

# **Artificial Intelligence**

## **Lab-02**

**Sameer Ali Butt**  
**01-134202-095**  
**BSCS-6B**

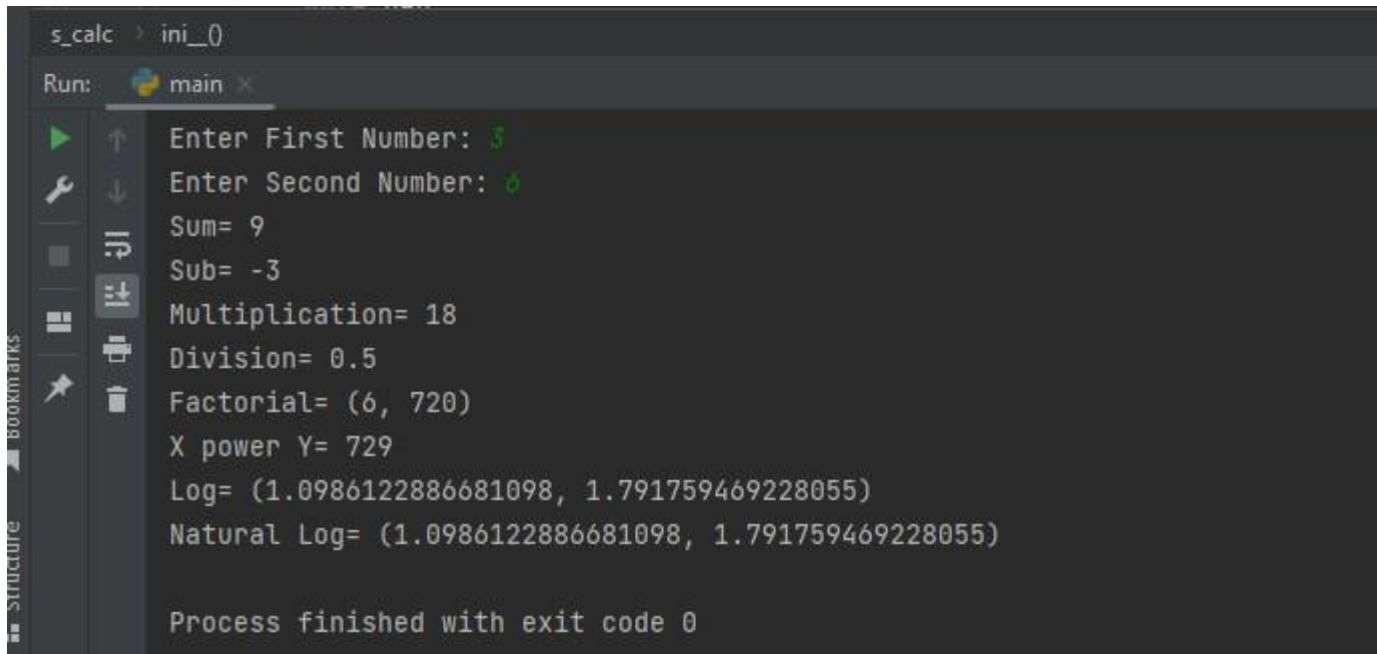
## Lab Journal 2-A:

1. Create a class name basic\_calc with following attributes and methods;  
Two integers (values are passed with instance creation)  
Different methods such as addition, subtraction, division, multiplication  
Create another class inherited from basic\_calc named s\_calc which should have the following additional methods;  
Factorial, x\_power\_y, log, ln etc

### CODE:

```
import math
class basic_calc():
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def Sum(self):
        return self.x+self.y
    def Sub(self):
        return self.x-self.y
    def Mul(self):
        return self.x*self.y
    def Div(self):
        return self.x/self.y
class s_calc(basic_calc):
    def ini_(self,x,y):
        self.x=x
        self.y=y
    def Factorial(self):
        return (math.factorial(self.x),math.factorial(self.y))
    def XpowerY(self):
        return pow(self.x,self.y)
    def Log(self):
        return (math.log(self.x),math.log(self.y))
    def ln(self):
        return (math.log(self.x),math.log(self.y))
num1=int(input("Enter First Number: "))
num2=int(input("Enter Second Number: "))
BC=basic_calc(num1,num2)
SC=s_calc(num1,num2)
print("Sum=",BC.Sum())
print("Sub=",BC.Sub())
print("Multiplication=",BC.Mul())
print("Division=",BC.Div())
print("Factorial=",SC.Factorial())
print("X power Y=",SC.XpowerY())
print("Log=",SC.Log())
print("Natural Log=",SC.ln())
```

