Part-A

1. Write a program to convert number of days to measure of time given in years, weeks and days(Ignore leap year)

```
def convert_days(num_days):
    # Define constants
    days_in_year = 365
    days_in_week = 7

    # Calculate years
    years = num_days // days_in_year
    remaining_days = num_days % days_in_year

    # Calculate weeks
    weeks = remaining_days // days_in_week
    remaining_days %= days_in_week

    return years, weeks, remaining_days

# Input number of days
num_days = int(input("Enter number of days: "))

# Perform conversion
years, weeks, remaining_days = convert_days(num_days)

# Display the result
print(f"{num_days} days is equal to {years} years, {weeks} weeks, and {remaining_days} days.")
```

Approach 2

```
def convert_days(num_days):
    years = num_days // 365
    weeks = (num_days % 365) // 7
    days = (num_days % 365) % 7
    return years, weeks, days

num_days = int(input("Enter number of days: "))
years, weeks, days = convert_days(num_days)
print(f"{num_days} days is equal to {years} years, {weeks} weeks, and {days} days.")
```

2. Write a Program to Prompt for a Score between 0 and 100 Score Grade: >= 90 S, >= 80 A, >= 70 B, >= 60 C, >= 50 D, >= 40 E, < 40 F

```
score = float(input(“Enter your score"))
if score < 0 or score > 100:
print('Wrong Input')
elif score >= 90:
print('Your Grade is “S" ')
elif score >= 80:
print('Your Grade is “A" ')
elif score >= 70:
```

```
print('Your Grade is “B" ')
elif score >= 60:
print('Your Grade is “C" ')
elif score >= 60:
print('Your Grade is “D ')
elif score >= 60:
print('Your Grade is “E" ')
else:
print('Your Grade is "F" ‘)
```

3. Program to Check If a Given Year Is a Leap Year

Approach 1

```
year = int(input('Enter a year'))
if year % 4 == 0:
if year % 100 == 0:
if year % 400 == 0:
print(f'{year} is a Leap Year')
else:
print(f'{year} is not a Leap Year')
else:
print(f'{year} is a Leap Year')
else:
print(f'{year} is not a Leap Year’)
```

Approach 2

```
year=int(input('enter the year'))
if(year%4==0 and (year%400==0 or year%100!=0)):
    print(f'lear year')
else:
    print(f'not a leap year')
```

```
enter the year2020
lear year
```

4. Enter the numbers till the user enters word “end”, Count and print the number of Odd, Even, Positive number, Negative number, Prime number and average in each case.

```
def is_prime(n):
    return n > 1 and all(n % i != 0 for i in range(2, int(n ** 0.5) + 1))

odd_count = even_count = positive_count = negative_count = prime_count = 0

while True:
    user_input = input("Enter a number or 'end' to finish: ").strip().lower()
    if user_input == 'end':
        break
    try:
        number = int(user_input)
        if number % 2 == 0:
            even_count += 1
```

```
        else:
            odd_count += 1
        if number > 0:
            positive_count += 1
        elif number < 0:
            negative_count += 1
        if is_prime(abs(number)):
            prime_count += 1
    except ValueError:
        print("Invalid input. Please enter a number or 'end'.")

print(f"Odd numbers: {odd_count}")
print(f"Even numbers: {even_count}")
print(f"Positive numbers: {positive_count}")
print(f"Negative numbers: {negative_count}")
print(f"Prime numbers: {prime_count}")
```

5. Write a python program to display the Fibonacci sequence between the intervals specified by the user and also print the count of odd, even and prime Fibonacci numbers in the same interval.

```
def is_prime(n):
    if n < 2: return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

start = int(input("Enter the start of the interval: "))
end = int(input("Enter the end of the interval: "))

fib_sequence = [0, 1]
while fib_sequence[-1] <= end:
    fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])

fib_sequence = [num for num in fib_sequence if start <= num <= end]
odd_count = sum(1 for num in fib_sequence if num % 2 != 0)
even_count = sum(1 for num in fib_sequence if num % 2 == 0)
prime_count = sum(1 for num in fib_sequence if is_prime(num))

print(f"Fibonacci sequence between {start} and {end}: {fib_sequence}")
print(f"Count of odd Fibonacci numbers: {odd_count}")
print(f"Count of even Fibonacci numbers: {even_count}")
print(f"Count of prime Fibonacci numbers: {prime_count}")
```

