1. Write a program that allow user to enter the particulars of a student and store them into variables. Then, display the output as follows.

```
***Welcome to the Department of Electrical Engineering***
Student ID: KIE170001
Name: Amin Ali
Date of Birth: 17/01/1997
Height(cm): 174
Weight(kg): 80
*** End ***
```

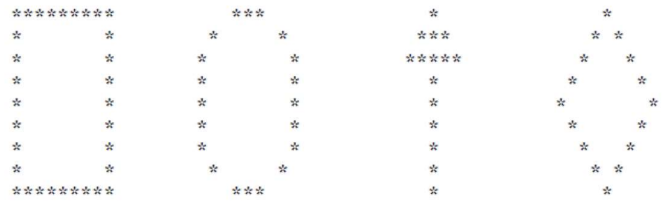
Sample Answer:

```
import os #implement clear function

student={}
student["id"]=input("Enter Student ID: ")
student["name"]=input("Enter Name: ")
student["dob"]=input("Enter Date of Birth: ")
student["height"]=input("Enter Height (cm): ")
student["weight"]=input("Enter Weight (kg): ")

clear = lambda: os.system('cls') #declare clear
clear()
print("***Welcome to the Department of Electrical Engineering***")
print("Student ID: "+student["id"])
print("Name: "+student["name"])
print("Date of Birth: "+student["dob"])
print("Height(cm): "+student["height"])
print("Weight(kg): "+student["weight"])
print("*** End ***")
```

2. Write a program that prints a box, an oval, an arrow and a diamond in the same line as below. Try to use single, double and triple quotes in quoting below lines. In this question, which type of quote you prefer to use?



Sample Answer:

```
print('*********\t\t  ***  \t\t  *  \t\t    *')
print('*       *\t\t *   * \t\t *** \t\t   * *')
print('*       *\t\t*     *\t\t*****\t\t  *   *')
print('*       *\t\t*     *\t\t  *  \t\t *     *')
print('*       *\t\t*     *\t\t  *  \t\t*       *')
print('*       *\t\t*     *\t\t  *  \t\t *     *')
print('*       *\t\t*     *\t\t  *  \t\t  *   *')
print('*       *\t\t *   * \t\t  *  \t\t   * *')
print('*********\t\t  ***  \t\t  *  \t\t    *')
```

3. Write a program that allow user to calculate area for a chosen shape. An example of execution of such program is as below.

```
Area Calculator for Basic Shapes
================================

   1. Square
   2. Rectangle
   3. Triangle
   4. Circle

Choose a shape [1-4] that you want calculate its area : 2

Enter length : 4
Enter base   : 2

Area of rectangle is 8.0
```

Sample Answer:

```
import math

print('Area Calculator for Basic Shapes')
print('================================',end='\n\n\n\t')
print('1. Square','2. Rectangle','3. Triangle','4. Circle',sep='\n\t')
a=int(input('Choose a shape [1-4] that you want calculate its area : '))
if a==1:
    l=float(input('Enter length : '))
    b=float(input('Enter base   : '))
    ans=l*b
    print('Area of square is %f' %ans)
elif a==2:
    l=float(input('Enter length : '))
    b=float(input('Enter base   : '))
    ans=l*b
    print('Area of rectangle is %f' %ans)
elif a==3:
    l=float(input('Enter length : '))
    b=float(input('Enter base   : '))
    ans=0.5*l*b
    print('Area of triangle is %f' %ans)
elif a==4:
    r=float(input('Enter radius : '))
    ans=math.pi*r*r
    print('Area of circle is %f' %ans)
else:
    print('Error!')
```

1. Complex number in rectangular form,

$$x + yj$$

can be expressed in polar form as a combination of magnitude m and phase p (in radian)

$$m \angle p$$

where

$$m = \sqrt{x^2 + y^2}$$

$$p = \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & x > 0 \\ \tan^{-1}\left(\frac{y}{x}\right) + \pi & x < 0 \end{cases}$$

Write a program to convert a given complex number to its equivalent polar form. Use `numpy.arctan()`, a trigonometric inverse tangent function ($\tan^{-1}$) from `numpy` module, which calculates the radian, given a tangent value.

```
***Complex Number in Polar Form ***
Please enter a complex number: 3+4j
The complex number in polar form
m = 5.0
p = 0.9272952180016122
*** End ***
```