## OUTPUT:



```
s_calc > ini_()
Run: main x
Enter First Number: 3
Enter Second Number: 6
Sum= 9
Sub= -3
Multiplication= 18
Division= 0.5
Factorial= (6, 720)
X power Y= 729
Log= (1.0986122886681098, 1.791759469228055)
Natural Log= (1.0986122886681098, 1.791759469228055)

Process finished with exit code 0
```

2. Modify the classes created in the above task under as follows:  
Create a module name basic.py having the class name basic\_calc with all the attributes and methods defined before.  
Now import the basic.py module in your program and do the inheritance step defined before i.e.  
Create another class inherited from basic\_calc named s\_calc which should have the following additional methods;  
Factorial, x\_power\_y, log, ln etc

## CODE:

test.py:

```
class basic_calc():
    def init (self,x,y):
        self.x=x
        self.y=y
    def Sum(self):
        return self.x+self.y
    def Sub(self):
        return self.x-self.y
    def Mul(self):
        return self.x*self.y
    def Div(self):
```

```

        return self.x/self.y
num1=int(input("Enter First Number: "))
num2=int(input("Enter Second Number: "))
BC=basic_calc(num1,num2)
print("Sum=",BC.Sum())
print("Sub=",BC.Sub())
print("Multiplication=",BC.Mul())
print("Division=",BC.Div())

```

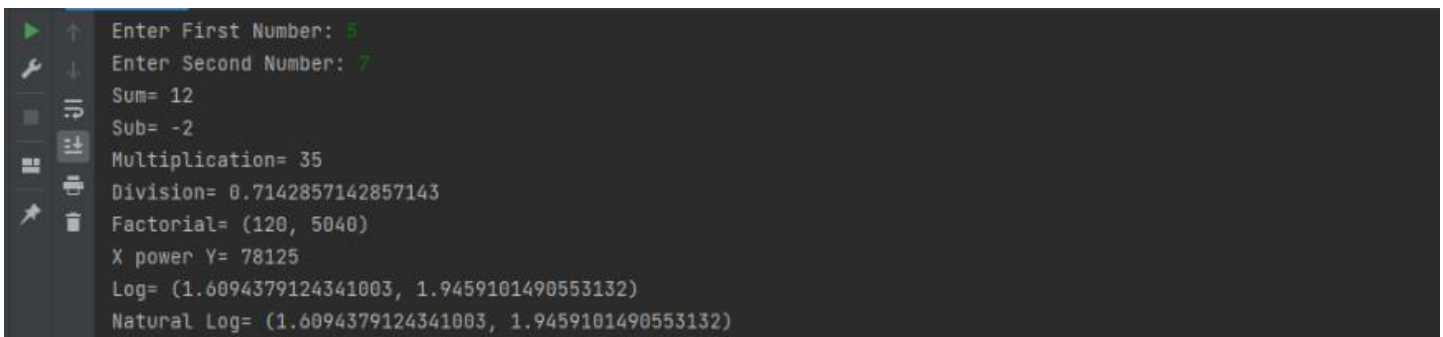
main.py

```

import test
import math
class s_calc(BasicCalculator.basic_calc):
    def ini__(self,x,y):
        self.x=x
        self.y=y
    def Factorial(self):
        return (math.factorial(self.x),math.factorial(self.y))
    def XpowerY(self):
        return pow(self.x,self.y)
    def Log(self):
        return (math.log(self.x),math.log(self.y))
    def ln(self):
        return (math.log(self.x),math.log(self.y))
SC=s_calc(BasicCalculator.num1,BasicCalculator.num2)
print("Factorial=",SC.Factorial())
print("X power Y=",SC.XpowerY())
print("Log=",SC.Log())
print("Natural Log=",SC.ln())

