
Dice result: 4
```

02) WAP to generate 3 random integers between 100 and 999 which is divisible by 5.

In [36]:
```
import random

for i in range (1,4):
    print(i,"- Number :-",random.randint(20, 199) * 5 )
```

```
1 - Number :- 115
2 - Number :- 215
3 - Number :- 610
```

03) WAP to generate 100 random lottery tickets and pick two lucky tickets from it as a winner.

In [41]:
```
import random

lottery_ticket = [random.randint(10000,99999) for _ in range(100)]

lucky_ticket = random.sample(lottery_ticket, 2)
print("Tickets :- ",lottery_ticket)
print("Lucky tickets :- ",lucky_ticket)
```

```
Tickets :-  [43617, 37162, 94016, 15518, 38297, 91712, 29182, 23714, 35948, 70084, 5
9549, 57401, 81616, 29835, 23729, 88150, 73950, 29451, 83912, 63203, 93671, 99119, 6
5474, 78318, 74933, 99039, 52265, 75331, 75358, 93239, 97827, 36475, 81147, 89923, 3
8675, 11272, 54593, 51714, 52182, 14648, 78826, 29446, 43670, 88982, 30434, 59676, 8
6408, 48585, 71660, 18697, 21094, 77702, 15166, 18697, 39503, 27104, 15325, 49380, 1
2003, 68797, 53331, 31061, 29506, 95994, 70389, 58669, 76191, 60099, 79442, 75854, 1
4406, 85218, 21881, 98941, 77895, 88661, 20008, 65884, 37014, 47966, 80178, 88461, 6
4767, 73204, 60930, 89617, 86859, 40614, 12684, 96112, 10031, 33851, 49641, 76434, 8
4745, 43352, 53605, 18601, 74685, 44341]
Lucky tickets :-  [52182, 74685]
```

04) WAP to print current date and time in Python.

In [43]:
```
import datetime as dt

print("Current Date And Time :-", dt.datetime.now())
```

```
Current Date And Time :- 2024-04-10 21:40:56.513469
```

05) Subtract a week (7 days) from a given date in Python.

In [52]:
```python
from datetime import datetime, timedelta

date = '20-03-2024'
date_obj = datetime.strptime(date, '%d-%m-%Y')
n_date_obj = date_obj - timedelta(days=7)
new_date = n_date_obj.strftime('%d-%m-%Y')
print("Given date:", date)
print("New date:", new_date)
```

```
Given date: 20-03-2024
New date: 13-03-2024
```

06) WAP to Calculate number of days between two given dates.

In [64]:
```python
import datetime as dt

start = '15-03-2023'
end = '22-03-2023'

start_date = dt.datetime.strptime(start, '%d-%m-%Y')
end_date = dt.datetime.strptime(end, '%d-%m-%Y')
delta = end_date - start_date
num_days = delta.days
print("Number of days between", start, "and", end, ":", num_days)
```

```
Number of days between 15-03-2023 and 22-03-2023 : 7
```

07) WAP to Find the day of the week of a given date.

In [76]:
```python
import datetime as dt

date = '22-03-2023'
date_obj = dt.datetime.strptime(date,'%d-%m-%Y')
day = date_obj.weekday()
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday
day = days[day]
print("Given date:", date)
print("Day of the week:", day)
```

```
Given date: 22-03-2023
Day of the week: Wednesday
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 11

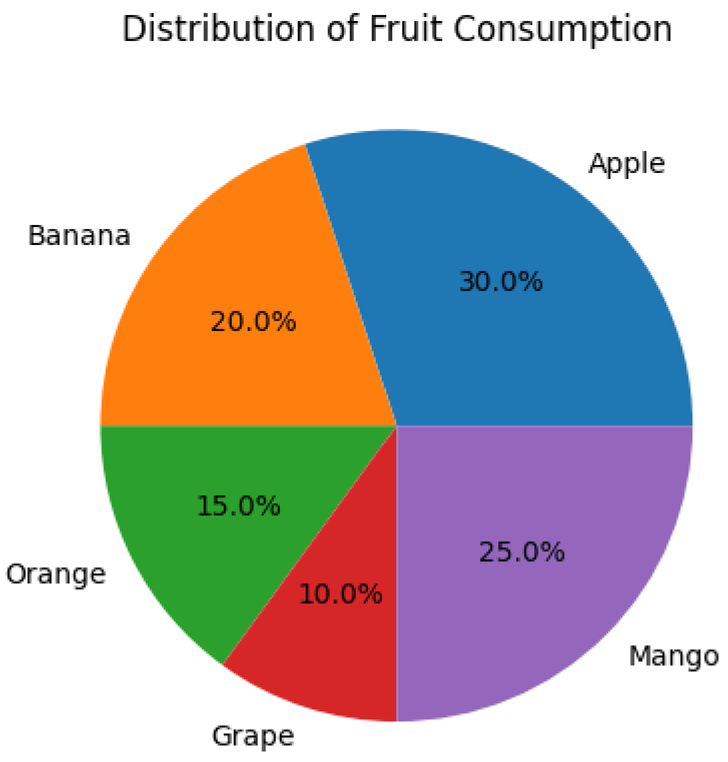
```
In [ ]: Name :- Vora Yagnik
        Enrollment No :- 23010101661
```