Sample Answer (Using math):

```
import math

print('***Complex Number in Polar Form ***')
a=complex(input('Please enter a complex number: '))
m=math.sqrt(a.real**2+a.imag**2)
if a.real>0:
    p=math.atan(a.imag/a.real)
else:
    p=math.atan(a.imag/a.real)+math.pi
print('The complex number in polar form ')
print('m = %f\np = %f' %(m,p))
print('*** End ***')
```

Sample Answer (Using numpy):

```
from numpy import arctan
import math

print('***Complex Number in Polar Form ***')
a=complex(input('Please enter a complex number: '))
m=(a.real**2+a.imag**2)**0.5
if a.real>0:
    p=arctan(a.imag/a.real)
else:
    p=arctan(a.imag/a.real)+math.pi
print('The complex number in polar form ')
print('m = %f\np = %f' %(m,p))
print('*** End ***')
```

2. Quadratic Equation Solution

$$ax^2 + bx + c = 0$$

Given quadratic equation coefficients, a, b and c, from user, calculate the solution(s) for the quadratic equation.

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}, \qquad \Delta = b^2 - 4ac$$

```
***Quadratic Equation Solver***
Please key in the following quadratic equation coefficients
a : 1.0
b : 1.0
c : -3.75
The solutions for the quadratic equations are
x1 = -2.5
x2 = 1.5
*** End ***
```

Please handle all of below cases, when the discriminant Δ is

- $\Delta > 0$, the solutions are two distinct real numbers
- $\Delta = 0$, the solution is only one real number
- $\Delta < 0$, no solutions for real number

Bonus: Provide also solutions for $\Delta < 0$, which are comprised of two distinct complex numbers.

Sample Answer:

```
import math

print('***Quadratic Equation Solver***')
print('Please key in the following quadratic equation coefficients')
a=float(input('a: '))
b=float(input('b: '))
c=float(input('c: '))

coeff=b**2-4*a*c

print('The solutions for the quadratic equations are')
if coeff>0:
    x1=(-b+math.sqrt(coeff))/(2*a)
    x2=(-b-math.sqrt(coeff))/(2*a)
    print('x1 = %f\nx2 = %f' %(x1,x2))
elif coeff==0:
    x1=(-b+math.sqrt(coeff))/(2*a)
    x2=x1
    print('x1 = %f\nx2 = %f' %(x1,x2))
else:
    x1r=(-b/(2*a))
    x1i=((math.sqrt(-coeff))/(2*a))
    x2r=(-b/(2*a))
    x2i=((math.sqrt(-coeff))/(2*a))
    print('x1 = %f+%fj' %(x1r,x1i))
    print('x2 = %f-%fj' %(x2r,x2i))

print('*** End ***')
```

1. Rewrite the following '`for`' loop as a '`while`' loop.

```
X = 10
for i in range(1, X+1):
    if X % i == 0:
        print(i)
```

Sample Answer:

```
X=10
i=1
while i<X+1:
    if X%i==0:
        print(i)
    i=i+1
```

2. Write a program that prompts for an integer and prints the integer. If something other than an integer is entered, the program keeps asking for an integer.

```
*** Integer Input ***
Please enter an integer: 3a
Error! Please enter an integer: 3.5
Error! Please enter an integer: 35
Integer entered is 35.
*** End ***
```

Hint: Use string `isdigit()` method to decide if there are only numbers inside a string.

Sample Answer:

```
print('*** Integer Input ***')

while 1:
   a=input('Please enter an integer: ')
   if a.isdigit():
       break
   print('Error!',end=' ')
print('Integer entered is ',a)
print('*** End ***')
```

3. Analysis of Student Marks

Given below student marks of a course

Student	1	2	3	4	5	6	7	8	9	10
Mark	35	40	50	60	80	88	20	40	75	99

Count how many students are under each grade, if mark criteria for each grade is as follow

Grade	Marks
A	80~100
B	60~79
C	50~59
D	0~49

and find the overall average μ, standard deviation σ & variance σ^2.

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}, \qquad \sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

```
*** Analysis of Student Marks ***
A : 1
B : 4
C : 1
D : 4
Average = 58.7
Variance = 603.81
Standard Deviation = 24.57
*** End ***
```

Hint: Use a tuple to store the marks, for eg: `marks=(35,40,..)` and access individual mark using an index to the tuple, for eg: `marks[index]`.