6. Write a Program to compute area of hexagon, area of a pentagon, area of octagon, area of decagon using functions

$$A = \frac{1}{4}\sqrt{5\left(5+2\sqrt{5}\right)}\,a^2$$

```
#finds the area of a pentagon
#import math
from math import sqrt
a = 5
area1 = (sqrt(5 * (5 + 2 *(sqrt(5)))) * a * a) / 4
print("Area of Pentagon: ", area1)
```

7.WAP that accepts a sentence and calculate the number of vowels, consonants, words, digits, blanks, uppercase letters and lowercase letters.

```
def count_characters(sentence):
    vowels = consonants = words = digits = blanks = uppercase = lowercase = 0
    for char in sentence:
        if char.isalpha():
            if char.lower() in 'aeiou':
                vowels += 1
            else:
                consonants += 1
            if char.isupper():
                uppercase += 1
            else:
                lowercase += 1
        elif char.isdigit():
            digits += 1
        elif char.isspace():
            blanks += 1
    words = len(sentence.split())
    return vowels, consonants, words, digits, blanks, uppercase, lowercase
sentence = input("Enter a sentence: ")
vowels, consonants, words, digits, blanks, uppercase, lowercase = count_characters(sentence)
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
print(f"Number of words: {words}")
print(f"Number of digits: {digits}")
print(f"Number of blanks: {blanks}")
print(f"Number of uppercase letters: {uppercase}")
print(f"Number of lowercase letters: {lowercase}")
```

8. Write a Program to display required patterns with Stars/ Alphabets /Numbers :

```
num=int(input("Enter the number of rows"))
for i in range(num):
        print((str(i+1)+ ' ') *num)
```

```
Enter the number of rows5
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

```
num=int(input("Enter the number of rows"))
for i in range(num):
        print((str(num)+ ' ') *num)
```

```
Enter the number of rows5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
```

```
num=int(input("Enter the number of rows"))
for i in range(num):
        print((chr(65+i)+' ')*(i+1))
```

```
Enter the number of rows5
A
B B
C C C
D D D D
E E E E E
F F F F F F
```

```
num=int(input("Enter the number of rows"))
for i in range(num+1):
        print("  "*(num-i)+ "* "*i)
```

```
Enter the number of rows5

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

```
num=int(input("Enter the number of rows"))
for i in range(num,0,-1):
        print(" "*(num-i)+ "* "*i)
```

```
Enter the number of rows5
* * * * *
 * * * *
  * * *
   * *
    *
```

```
num=int(input("Enter the number of rows "))
for i in range(num+1):
        print(" "*(num-i)+ "* "*i)
for i in range(num-1,0,-1):
        print(" "*(num-i)+ "* "*i)
```

```
Enter the number of rows 5

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

```
num=int(input("Enter the number of rows "))
for i in range(num+1):
        print("* "*i)
for i in range(num-1,0,-1):
        print("* "*i)
```

```
Enter the number of rows 5

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

```
size = int(input("Enter the size of the butterfly: "))
```

```
# Upper part of the butterfly
for i in range(size):
    for j in range(i + 1):
        print("*", end=" ")
    for j in range(2 * (size - i - 1)):
        print(" ", end=" ")
    for j in range(i + 1):
        print("*", end=" ")
    print()

# Lower part of the butterfly
for i in range(size - 1, -1, -1):
    for j in range(i + 1):
        print("*", end=" ")
    for j in range(2 * (size - i - 1)):
        print(" ", end=" ")
    for j in range(i + 1):
        print("*", end=" ")
    print()
```

```
Enter the size of the butterfly: 5
*             *
* *           * *
* * *       * * *
* * * *     * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * *     * * * *
* * *       * * *
* *           * *
*             *
```

```
for row in range(7):
    for col in range(7):
        if row in {0,1,2,3,4,5,6} and col in {0,6}:
            print("*", end=" ")
        elif row==1 and col in {1,5}:
            print("*", end=" ")
        elif row==2 and col in {2,4}:
            print("*", end=" ")
        elif row==3 and col==3:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