Graphs

A

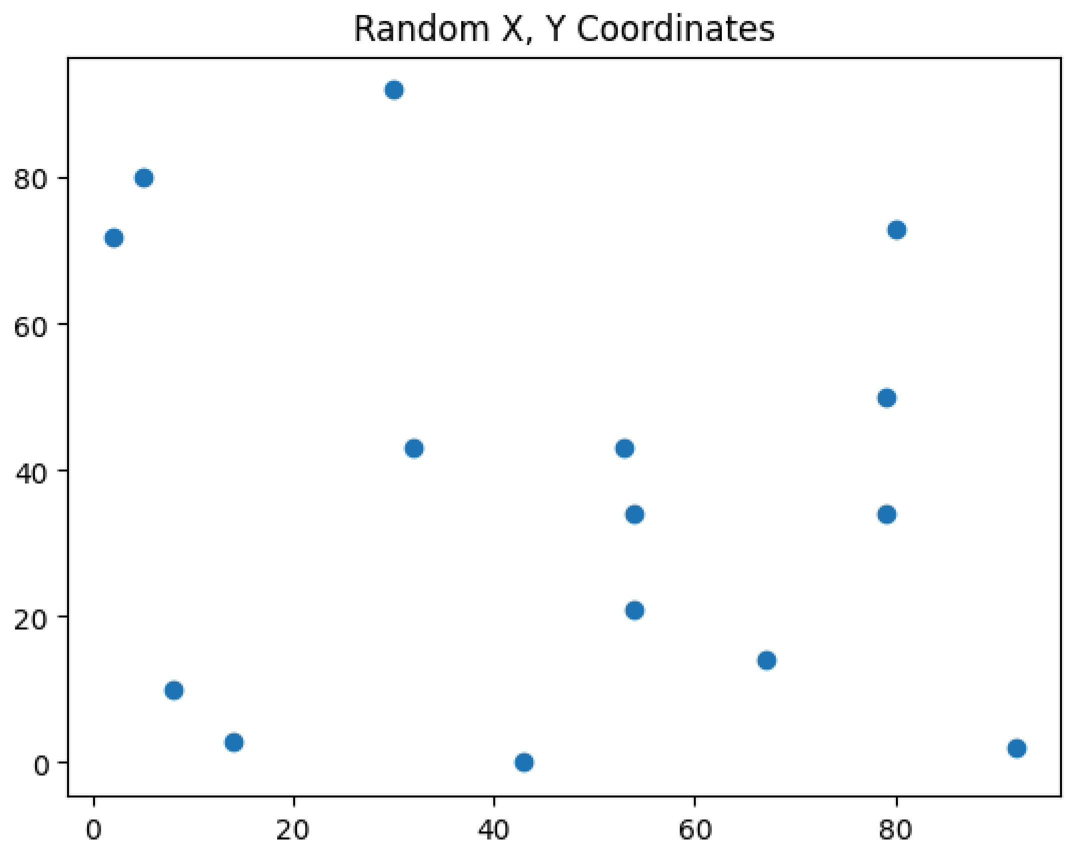
01) WAP to demonstrate the use of Pie chart.

```
In [6]: import matplotlib.pyplot as plt

        labels = ['Apple', 'Banana', 'Orange', 'Grape', 'Mango']
        sizes = [30, 20, 15, 10, 25]
        plt.pie(sizes, labels=labels, autopct='%1.1f%%')
        plt.title('Distribution of Fruit Consumption')
        plt.show()
```