Sample Answer:

```
import math

marks=(35,40,50,60,80,88,20,40,75,99)
miu=sum(marks)/len(marks)

a,b,c,d,xmiu=0,0,0,0,0
for i in range(0,len(marks)):
    if marks[i]>=80:
        a=a+1
    elif marks[i]>=60:
        b=b+1
    elif marks[i]>=50:
        c=c+1
    else:
        d=d+1
    xmiu=xmiu+(marks[i]-miu)**2

print('*** Analysis of Student Marks ***')
print('A : %i\nB : %i\nC : %i\nD : %i' %(a,b,c,d))
print('Average = %.1f' %miu)
print('Varience = %.2f' %(xmiu/len(marks)))
print('Standard Deviation = %.2f' %(math.sqrt(xmiu/len(marks))))
print('*** End ***')
```

Sample Answer (using numpy):

```
import numpy as np

marks=np.array([35,40,50,60,80,88,20,40,75,99])

a,b,c,d=0,0,0,0
for i in range(0,len(marks)):
    a=a+1 if (marks[i]>=80) else a
    b=b+1 if (marks[i]>=60 and marks[i]<=79) else b
    c=c+1 if (marks[i]>=50 and marks[i]<=59) else c
    d=d+1 if (marks[i]<=49) else d
print('*** Analysis of Student Marks ***')
print('A : %i\nB : %i\nC : %i\nD : %i' %(a,b,c,d))
print('Average = %.1f' %np.average(marks))
print('Varience = %.2f' %np.var(marks))
print('Standard Deviation = %.2f' %np.std(marks))
print('*** End ***')
```

1. Given `x = [1, 2, 3]`, write the Python code to:

   (a) Create a list `y` such that changing `x` also changing `y`
   (b) Create a list `y` such that changing `x` does not change `y`

```
x,y:  [1, 555, 3] [1, 555, 3]
x,y:  [1, 555, 3] [1, 2, 3]
```

Sample Answer:

```
#Question 1(a)
x=[1,2,3]
y=x
x[1]=555
print("x,y: ",x,y)
#Question 1(b)
x=[1,2,3]
y=x[:]
x[1]=555
print("x,y: ",x,y)
```

2. Come up with four different ways to create list of 25 1's without simply typing 25 1's(Hint: You can use `for` and `while` loop with `append()` method for two of the ways)

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Sample Answer:

```
list1,list2=[],[]
for i in range (25):
    list1.append(1)
while len(list2)<25:
    list2.append(1)
list3=[1]*25
list4=[1 for i in range(25)]
print(list1,list2,list3,list4,sep="\n")
```

3. Write a code using a `for` loop that take a string S as an argument and returns S in reversed order. For example if S = "Python", it should return "nohtyP"

```
Enter a string : Python
The reversed of  Python  is  nohtyP
```

Sample Answer 1:

```
S=input("Enter a string : ")
s=""
for i in range(1,len(S)+1):
    s+=S[len(S)-i]
print("The reversed of ",S," is ",s)
```

Sample Answer 2:

```
S=input("Enter a string : ")
print("The reversed of ",S," is ","".join(reversed(S)))
```

1. Define a function `f`, when used with `reduce` will return the element with maximum value in a sequence.

```
a = [10,30,20]
m = reduce(f,a)
print(m) # 30 will be printed out
```

Sample Answer:

```
from functools import reduce

a=[10,30,20]
m=reduce(lambda x,y: x if x>y else y, a )
print(m)
```

2. Given an integer number `n`, define a function `g`, when used with `reduce`, will calculate the factorial of `n`, which is `n!`.

```
n = 5
f = reduce(g,…) # Complete this code
print(f) # 120 will be printed out
```

Sample Answer:

```
from functools import reduce

n=5
f=reduce(lambda x,y:x*y, range(1,n+1) )
print(f)
```

3. Use `filter` and `reduce`, to calculate the sum of only positive numbers of a sequence. You can define any auxiliary function that you need.

```
a = [1, -2, 5, 6, 8]
t = reduce(…) # Complete this code
print(t) # 20 will be printed out
```

Sample Answer:

```
from functools import reduce

a=[1,-2,5,6,8]
t=reduce(lambda x,y: x+y, filter(lambda a:a>0,a))
print(t)
```

4. Use `map` and `reduce`, to calculate the sum of magnitudes of a sequence of complex numbers. You can define any auxiliary function that you need.

```
c = [3+4j, 5, 5j]
m = reduce(…) # Complete this code
print(m) # 15.0 will be printed out
```