```

OUTPUT:



```

Enter First Number: 5
Enter Second Number: 7
Sum= 12
Sub= -2
Multiplication= 35
Division= 0.7142857142857143
Factorial= (120, 5040)
X power Y= 78125
Log= (1.6094379124341003, 1.9459101490553132)
Natural Log= (1.6094379124341003, 1.9459101490553132)

```

## Lab 2-B

**Lab Journal 2-B:**

1. Create list of Fibonacci numbers after calculating Fibonacci series up to the number n which you will pass to a function as an argument. The number n must be input by the user. They are calculated using the following formula: The first two numbers of the series is always equal to 1, and each consecutive number returned is the sum of the last two numbers. Hint: Can you use only two variables in the generator function?

```
a = 1
```

```
b = 2
```

```
a, b = b, a
```

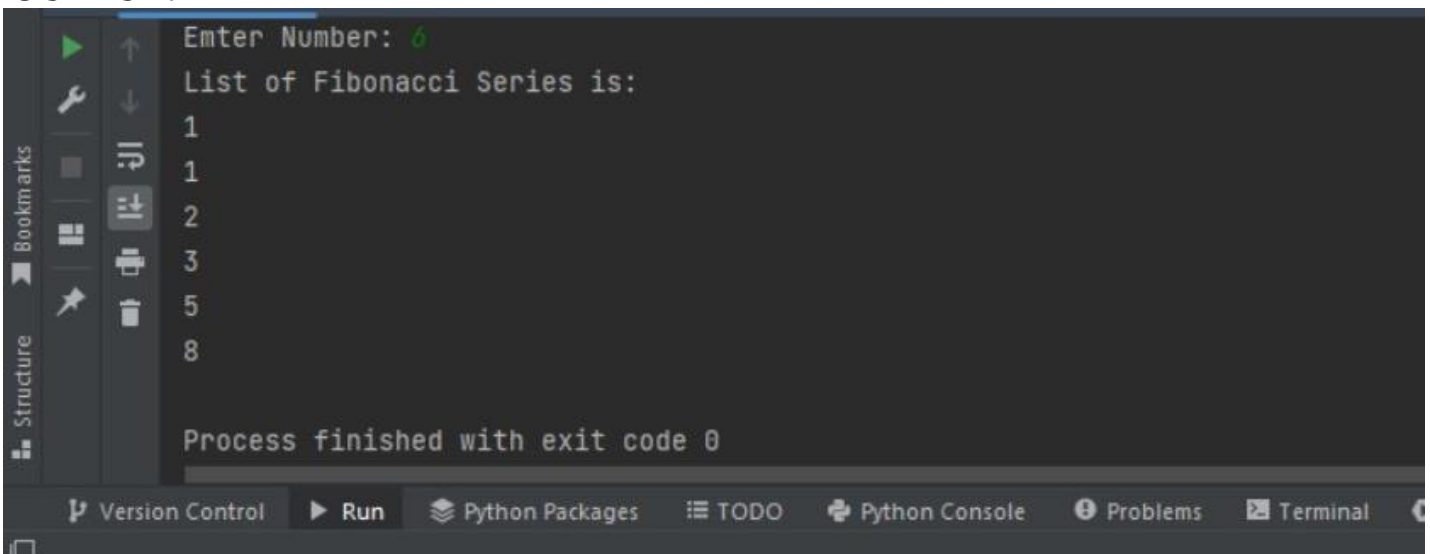
will simultaneously switch the values of a and b.

The first number in the series should be 1. (The output will start like 1,1,2,3,5,8,...)

## CODE:

```
def myfibfunction(n):  
    n1,n2=1,1  
    count=0  
    while count<n:  
        print(n1)  
        sum=n1+n2  
        n1=n2  
        n2=sum  
        count=count+1  
  
num=int(input("Emter Number: "))  
print("List of Fibonacci Series is: ")  
myfibfunction(num)
```

## OUTPUT:



```
Enter Number: 6  
List of Fibonacci Series is:  
1  
1  
2  
3  
5  
8  
  
Process finished with exit code 0
```

1. Write a program that lets the user enter some English text, then converts the text to Pig-Latin. To review, Pig-Latin takes the first letter of a word, puts it at the end, and appends -ay|. The only exception is if the first letter is a vowel, in which case we keep it as it is and append -hay| to the end. For example: -hello| -> -ellohay|, and -image| -

-imagehay|

It will be useful to define a list or tuple at the top called VOWELS. This way, you can check if a letter *x* is a vowel with the expression *x* in VOWELS.