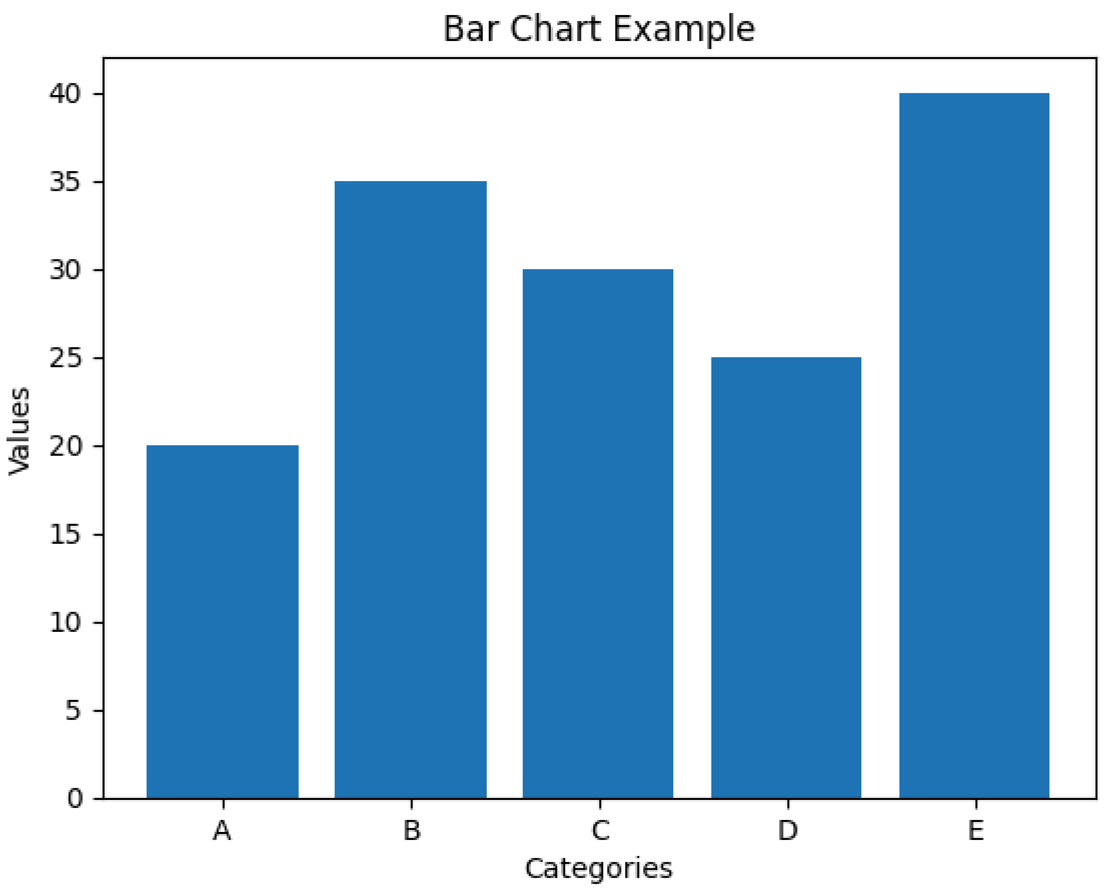
02) WAP to to Plot List random of X, Y Coordinates in Matplotlib.

In [19]:
```python
import matplotlib.pyplot as plt
import random

x = [random.randint(0, 100) for i in range(15)]
y = [random.randint(0, 100) for i in range(15)]
plt.scatter(x, y)
plt.title('Random X, Y Coordinates')
plt.show()
```