```
*         *
```

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

```
from colorama import Style, Back, Fore
def pattern_P():
    for row in range(7):
        for col in range(5):
            if col==0:
                print(f"{Fore.BLUE}*", end=" ")
            elif row in {0,3} and col in {1,2,3}:
                print(f"{Fore.BLUE}*", end=" ")
            elif row in {1,2} and col==4:
                print(f"{Fore.BLUE}*", end=" ")
            else:
                print(" ", end=" ")
        print()
    print()
pattern_P()
```

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

(Using funcations)

9. Write Python Program to Sort Numbers in a List in Ascending Order Using Bubble Sort by Passing the List as an Argument to the Function Call.

Write Python Program to Sort Numbers in a List in Ascending Order Using Bubble Sort by Passing the List as an Argument to the Function Call

```
def bubble_sort(list_items):
    for i in range(len(list_items)):
        for j in range(len(list_items)-i-1):
            if list_items[j] > list_items[j+1]:
                temp = list_items[j]
                list_items[j] = list_items[j+1]
                list_items[j+1] = temp
    print(f"The sorted list using Bubble Sort is {list_items}")
items_to_sort = [5, 3, 2, 4, 1]
bubble_sort(items_to_sort)
```

```
The sorted list using Bubble Sort is [1, 2, 3, 4, 5]
```

10. Prompt the user to enter list of marks, increase the odd marks by 1 and print the list of modified marks, average marks.

```
marks = input("Enter the list of marks separated by spaces: ").split()
marks = [int(mark) + (int(mark) % 2 != 0) for mark in marks]
average_marks = sum(marks) / len(marks)

print("Modified marks:", marks)
print("Average marks:", average_marks)
```

11. Write a program to find Mean, Variance and Standard Deviation of List Numbers

```
import math
def statistics(list_items):
    mean = sum(list_items)/len(list_items)
    print(f"Mean is {mean}")

    variance = 0
    for item in list_items:
        variance += (item-mean)**2
    variance /= len(list_items)
    print(f"Variance is {variance}")

    standard_deviation = math.sqrt(variance)
    print(f"Standard Deviation is {standard_deviation}")

statistics([1, 2, 3, 4])
```

```
Mean is 2.5
Variance is 1.25
Standard Deviation is 1.118033988749895
```

12. Write a python program to add/subtract two matrices

```
matrix_1=[[1,2,3],[4,5,6],[7,8,9]]
matrix_2=[[1,1,1],[1,1,1],[1,1,1]]
result_addition=[[0,0,0],[0,0,0],[0,0,0]]
result_subtraction=[[0,0,0],[0,0,0],[0,0,0]]
for row in range(len(matrix_1)):
   for column in range(len(matrix_2[0])):
      result_addition[row][column]=matrix_1[row][column]+matrix_2[row][column]
      result_subtraction[row][column]=matrix_1[row][column]-matrix_2[row][column]

print(f"Addition matrix is- ")
for i in result_addition:
```

```
  print(i)

print(f"Subtraction matrix is- ")
for i in result_subtraction:
  print(i)
```

```
Addition matrix is-
[2, 3, 4]
[5, 6, 7]
[8, 9, 10]
Subtraction matrix is-
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
```

```python
# Write Python Program to Add Two Matrices
matrix_1 = [[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]]
matrix_2 = [[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]]
matrix_result = [[0, 0, 0],
                 [0, 0, 0],
                 [0, 0, 0]]

for rows in range(len(matrix_1)):
    for columns in range(len(matrix_2[0])):

        matrix_result[rows][columns] = matrix_1[rows][columns] + matrix_2[rows][columns]

print("Addition of two matrices is")
for items in matrix_result:
    print(items)
```

```
Addition of two matrices is
[2, 4, 6]
[8, 10, 12]
[14, 16, 18]
```

13. Write a python program to input information for n number of students as given below: Name, Registration number, Attendance, Total marks. The user has to specify a value for n number of students. The program should output the registration number and marks of a specified student given his name.

```
def student_details(number_of_students):
    student_name={}
    for i in range(0, number_of_students):
        name=input("Enter the name of student- ")
        USN=input("Enter the USN of student- ")
        attendance=input("Enter the attendance of student- ")
        marks=input("Enter the total marks of student- ")
        student_name[name]=[USN,attendance,marks]
    student_search=input("Enter the name of the student you want to search- ")
    if student_search not in student_name.keys():
```