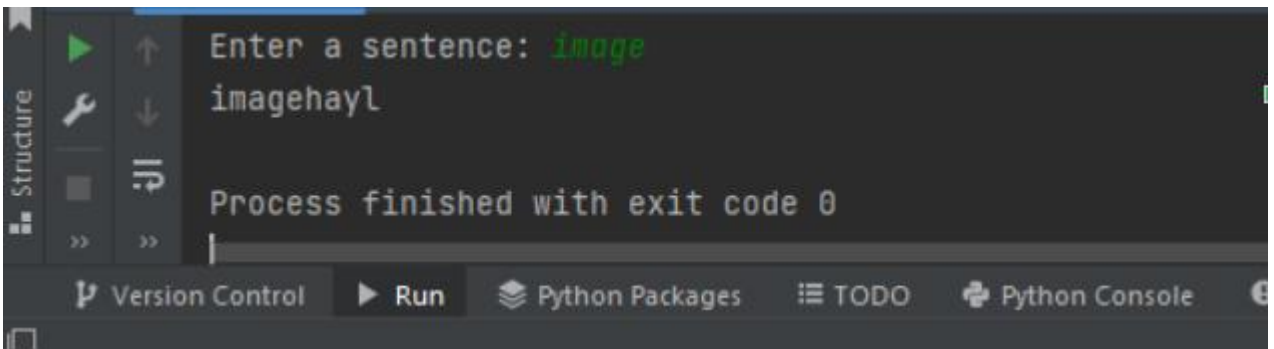
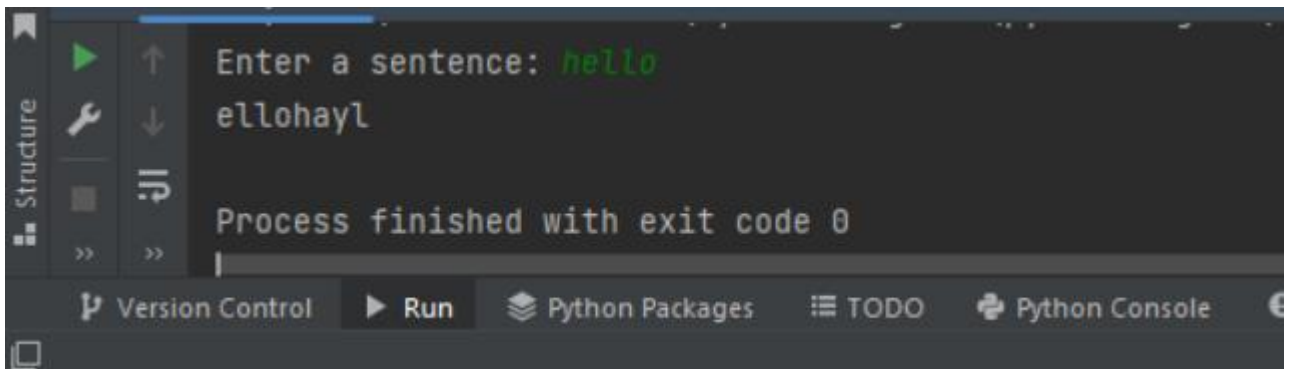
It's tricky for us to deal with punctuation and numbers with what we know so far, so instead, ask the user to enter only words and spaces. You can convert their input from a string to a list of strings by calling `split` on the string:

`-My name is John Smith|.split(' ') -> [-My|, -name|, -is|, -John|, -Smith|]`

## CODE:

```
vowels=('A','a','E','e','I','i','O','o','U','u')
sentence=input("Enter a sentence: ")
sentence=sentence.split()
for x in range(len(sentence)):
    if sentence[x][0] in vowels:
        sentence[x]+='hayl'
    else:
        sentence[x]=sentence[x][1:]+sentence[x][0]
        sentence[x]+='ayl'
sentence=' '.join(sentence)
print(sentence)
```