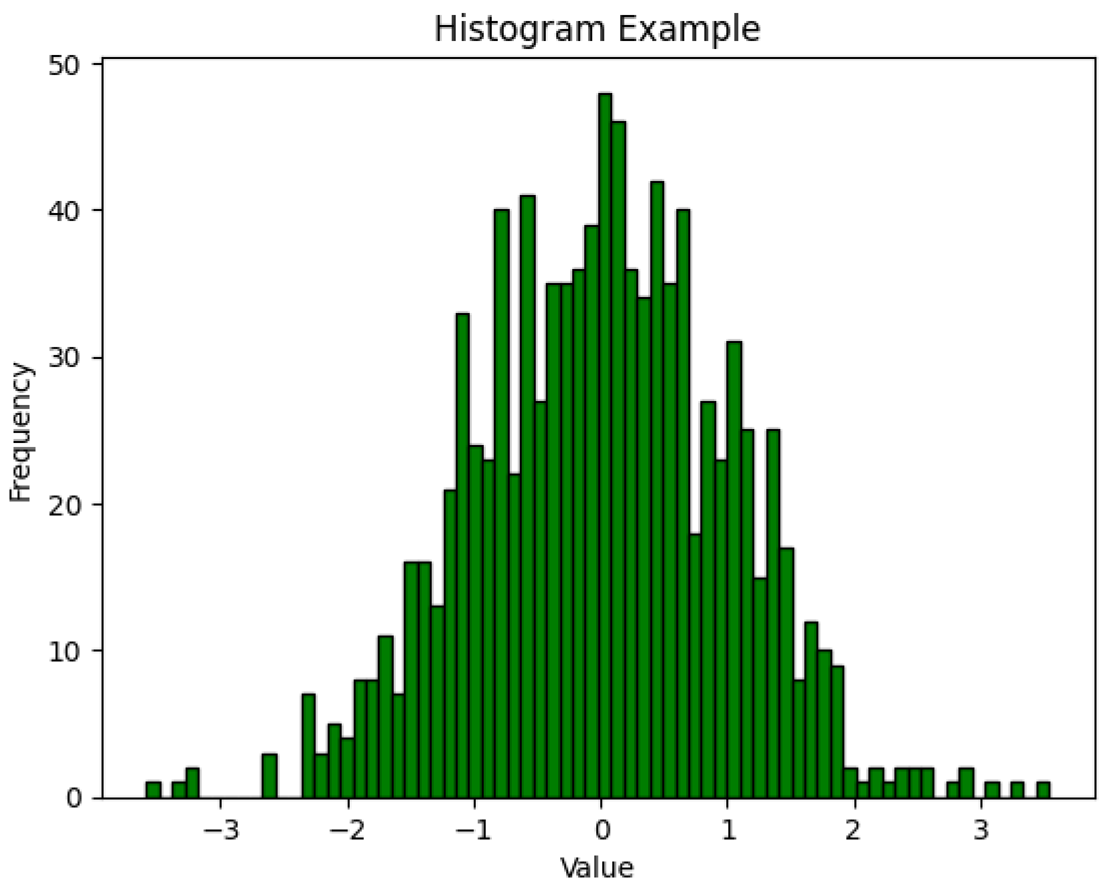
03) WAP to demonstrate the use of Bar chart.

In [23]:
```python
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D', 'E']
values = [20, 35, 30, 25, 40]
plt.bar(categories, values)
plt.title('Bar Chart Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```



04) WAP to demonstrate the use of Histogram.

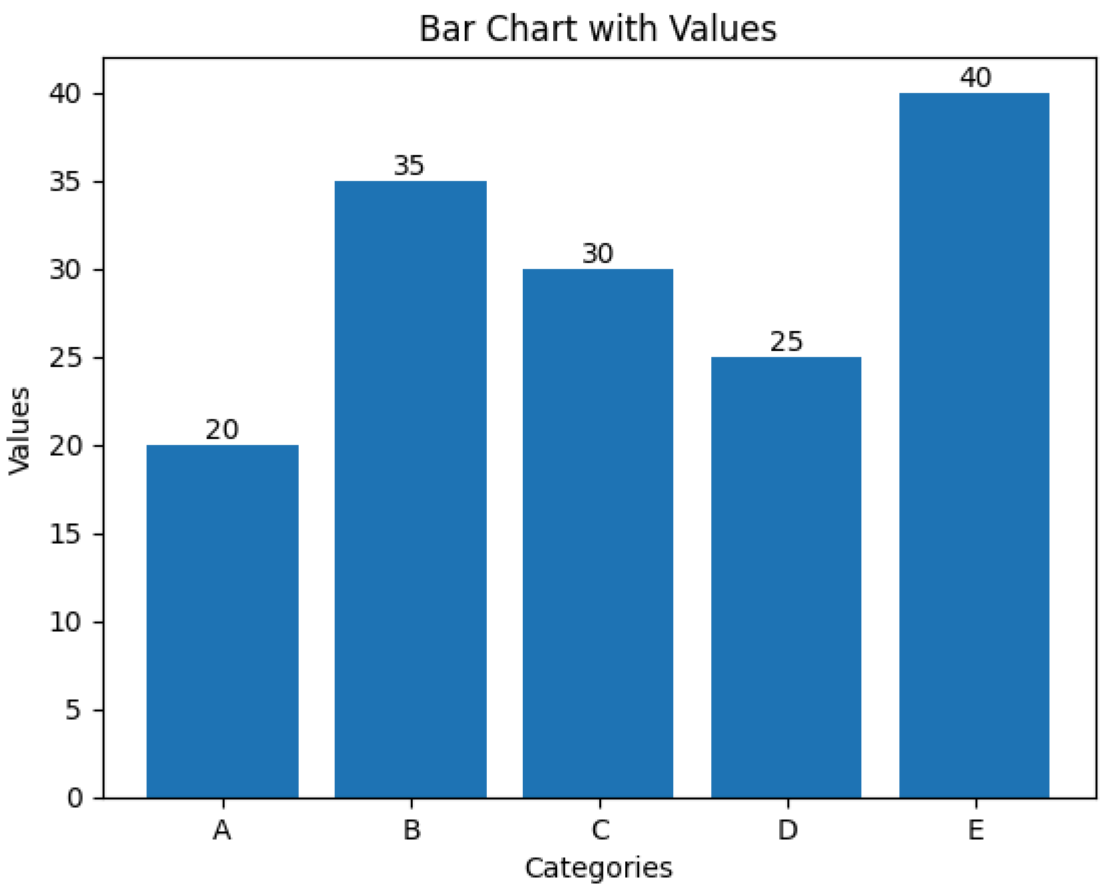
In [36]:
```
import matplotlib.pyplot as plt
import numpy as np