```
        print(f"Student you are searching for is not in class")
    else:
        print(f"Student you are searching for is in class")
        print(f"Student's USN is- {student_name[student_search][0]}")
        print(f"Student's attendance is- {student_name[student_search][1]}")
        print(f"Student's marks is- {student_name[student_search][2]}")

def main():
    number_of_students=int(input("Enter the total number of students- "))
    student_details(number_of_students)

if __name__== "__main__":
    main()
```

```
Enter the total number of students- 2
Enter the name of student- Asif
Enter the USN of student- 2345
Enter the attendance of student- 78
Enter the total marks of student- 67
Enter the name of student- Kiran
Enter the USN of student- 2346
Enter the attendance of student- 45
Enter the total marks of student- 67
Enter the name of the student you want to search- Asif
Student you are searching for is in class
Student's USN is- 2345
Student's attendance is- 78
Student's marks is- 67
```

14. Write a program to search an element using Binary Search of a Sorted List/ Linear search.

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
```

```
    return -1

# Example usage:
sorted_list = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
target = 13

result = binary_search(sorted_list, target)
if result != -1:
    print(f"Element {target} found at index {result}.")
else:
    print(f"Element {target} not found in the list.")
```

15. Write a Program to sort an array in ascending/ descending order using Bubble sort/selection sort/Insertion sort.

Bubble sort:

```
def bubble_sort(arr, ascending=True):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if (ascending and arr[j] > arr[j+1]) or (not ascending and arr[j] < arr[j+1]):
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Example usage:
arr = [64, 34, 25, 12, 22, 11, 90]
print("Sorted array in ascending order:", bubble_sort(arr.copy()))
print("Sorted array in descending order:", bubble_sort(arr.copy(), ascending=False))
```

selection sort:

```
def selection_sort(arr, ascending=True):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if ascending:
                if arr[j] < arr[min_idx]:
                    min_idx = j
            else:
                if arr[j] > arr[min_idx]:
                    min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

# Example usage:
arr = [64, 34, 25, 12, 22, 11, 90]
```

```
print("Sorted array in ascending order:", selection_sort(arr.copy()))
print("Sorted array in descending order:", selection_sort(arr.copy(), ascending=False))
```

Insertion sort:

```
def insertion_sort(arr, ascending=True):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >= 0 and ((ascending and arr[j] > key) or (not ascending and arr[j] < key)):
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
    return arr

# Example usage:
arr = [64, 34, 25, 12, 22, 11, 90]
print("Sorted array in ascending order:", insertion_sort(arr.copy()))
print("Sorted array in descending order:", insertion_sort(arr.copy(), ascending=False))
```

Part -B

1. Write a program to print the number of lines, words, and characters present in the given file.

```
#Program to print the number of lines,words and characters present in the given
file?
import os,sys
fname=input("Enter File Name: ")
if os.path.isfile(fname):
    print("File exists:",fname)
    f=open(fname,"r")
else:
    print("File does not exist:",fname)
    sys.exit(0)
lcount=wcount=ccount=0
for line in f:
    lcount=lcount+1
    ccount=ccount+len(line)
    words=line.split()
    wcount=wcount+len(words)
print("The number of Lines:",lcount)
print("The number of Words:",wcount)
print("The number of Characters:",ccount)
```

```
Enter File Name: studs.txt
File exists: studs.txt
The number of Lines: 1
The number of Words: 5
The number of Characters: 29
```

2. Write a python program to find the longest word in a file. Get the file name from the user.

Write Python Program to Find the Longest Word in a File. Get the File Name from User.

```
#Write Python Program to Find the Longest Word in a File.
#Get the File Name from User.
def read_file(file_name):
    with open(file_name) as fh:
        longest_word = ""
        for each_row in fh:
            word_list = each_row.rstrip().split()
            for each_word in word_list:
                if len(each_word) > len(longest_word):
                    longest_word = each_word
    print(f"The longest word in the file is: {longest_word}")
def main():
    file_name = input("Enter file name: ")
    read_file(file_name)
if __name__ == "__main__":
    main()
```

```
Enter file name: quotes.txt
The longest word in the file is: intelligence
```

3. Write a Python program to create a dictionary in a CSV file with the heading USN, NAME, Attendance ,Marks, then read and display the content of that CSV file

```
In [*]: #CSV==>Comma seperated values
import csv
with open("IAT1.csv","w", newline="") as f:
    w=csv.writer(f) # returns csv writer object
    w.writerow(["USN","STUD NAME", "ATTENDANCE", "MARKS"])
    n=int(input("Enter Number of Students:"))
    for i in range(n):
        usn=input("Enter Student USN :")
        studname=input("Enter Student Name:")
        attend=input("Enter Attendance:")
        marks=input("Enter Marks:")
        w.writerow([usn, studname, attend,marks])
print("Total student marks written to csv file successfully")
Enter Number of Students:

In [ ]:
```