Sample Answer:

```
from functools import reduce

c=[3+4j,5,5j]
m=reduce(lambda a,b:a+b,map(lambda a:(a.real**2+a.imag**2)**0.5,c))
print(m)
```

1. The following program calculates the Fibonacci value given a number (let say 50).

```
def fibonacci(n):
    """Recursive fibonacci that remembers previous values"""
    if n not in fibo_dict:
        # recursive case, store in the dict
        fibo_dict[n] = fibonacci(n-1) + fibonacci(n-2)
    return fibo_dict[n]

# global fibonacci dictionary.
fibo_dict = {}

# enter the base cases
fibo_dict[0] = 1
fibo_dict[1] = 1

fibo_val = input("Calculate what Fibonacci value:")
print("Fibonnaci value of",fibo_val,"is",
      fibonacci(int(fibo_val)))
```

Rewrite the program using generator function. Use the following template:

```
def fibo_generator():
    .
    .
    .


fibo_generator_object = fibo_generator()

print(next(fibo_generator_object))
print(next(fibo_generator_object))

for . . . . . :
    print(next(fibo_generator_object), end=', ')
```

Sample output:

```
0
1
1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181,
6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229,
832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817,
39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733,
1134903170, 1836311903, 2971215073, 4807526976, 7778742049, 12586269025,
20365011074,
```

Sample Answer:

```
def fibo_generator():
    a,b=0,1
    while 1:
        yield a
        a,b=b,a+b

fibo_generator_object = fibo_generator()
print(next(fibo_generator_object))
print(next(fibo_generator_object))

for _ in range (50) :
    print(next(fibo_generator_object), end=', ')
```

Create a `BankAccount` class, which has below methods

a. Constructor that receives the account holder name and the initial balance in the account.

b. A static method `displayTotalAccount` that displays the total number of accounts that have been created.

c. A `setBalance` method that sets the account with a new balance.

d. A `getBalance` method that returns the current account balance.

e. A `withdraw` method that withdraws certain amount from the account.

f. A `deposit` method that deposits certain amount into the account.

g. A `__str__` method to display the account holder name and current balance.

Example of how above methods are used is illustrated below:

```
a1 = BankAccount("Abu", 100)
b1 = BankAccount("Ali", 200)
BankAccount.displayTotalAccount()
# "Total number of account created is 2" will be displayed.
print(a1)
# "Abu has 100 in the account" will be displayed.
a1.withdraw(200)
# "Balance is sufficient" will be displayed.
a1.withdraw(50)
print(a1.getBalance())
# "50" will be displayed.
a1.deposit(100)
print(a1.getBalance())
# "150" will be displayed.
a1.setBalance(50)
print(a1.getBalance())
# "50" will be displayed.
```

Sample Answer:

```
class BankAccount:
    count=0
    def __init__(self,a,b):
        self.name=a
        self.bal=b
        BankAccount.count += 1
    def __str__(self):
        return '%s has %i in the account' %(self.name,self.bal)
    def displayTotalAccount():
        print('Total number of account created is %i' %(BankAccount.count))
    def setBalance(self,i):
        self.bal=i
    def getBalance(self):
        return self.bal
    def withdraw(self,i):
        if (i>self.bal):
            print('Balance is insufficient')
        else:
            self.bal-=i
    def deposit(self,i):
        self.bal+=i

a1 = BankAccount("Abu", 100)
b1 = BankAccount("Ali", 200)
BankAccount.displayTotalAccount()
print(a1)
a1.withdraw(200)
a1.withdraw(50)
print(a1.getBalance())
a1.deposit(100)
print(a1.getBalance())
a1.setBalance(50)
print(a1.getBalance())
```

1. Refactor out the duplicate codes from `Student` and `Lecturer` classes into a new `Person` class. Change the class definitions of `Student` and `Lecturer` to inherit from this new `Person` class.

```
class Student:
    def __init__(self, nm, mk, sn):
        self.name = nm
        self.mykad = mk
        self.studnumber = sn
    def getName(self):
        return self.name
    def getMyKad(self):
        return self.mykad
    def getStudNumber(self):
        return self.studnumber

class Lecturer:
    def __init__(self, nm, mk, sn, sal):
        self.name = nm
        self.mykad = mk
        self.staffnumber = sn
        self.salary = sal
    def getName(self):
        return self.name
    def getMyKad(self):
        return self.mykad
    def getStaffNumber(self):
        return self.staffnumber
    def getSalary(self):
        return self.salary
```