data = np.random.randn(1000)
plt.hist(data, bins=70, color='green', edgecolor='black')
plt.title('Histogram Example')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



B

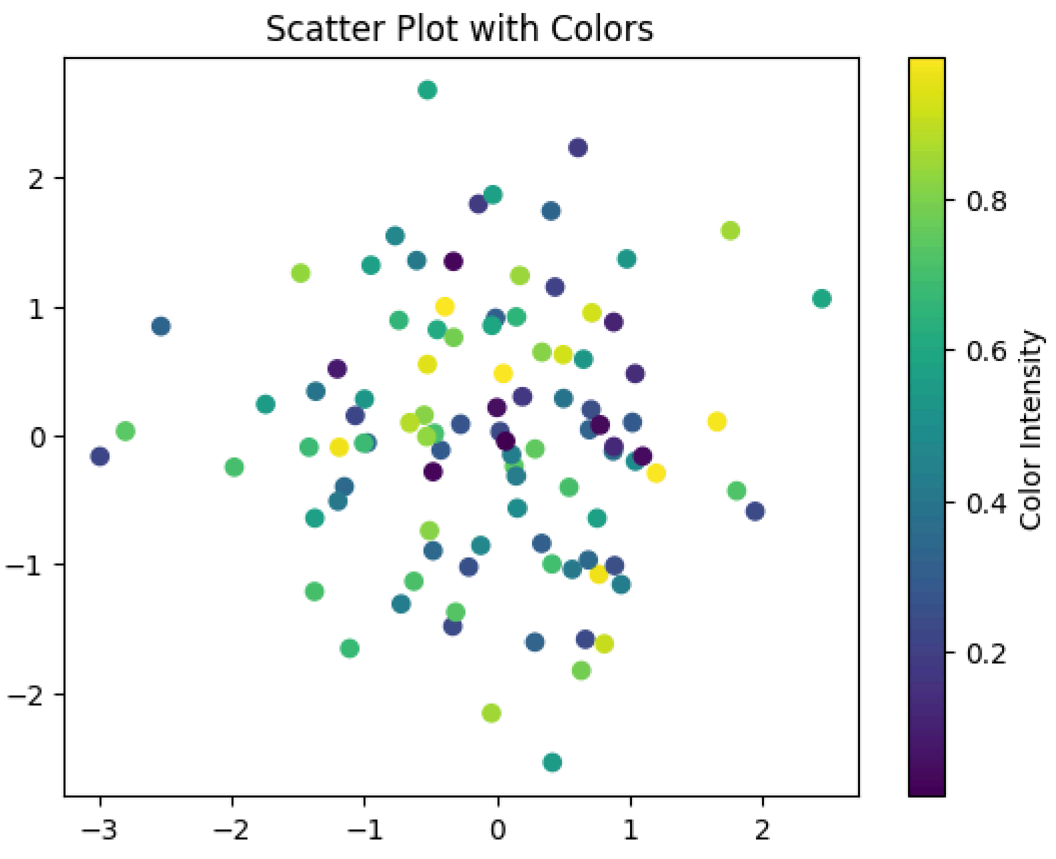
01) WAP to display the value of each bar in a bar chart using Matplotlib.

In [45]:
```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D', 'E']
values = [20, 35, 30, 25, 40]
bars = plt.bar(categories, values)
plt.title('Bar Chart with Values')
plt.xlabel('Categories')
plt.ylabel('Values')
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height, str(height),ha='center', va
plt.show()
```



02) WAP create a Scatter Plot with several colors in Matplotlib?