5. Write Python Program to Calculate Area and Perimeter of different Shapes using Polymorphism

```python
#Write Python Program to Calculate Area and Perimeter of
#Different Shapes Using Polymorphism
import math
class Shape:
    def area(self):pass
    def perimeter(self):pass
class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height
    def area(self):print(f"Area of Rectangle is {self.width * self.height}")
    def perimeter(self):print(f"Perimeter of Rectangle is {2 * (self.width + self.height)}")
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):print(f"Area of Circle is {math.pi * self.radius ** 2}")
    def perimeter(self):print(f"Perimeter of Circle is {2 * math.pi * self.radius}")
def shape_type(shape_obj):
    shape_obj.area()
    shape_obj.perimeter()
def main():
    rectangle_obj = Rectangle(10,20)
    circle_obj = Circle(10)
    for each_obj in [rectangle_obj, circle_obj]:
        shape_type(each_obj)
if __name__ == "__main__":
    main()
```

Conclusion:

6. Write a python program to demonstrate polymorphism using classes Vehicle, Bike, Car, Aeroplane.

```python
#Program 11.24: Program to Demonstrate Polymorphism in Python
class Vehicle:
    def __init__(self, model):
        self.model = model
    def vehicle_model(self):
        print(f"Vehicle Model name is {self.model}")
class Bike(Vehicle):
    def vehicle_model(self):
        print(f"Vehicle Model name is {self.model}")
class Car(Vehicle):
    def vehicle_model(self):
        print(f"Vehicle Model name is {self.model}")
class Aeroplane:
    pass
def vehicle_info(vehicle_obj):
    vehicle_obj.vehicle_model()

def main():
    ducati = Bike("Ducati-Scrambler")
    beetle = Car("Volkswagon-Beetle")
    boeing = Aeroplane()
    for each_obj in [ducati, beetle, boeing]:
        try:
            vehicle_info(each_obj)
        except AttributeError:
            print("Expected method not present in the object")
if __name__ == "__main__":
    main()
```

7. Write a program to simulate a bank account with support for DepositMoney, WithdrawMoney and ShowBalance operations.

```python
#Write Python Program to Simulate a Bank Account with Support
#for depositMoney, withdrawMoney and showBalance Operations
class BankAccount:
    def __init__(self, name):
        self.user_name = name
        self.balance = 0.0
    def show_balance(self):
        print(f"{self.user_name} has a balance of {self.balance} dollars")
    def withdraw_money(self, amount):
        if amount > self.balance:
            print("You don't have sufficient funds in your account")
        else:
            self.balance -= amount
            print(f"{self.user_name} has withdrawn an amount of {self.balance} dollars")
    def deposit_money(self, amount):
        self.balance += amount
        print(f"{self.user_name} has deposited an amount of {self.balance} dollars")
def main():
    savings_account = BankAccount("Anamika")
    savings_account.deposit_money(1000)
    savings_account.show_balance()
    savings_account.withdraw_money(500)
    savings_account.show_balance()
if __name__ == "__main__":
    main()
```

8. Write a program to demonstrate multiple inheritance

```python
#Program 11.19: Program to Demonstrate Multiple Inheritance
class Poissonier:
    def __init__(self, poissonier_role):
        self.poissonier_role = poissonier_role
    def display_poissonier_chef_info(self):
        print(f"Chef is mainly involved in preparing {self.poissonier_role}")

class Entremetier:
    def __init__(self, entremetier_role):
        self.entremetier_role = entremetier_role
    def display_entremetier_chef_info(self):
        print(f"Chef is mainly involved in preparing {self.entremetier_role}")

class Cook(Poissonier, Entremetier):
    def __init__(self, poissonier_role, entremetier_role):
        Poissonier.__init__(self, poissonier_role)
        Entremetier.__init__(self, entremetier_role)
    def invoke_base_class_methods(self):
        Poissonier.display_poissonier_chef_info(self)
        Entremetier.display_entremetier_chef_info(self)

def main():
    print(f"Is Cook a derived class of Poissonier Base Class? {issubclass(Cook,(Entremetier, Poissonier))}")
    chef = Cook("SeaFood", "Vegetables")
    chef.invoke_base_class_methods()

if __name__ == "__main__":
    main()
```