Sample Answer:

```
class Person:
    def __init__(self,nm,mk,sn):
        self.name=nm
        self.mykad=mk
        self.id=sn
    def getName(self):
        return self.name
    def getMyKad(self):
        return self.mykad

class Student(Person):
    def __init__(self,nm,mk,sn):
        Person.__init__(self,nm,mk,sn)
    def getStudNumber(self):
        return self.id

class Lecturer(Person):
    def __init__(self,nm,mk,sn,sal):
        Person.__init__(self,nm,mk,sn)
        self.salary=sal
    def getStaffNumber(self):
        return self.id
    def getSalary(self):
        return self.salary
```

2. Define a `Vector3D` class, and overload its two `__add__` and `__mul__` operators, so that the class can be used such as below.

```
v1 = Vector3D(1,2,3)
v2 = Vector3D(4,5,6)
v3 = v1 + v2 # Carry out vector addition, result is another vector.
print(v3) # "[X=5.00,Y=7.00,Z=9.00]" is printed out
dot = v1 * v2 # Calculate dot product of vectors, result is a scalar.
print(dot) # "32" is printed out.
```

Sample Answer:

```
class Vector3D:
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __add__(self,other):
        return Vector3D(self.x+other.x,self.y+other.y,self.z+other.z)
    def __mul__(self,other):
        return (self.x*other.x)+(self.y*other.y)+(self.z*other.z)
    def __str__(self):
        return "[X=%.2f,Y=%.2f,Z=%.2f]" %(self.x,self.y,self.z)

v1=Vector3D(1,2,3)
v2=Vector3D(4,5,6)
v3=v1+v2
print(v3)
dot=v1*v2
print(dot)
```

1. A file named "dictionary.txt" contains a list of English words, stored line-by-line. Given a search string (for eg: "ach*"), where asterisk '*' is a wildcard that matches any **single** character, find all words from that file that match the search string (for eg: "ache" & "achy" will be printed out, but not "aches").

Sample Answer:

```
f=open('dictionary.txt')
lines=f.readlines()
f.close()
for l in lines:
    if(len(l)==5 and l[0]=='a' and l[1]=='c' and l[2]=='h'):
        print(str(l))
```

2. Given a text file (for eg: "article.txt"), write a program to automatically read the file, capitalize the first character of the first word in a sentence, for each sentence in the file, and save the results into another file (for eg: "article-cap.txt")

Contents of 'article.txt'.

```
python uses whitespace indentation, rather than curly brackets or
keywords, to delimit blocks. an increase in indentation comes after
certain statements; a decrease in indentation signifies the end of the
current block. thus, the program's visual structure accurately
represents the program's semantic structure. this feature is also
sometimes termed the off-side rule.
```

The results, should be in 'article-cap.txt'.

```
Python uses whitespace indentation, rather than curly brackets or
keywords, to delimit blocks. An increase in indentation comes after
certain statements; a decrease in indentation signifies the end of the
current block. Thus, the program's visual structure accurately
represents the program's semantic structure. This feature is also
sometimes termed the off-side rule.
```

Sample Answer:

```
f=open('article.txt','r')
l=f.read()
re=""
l=l.split('. ')
for i in l:
    re+=i[0].upper()+i[1:]+'. '
g=open('article-cap.txt','w')
g.write(re[:-3])
```

Question 1

Write a program that

1. Reads a five-digit integer.
2. Separates the number into its individual digits [Hint: Use combinations of integer division and the remainder operator].
3. Prints the digits in a reverse order separated from one another by a tab.
4. Performs addition with the first, third and the last of the reverse ordered digits; and multiplication with the second and fourth reversed digits, then display the output.

```
Enter a five-digit number: 12345
The digits in reverse order :   5       4       3       2       1
Addition : 5 + 3 + 1 = 9
Multiplication : 4 x 2 = 8
```

Sample Answer 1:

```
a=int(input('Enter a five-digit number: '))
num=[]
for i in range(5):
    num.append(int(a/(10**i)%10))
print('The digits in reverse order : \t',end="")
for i in range(5):
    print(num[i],end="\t")
print('',end='\n')
print('Addition : ',num[0],'+',num[2],'+',num[4],'=',num[0]+num[2]+num[4])
print('Multiplication :',num[1],'x',num[3],'=',num[1]*num[3])
```