In [57]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.rand(100)
plt.scatter(x, y, c=colors)
plt.title('Scatter Plot with Colors')
plt.colorbar(label='Color Intensity')
plt.show()
```



03) WAP to Display an Image in Grayscale in Matplotlib.

In [66]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('my_img.jpg')
grayscale_img = img.mean(axis=2)
plt.imshow(grayscale_img, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')
plt.show()
```

Grayscale Image

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 12

```
In [ ]: Name :- Vora Yagnik
        Enrollment No :- 23010101661
```

OOP

01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

```
In [4]: class Student:
            def __init__(self,name,age,grade):
                self.name=name
                self.age=age
                self.grade=grade

        s1 = Student('Yagnik',19,'A++')
        print("Student Name :",s1.name)
        print("Student Age :",s1.age)
        print("Student Grade :",s1.grade)
```

```
Student Name : Yagnik
Student Age : 19
Student Grade : A++
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

```
In [8]: class Bank_Acoount:
            def GetAccountDetails(self):
```

```python
        self.A_No=int(input('Enter Account Number: '))
        self.User_Name=input('Enter User Name: ')
        self.Email=input('Enter Email: ')
        self.A_Type=input('Enter Account Type: ')
        self.A_Balance=int(input('Enter Account Balance'))

    def DisplayAccountDetails(self):
        print('---------------------------------------------')
        print('Account No is : ', self.A_No)
        print('User Name is : ',self.User_Name)
        print('Email ID is: ',self.Email)
        print('Account Type is: ',self.A_Type)
        print('Account Balance is: ',self.A_Balance)
        print('---------------------------------------------')

b1 = Bank_Acoount()
b1.GetAccountDetails()
b1.DisplayAccountDetails()
```

```
---------------------------------------------
Account No is :  23010101661
User Name is :  Vora Yagnik
Email ID is:  xyz123@gmail.com
Account Type is:  shaving
Account Balance is:  120000
---------------------------------------------
```

03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

In [10]:
```python
class Circle:
    def __init__(self,r):
        self.r = r
    def area(self):
        print('Area of circle is: ',3.14*self.r*self.r)

    def perimeter(self):
        print('Perimeter of cicle is: ',2*3.14*self.r)

c1 = Circle(7)
c1.area()
c1.perimeter()
```

```
Area of circle is:  153.86
Perimeter of cicle is:  43.96
```

04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

In [15]:
```python
class Employee:
    def add(self):
        self.name=input('Enter Name: ')
        self.age=int(input('Enter Age: '))
        self.salary=int(input('Enter Salary: '))

    def display(self):
        print('-------------------------')
        print('Name is : ',self.name)
        print('Age is : ',self.age)
        print('Salary is : ',self.salary)
        print('-------------------------')

    def updatename(self):
        self.name=input('Enter New Name: ')

e1 = Employee()
e1.add()
e1.display()
e1.updatename()
e1.display()
```

```
-------------------------
Name is :  Yagnik
Age is :  19
Salary is :  60000
-------------------------
-------------------------
Name is :  Yagnik Vora
Age is :  19
Salary is :  60000
-------------------------
```

05) Create a bank account class with methods to deposit, withdraw, and check balance.

In [18]:

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def deposit(self,a):
        self.balance=self.balance + a
        print("Amount",self.balance,"is deposit")

    def withdraw(self,a):
        self.balance=self.balance - a
        print("Amount",self.balance,"is withdraw")

    def checkBalance(self):
        print("Your Account Balance is - ",self.balance)

b = BankAccount()
b.deposit(1000)
b.deposit(2000)
b.withdraw(2000)
b.withdraw(500)
b.checkBalance()
```

```
Amount 1000 is deposit
Amount 3000 is deposit
Amount 1000 is withdraw
Amount 500 is withdraw
Your Account Balance is -  500
```

06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

In [24]:

```

```

```
Item Details:
Product Name: Laptop ,Stock Count: 100 ,Price: 500.0
None
Product Name: Mobile ,Stock Count: 110 ,Price: 450.0
None
Product Name: Desktop ,Stock Count: 120 ,Price: 500.0
None
Product Name: Tablet ,Stock Count: 90 ,Price: 550.0
None

Update the price of item code - 'I001':
Item updated...
Product Name: Laptop ,Stock Count: 100 ,Price: 505.0
None

Update the stock of item code - 'I003':
Item updated...
Product Name: Desktop ,Stock Count: 115 ,Price: 500.0
None
```