9. Write a python program for simulation of Amplitude Modulated signal with time domainrepresentation.

```
import numpy as np
import matplotlib.pyplot as plt

A_c = 2 #initialize values
f_c = 10
f_m = 1
m1 = 0.7
m2 = 1
m3 =1.3
t_max = 10
fs =100

t = np.linspace(0, t_max, t_max*fs+1)
carrier = np.cos(2*np.pi*f_c*t)
modulator = np.cos(2*np.pi*f_m*t)
amunder = (A_c+A_c*m1*modulator)*carrier
am100 = (A_c+A_c*m2*modulator)*carrier
amover = (A_c+A_c*m3*modulator)*carrier

plt.subplot(5,1,1)
plt.title('Amplitude Modulation'); plt.plot(modulator,'g')
plt.ylabel('Amplitude') ; plt.xlabel('Message signal')

plt.subplot(5,1,2)
plt.plot(carrier,'r') ; plt.ylabel('Amplitude') ; plt.xlabel('Carrier signal')

plt.subplot(5,1,3)
plt.plot(amunder,color="blue"); plt.ylabel('Amplitude');plt.xlabel('AM signal under')

plt.subplot(5,1,4)
plt.plot(am100,color="black"); plt.ylabel('Amplitude'); plt.xlabel('AM signal 100%')

plt.subplot(5,1,5)
plt.plot(amover,color="purple") ; plt.ylabel('Amplitude')
plt.xlabel('AM signal overmodulation'); plt.subplots_adjust(hspace=1); plt.rc('font',size=10)

plt.show()
display('Amplitude Modulation')
```

10. Write a python program to simulate removal of noise from signals.

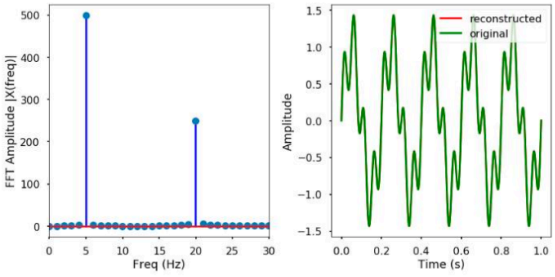
```
#REMOVAL OF NOISE FROM SIGNALS
from math import pi
import scipy.fftpack as sf
import scipy.signal as sig
import matplotlib.pyplot as plt
import numpy as np

# Generate a signal
Fs = 200; #sampling freq
t = 4; # no of taps
n = np.arange(0,t,1/Fs) #number of samples
f = 3; #frequency
x = np.sin(2*pi*f*n)
xx=x

# Generate a noise
y = np.random.normal(0, 0.2, np.size(x)); # AWGN
x = x + y; # noisy signal

plt.figure(1); plt.subplot(2,1,1) ; plt.plot(n,x);
plt.title('Noisy Sinusoidal Wave') ; plt.xlabel('Time(s)');plt.ylabel('Amplitude')

# Take spectral analysis
X_f = abs(sf.fft(x))
l = np.size(x)
fr = (Fs/2)*np.linspace(0,1,l) #frequency range
xl_m = (2/l)*abs(X_f[0:np.size(fr)]);

plt.subplot(2,1,2) ; plt.plot(fr,20*np.log10(xl_m)); plt.title('Spectrum of Noisy signal')
plt.xlabel('Frequency(Hz)'); plt.ylabel('Magnitude'); plt.tight_layout()


# Create a BPF
o = 2; #filter order
fc = np.array([1,5]) #freq band
wc = 2*fc/Fs; [b,a] = sig.butter(o, wc, btype = 'bandpass') #obtaining coefficients

# filter response
[W,h] = sig.freqz(b,a, worN = 1024)
W = Fs* W/(2*pi)

plt.figure(2); plt.subplot(3,1,1) ; plt.plot(W, 20*np.log10(h));
plt.title('Filter Freq. Response') ; plt.xlabel('Frequency(Hz)'); plt.ylabel('Magnitude')

# Filter signal
x_filt = sig.lfilter(b,a,x)

plt.subplot(3,1,2)
plt.plot(n,x_filt); plt.title('Filtered Signal'); plt.tight_layout();

plt.subplot(3,1,3) ; plt.plot(n,xx);
plt.title('Original Sinusoidal Wave') ; plt.xlabel('Time(s)'); plt.ylabel('Amplitude')
```

11. Write a program to compute DFT and IDFT for a discrete signal

Computation of DFT (Discrete Fourier Transform) and Inverse DFT for an input discrete time signal.