Sample Answer 2:

```
a=str(input('Enter a five-digit number: '))
print('The digits in reverse order :',end="")
for i in range(len(a)):
    print((a)[::-1][i],end="\t")
print('\nAddition : ',(a)[::-1][0],'+',(a)[::-1][2],'+',(a)[::-1][4],'=',\
   int((a)[::-1][0])+int((a)[::-1][2])+int((a)[::-1][4]))
print('Multiplication : ',(a)[::-1][1],'x',(a)[::-1][3],'=',\
   int((a)[::-1][1])*int((a)[::-1][3]))
```

Question 2

Let say you need to develop a program to fly a drone. Prior to that you need to find out how far a drone could travel based on certain parameters provided. In the preliminary developing stage, assume that these values are provided by the users although in a real situation these values should be obtained from the system. Assume that parameters needed are voltage (in volts), current (in amperes) and blade speed (in RPM). Assume the formula to calculate the range of travel is based on the following formula:

Range = (Voltage*1.2) + (Current*0.5) + (Blade speed * 0.2)

Write a program that request these three values and print all the values including the range for 4 different cases in a tabular format.

```
Enter the voltage : 2
Enter the current : 1
Enter the rpm :  2000

.
.
.

Enter the voltage : 4
Enter the current : 5
Enter the rpm :  2000

Voltage Current   RPM           Range
2.0      1.0     2000.0         402.9
3.0      2.0     1000.0         204.6
4.0      2.0     1333.0         272.40000000000003
4.0      5.0     2000.0         407.3…
```

Sample Answer:

```
v,c,r=[],[],[]
for _ in range(4):
    v.append(float(input('Enter the voltage : ')))
    c.append(float(input('Enter the current : ')))
    r.append(float(input('Enter the rpm : ')))

print('Voltage\tCurrent\t  RPM\t\t Range')
for i in range(4):
    q=float((v[i]*1.2)+(c[i]*0.5)+(r[i]*0.2))
    print(v[i],c[i],r[i],sep="\t ",end="")
    print('\t\t',q)
```

Question 1

Given below list `A`

```
A = [1, -2, 3.5+4j, 4.5, 5, 4-3.5j, 3.5, 3, -5.5]
```

using `map`, `filter`, and `reduce`, write a program to

1. Create a new list which contains only floating point numbers, and print that list.
2. Create a new list which contains only complex numbers (which has imaginary part), and print that list.
3. Find the total sum of non-negative integer numbers in list `A`, and print the sum.
4. Find the total sum of magnitude for all numbers in list `A`, and the print the sum.
5. Find the minimum and maximum values, in term of magnitude, and print the values.

Sample Answer:

```
from functools import reduce

A=[1,-2,3.5+4j,4.5,5,4-3.5j,3.5,3,-5.5]

#Question 1
floating=list(filter(lambda x:x if x.imag==0 and type(x)==float else
0,A))
print(floating)

#Question 2
comp=list(filter(lambda x:x if x.imag!=0 else 0,A))
print(comp)

#Question 3
sum_non=reduce(lambda x,y:x+y,filter(lambda x: x if type(x)==int and
x>=0 else 0,A))
print(sum_non)

#Question 4
sum_mag=reduce(lambda x,y:x+y,map(lambda a:(a.real**2+a.imag**2)**0.5
if type(a)==complex else a,A))
print(sum_mag)

#Question 5
maxi=reduce(lambda x,y: x if x>y else y, map(lambda
a:(a.real**2+a.imag**2)**0.5,A))
print(maxi)
mini=reduce(lambda x,y: y if x>y else x, map(lambda
a:(a.real**2+a.imag**2)**0.5,A))
print(mini)
```

Question 2

Write a program to find the sum of the first n positive odd numbers, s_{odd}, for example:

$$s_{odd}(3) = 1 + 3 + 5 = 9$$

Using both of below techniques:

1. Iterations, and
2. Recursion

Sample Answer:

```
def sum_iter(n):
    t=0
    i=1
    while i<(2*n)+1:
        if i%2==1:
            t+=i
        i+=2
    return t

def sum_rec(n):
    if n==1:
        return 1
    else:
        return 2*n-1+sum_rec(n-1)

print(sum_iter(3))
print(sum_rec(3))
```