09) Create a Class with instance attributes

In [25]:

```
class Inventory:
    def __init__(self):
        self.inventory = {}
    def add_item(self, item_id, item_name, stock_count, price):
        self.inventory[item_id] = {"item_name": item_name, "stock_count": stock_cou

    def update_item(self, item_id, stock_count, price):
        if item_id in self.inventory:
            self.inventory[item_id]["stock_count"] = stock_count
            self.inventory[item_id]["price"] = price
        else:
            print("Item not found in inventory.")

    def check_item_details(self, item_id):
        if item_id in self.inventory:
            item = self.inventory[item_id]
            return f"Product Name: {item['item_name']}, Stock Count: {item['stock_c
        else:
            return "Item not found in inventory."
```

```
inventory = Inventory()

inventory.add_item("I001", "Laptop", 100, 500.00)
inventory.add_item("I002", "Mobile", 110, 450.00)
inventory.add_item("I003", "Desktop", 120, 500.00)
inventory.add_item("I004", "Tablet", 90, 550.00)
print("Item Details:")
print(inventory.check_item_details("I001"))
print(inventory.check_item_details("I002"))
print(inventory.check_item_details("I003"))
print(inventory.check_item_details("I004"))
print("\nUpdate the price of item code - 'I001':")
inventory.update_item("I001", 100, 505.00)
print(inventory.check_item_details("I001"))
print("\nUpdate the stock of item code - 'I003':")
inventory.update_item("I003", 115, 500.00)
print(inventory.check_item_details("I003"))
```

```
Item Details:
Product Name: Laptop, Stock Count: 100, Price: 500.0
Product Name: Mobile, Stock Count: 110, Price: 450.0
Product Name: Desktop, Stock Count: 120, Price: 500.0
Product Name: Tablet, Stock Count: 90, Price: 550.0

Update the price of item code - 'I001':
Product Name: Laptop, Stock Count: 100, Price: 505.0

Update the stock of item code - 'I003':
Product Name: Desktop, Stock Count: 115, Price: 500.0
```

07)

Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

In [26]:
```
class Student_Kit:
    Principal_Name="Mr. ABC"

    def __init__(self,name):
        self.name=name

    def attendance(self,days):
        self.days=days
        return self.days

s1 = Student_Kit('Madhav')
print(Student_Kit.Principal_Name ,'has issued a certificate to',s1.name,',who was p
```

```
Mr. ABC has issued a certificate to Madhav ,who was present 15 days
```

08) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

In [29]:
```
class Time:
    def __init__(self,h,m):
        self.hour=h
        self.minute=m

    def add(self,a,b):
        self.hour=a.hour + b.hour
        self.minute=a.minute + b.minute

    def display(self):
        print(self.hour,self.minute)

obj1 = Time(11,35)
obj2 = Time(19,20)
obj3 = Time(0,0)
obj3.add(obj1,obj2)
obj1.display()
obj2.display()
obj3.display()
```

```
11 35
19 20
30 55
```

09) WAP to demonstrate inheritance in python.

In [30]:
```
class A:
    def displayA(self):
        print('This is class A')

class B(A):
    def displayB(self):
        print('This is class B')

class C(B):
    def displayC(self):
        print('This is class C')

obj1 = C()
obj1.displayA()
obj1.displayB()
obj1.displayC()
```

```
This is class A
This is class B
This is class C
```

10) Create a child class Bus that will inherit all of the variables and methods of the Vehicle class

class Vehicle:

```
    def __init__(self, name, max_speed, mileage):
        self.name = name
        self.max_speed = max_speed
        self.mileage = mileage
```

Create a Bus object that will inherit all of the variables and methods of the parent Vehicle class and display it.

In [38]:
```
class Vehicle:
    def __init__(self, name, max_speed, mileage):
        self.name = name
        self.max_speed = max_speed
        self.mileage = mileage

class Bus(Vehicle):
    def __init__(self,name, max_speed, mileage):
        super().__init__(name, max_speed, mileage)

    def display(self):
        print("Name:", self.name)
        print("Max Speed:", self.max_speed)
        print("Mileage:", self.mileage)

my_bus = Bus("My Bus", 70, 15)
my_bus.display()
```

```
Name: My Bus
Max Speed: 70
Mileage: 15
```

11) Create a class hierarchy for different types of animals, with a parent Animal class and child classes for specific animals like Cat, Dog, and Bird.

In [41]:
```
class Animal:
    def __init__(self, name):
        self.name = name

class Cat(Animal):
    def make_sound(self):
        return "Meow"

class Dog(Animal):
    def make_sound(self):
        return "Woof"

class Bird(Animal):
    def make_sound(self):
        return "Chirp"
```

```
my_cat = Cat("Cat")
my_dog = Dog("Dog")
my_bird = Bird("Bird")

print(my_cat.name, "says", my_cat.make_sound())
print(my_dog.name, "says", my_dog.make_sound())
print(my_bird.name, "says", my_bird.make_sound())
```

```
Cat says Meow
Dog says Woof
Bird says Chirp
```

12) Create a class hierarchy for different types of vehicles, with a parent Vehicle class and child classes for specific vehicles like Car, Truck, and Motorcycle.