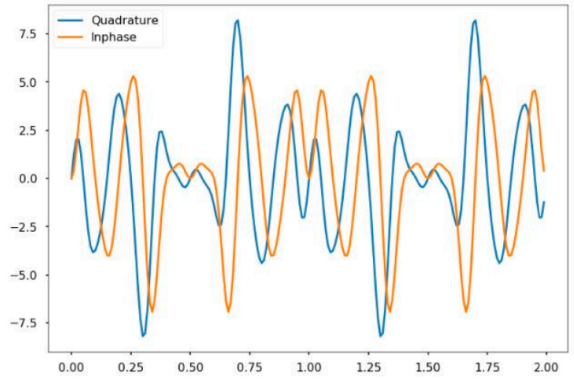
```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('seaborn-poster')
# sampling rate
sr = 100
# sampling interval
ts = 1.0/sr
t = np.arange(0,1,ts)
freq = 1
x = 3*np.sin(2*np.pi*freq*t)
freq = 4
x += np.sin(2*np.pi*freq*t)
freq = 7
x += 0.5* np.sin(2*np.pi*freq*t)
plt.figure(figsize = (8, 6))
plt.plot(t, x, 'r')
plt.ylabel('Amplitude')
plt.show()
display('Amplitude')
```

```
#example of using FFT and IFFT with some random data
import numpy as np
import matplotlib.pyplot as plt

# Generate some random data
N = 1000
t = np.linspace(0, 1, N)
x = np.sin(2 * np.pi * 5 * t) + 0.5 * np.sin(2 * np.pi * 20 * t)

# Perform FFT
X = np.fft.fft(x)
freq = np.fft.fftfreq(N, d=t[1] - t[0])

# Plot FFT
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.stem(freq, np.abs(X), 'b')
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amplitude |X(freq)|')
plt.xlim(0, 30)

# Perform IFFT
x_reconstructed = np.fft.ifft(X)

# Plot original and reconstructed signals
plt.subplot(122)
plt.plot(t, np.real(x_reconstructed), color='r', label='reconstructed')
plt.plot(t, x, color='g', label='original')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.tight_layout()
plt.legend()
plt.show()
```

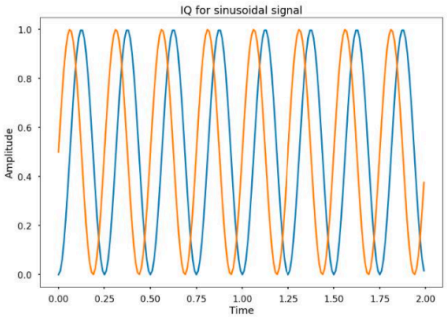


12. Write a program to extract In-phase and quadrature phase signals from a random data.

```python
#Extract the inphase and quadrature signals from a random signal.
import numpy as np
import matplotlib.pyplot as plt
ts = np.arange(0,2,0.01)
f = 5
AI = 2*np.sin(2*np.pi*f*ts) #Asin(wt)
AQ = 2*np.cos(2*np.pi*f*ts) #Acos(wt)
t = np.arange(0,2,0.01)
freq = 1.
x = 3*np.sin(2*np.pi*freq*t)
freq = 4
x += np.sin(2*np.pi*freq*t)
freq = 7
x += 0.5* np.sin(2*np.pi*freq*t)
Quad = x * AQ
Inph = x * AI
plt.plot(ts,Quad)
plt.plot(ts,Inph)
plt.legend(['Quadrature','Inphase'])
plt.show()
```




```python
#For simple Sinusoid signal:
import matplotlib.pyplot as plt
import numpy as np
Fs = 100
fc=2
t = np.arange(0,2,1/Fs)
Ip = np.sin(2*np.pi*fc*t)
Qp = np.cos(2*np.pi*fc*t)
x3 = np.sin(2*np.pi*2*t)
x3Ip = x3 * Ip
x3Iq = x3 * Qp + 0.5
plt.figure()
plt.plot(t,x3Ip)
plt.plot(t,x3Iq)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('IQ for sinusoidal signal')
```



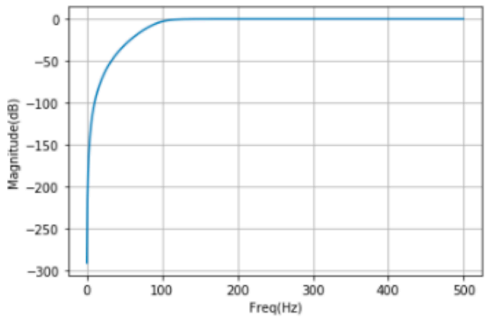
13. Write a program to design a IIR Butterworth Low pass/High pass/Bandpass/Band stop filter for a given specifications

IIR AND FIR FILTER DESIGN

Design IIR Butterworth Low pass filter

```
import matplotlib.pyplot as plt
import scipy.signal as sig
import numpy as np
from math import pi
#Design IIR butterworth
Fs=1000;
n=5;
fc=100;
w_c = 2*fc/Fs #conversion to radians
[b,a] = sig.butter(n,w_c,btype='low') #obtain coefficients
[w,h] = sig.freqz(b,a,worN=2000) #getting the frequency response
w = Fs*w/(2*pi) #digital frequency domain
h_db = 20*np.log10(abs(h)) #Magnitude in dB
plt.figure()
plt.plot(w,h_db)
plt.xlabel('Freq(Hz)')
plt.ylabel('Magnitude(dB)')
plt.grid('on')
```

```
#Design IIR Butterworth High pass filter
import matplotlib.pyplot as plt
import scipy.signal as sig
import numpy as np
from math import pi
#Design IIR butterworth High Pass
Fs=1000;
n=5;
fc=100;
w_c = 2*fc/Fs
[b,a] = sig.butter(n,w_c,btype='high')
[w,h] = sig.freqz(b,a,worN=2000)
w = Fs*w/(2*pi)
h_db = 20*np.log10(abs(h))
plt.figure()
plt.plot(w,h_db)
plt.xlabel('Freq(Hz)')
plt.ylabel('Magnitude(dB)')
plt.grid('on')
```



14. Write a program to design a FIR low pass filter using Hamming Window

```python
#Design FIR Low pass filter using Hamming Window
import matplotlib.pyplot as plt
import scipy.signal as sig
import numpy as np
from math import pi
# Low pass
Fs=1000;
N=17
fc=200;
w_c=2*fc/Fs #converting to radians/sec
t=sig.firwin(N,w_c) #value for taps
[w,h] = sig.freqz(t,worN=2000) #normalized with 2000
w = Fs*w/(2*pi)
h_db = 20*np.log10(abs(h))
plt.figure()
plt.plot(w,h_db)
plt.xlabel('Freq(Hz)')
plt.ylabel('Magnitude(dB)')
plt.grid('on')
```

15. Write a program to simulate digital modulation technique ON-OFF Keying.

DIGITAL MODULATION OF SIGNALS

```python
#DIGITAL MODULATION OF SIGNALS
#Amplitude Shift Keying
import matplotlib.pyplot as plt
import numpy as np
from math import pi
plt.close('all')
def binary(symbol,sym_len):
    import numpy as np
    rand_n=np.random.rand(symbol)
    rand_n[np.where(rand_n >= 0.5)]=1
    rand_n[np.where(rand_n <= 0.5)]=0
    sig=np.zeros(int(symbol*sym_len))
    id_1=np.where(rand_n==1)
    for i in id_1[0]:
        temp=int(i*sym_len)
        sig[temp:temp+sym_len] = 1
    return sig
# carrier information
Fs = 1000; # sampling frequency
fc = 100; # carrier frequency
T = 1 # simulation time (sec)
t = np.arange(0,T,1/Fs)
# carrier wave
x = np.sin(2*pi*fc*t);
#Generate random binary signal
Td = 0.1; # Bit duration
samples = int(Td * Fs) # samples in one bit
sym = int(np.floor(np.size(t)/samples))
sig = binary(sym, samples)
# plot
plt.subplot(2,1,1)
plt.plot(t,sig); plt.xlabel('Time(s)'); plt.ylabel('Amplitude')
plt.title('Random Binary Signal'); plt.grid()
# Generate ASK
ask = x * sig;
plt.subplot(2,1,2) ; plt.plot(t,ask); plt.xlabel('Time(s)');
plt.ylabel('Amplitude') ; plt.title('Amplitude Shift Keying')
plt.plot(t,sig)
plt.grid()
plt.tight_layout()
```