In [42]:

```
class Vehicle:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def start(self):
        print("Starting the vehicle.")

    def stop(self):
        print("Stopping the vehicle.")

    def accelerate(self):
        print("Accelerating the vehicle.")

    def brake(self):
        print("Applying the brakes.")


class Car(Vehicle):
    def __init__(self, make, model, year, num_doors):
        super().__init__(make, model, year)
        self.num_doors = num_doors

    def park(self):
        print("Parking the car.")

    def honk(self):
        print("Honking the car horn.")

    def display(self):
        print("Make - ",self.make)
        print("Model - ",self.model)
        print("Year - ",self.year)
        print("Year - ",self.num_doors)

class Truck(Vehicle):
    def __init__(self, make, model, year, payload_capacity):
        super().__init__(make, model, year)
        self.payload_capacity = payload_capacity

    def load_cargo(self):
        print("Loading cargo into the truck.")

    def unload_cargo(self):
        print("Unloading cargo from the truck.")

    def display(self):
        print("Make - ",self.make)
        print("Model - ",self.model)
        print("Year - ",self.year)
        print("Year - ",self.payload_capacity)


class Motorcycle(Vehicle):
    def __init__(self, make, model, year, num_wheels):
        super().__init__(make, model, year)
        self.num_wheels = num_wheels

    def wheelie(self):
        print("Popping a wheelie on the motorcycle.")

    def lean(self):
        print("Leaning into the turn while riding the motorcycle.")

    def display(self):
        print("Make - ",self.make)
        print("Model - ",self.model)
        print("Year - ",self.year)
        print("Year - ",self.num_wheels)

t = Truck("Volvo","S20",2020,"23 tonnes")
t.start()
t.accelerate()
t.brake()
t.stop()
t.display()
```

```
Starting the vehicle.
Accelerating the vehicle.
Applying the brakes.
Stopping the vehicle.
Make -  Volvo
Model -  S20
Year -  2020
Year -  23 tonnes
```

13) Create a class hierarchy for different types of bank accounts, with a parent Account class and child classes for specific account types like Checking, Savings, and Credit.

In [43]:
```
class Account:
    def __init__(self, account_number, balance):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited ${amount}. New balance is ${self.balance}.")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds.")
        else:
            self.balance -= amount
            print(f"Withdrew ${amount}. New balance is ${self.balance}.")


class Checking(Account):
    def __init__(self, account_number, balance, overdraft_limit):
        super().__init__(account_number, balance)
        self.overdraft_limit = overdraft_limit

    def withdraw(self, amount):
        if amount > self.balance + self.overdraft_limit:
            print("Insufficient funds.")
        else:
            self.balance -= amount
            print(f"Withdrew ${amount}. New balance is ${self.balance}.")


class Savings(Account):
    def __init__(self, account_number, balance, interest_rate):
        super().__init__(account_number, balance)
        self.interest_rate = interest_rate

    def accrue_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        print(f"Accrued ${interest} in interest. New balance is ${self.balance}.")


class Credit(Account):
    def __init__(self, account_number, balance, credit_limit):
        super().__init__(account_number, balance)
        self.credit_limit = credit_limit

    def make_purchase(self, amount):
        if amount > self.balance + self.credit_limit:
            print("Purchase declined.")
        else:
            self.balance -= amount
            print(f"Made purchase for ${amount}. New balance is ${self.balance}.")

    def make_payment(self, amount):
        self.balance += amount
        print(f"Made payment of ${amount}. New balance is ${self.balance}.")

abc = Checking("122",500,200)
abc.withdraw(900)
```

```
Insufficient funds.
```

14) Create a Shape class with a draw method that is not implemented. Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors. Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

In [44]:

```python
class Shap:
    def draw(self):
        raise NotImplementedError("draw method not implemented")


class Rectangle(Shap):
    def draw(self):
        print("Drawing a rectangle")


class Circle(Shap):
    def draw(self):
        print("Drawing a circle")


class Triangle(Shap):
    def draw(self):
        print("Drawing a triangle")

shapes = [Rectangle(), Circle(), Triangle()]

for s in shapes:
    s.draw()
```

```
Drawing a rectangle
Drawing a circle
Drawing a triangle
```

15) Create a Person class with a constructor that takes two arguments name and age. Create a child class Employee that inherits from Person and adds a new attribute salary. Override the **init** method in Employee to call the parent class's **init** method using the super keyword, and then initialize the salary attribute.

In [46]:

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age


class Employee(Person):
    def __init__(self, name, age, salary):
        super().__init__(name, age)
        self.salary = salary

    def display(self):
        print("Name - ",self.name)
        print("Age - ",self.age)
        print("Salary - ",self.salary)

e = Employee("abc",27,50000)
e.display()
```

```
Name -  abc
Age -  27
Salary -  50000
```