## OUTPUT:



## u s Task:

1. Add the following functions to the s\_calc you created in Task 2A-1 and Task2A-2.
  - a. sin(x)
  - b. cos(x)
  - c. tan(x)
  - d. sqrt

## CODE:

```
import math

class basic_calc():
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def Sum(self):
        return self.x+self.y
    def Sub(self):
        return self.x-self.y
    def Mul(self):
        return self.x*self.y
    def Div(self):
        return self.x/self.y

class s_calc(basic_calc):
    def __init__(self, x, y):
        super().__init__(x, y)
    def Factorial(self):
        return (math.factorial(self.x),math.factorial(self.y))
    def XpowerY(self):
        return pow(self.x,self.y)
    def Log(self):
        return (math.log(self.x),math.log(self.y))
    def ln(self):
        return (math.log(self.x),math.log(self.y))

    def sin(self):
```



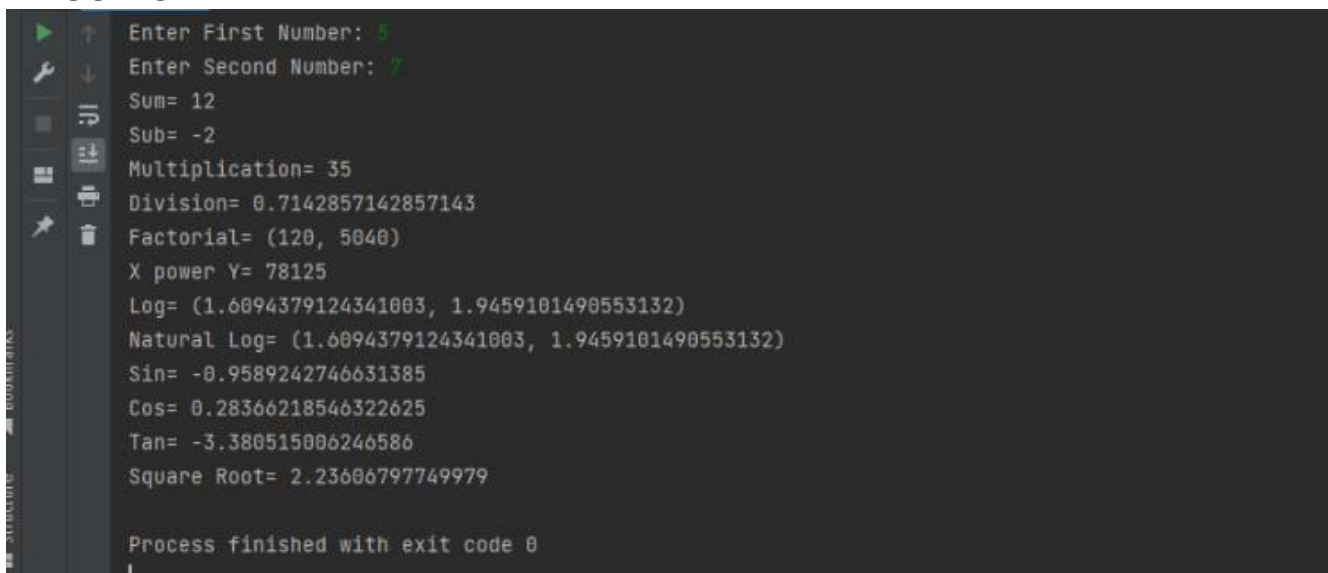
```

        return math.sin(self.x)
def cos(self):
    return math.cos(self.x)
def tan(self):
    return math.tan(self.x)
def sqrt(self):
    if self.x < 0:
        return "Cannot calculate square root of a negative
number"
    else:
        return math.sqrt(self.x)

num1=int(input("Enter First Number: "))
num2=int(input("Enter Second Number: "))
BC=basic_calc(num1,num2)
SC=s_calc(num1,num2)
print("Sum=",BC.Sum())
print("Sub=",BC.Sub())
print("Multiplication=",BC.Mul())
print("Division=",BC.Div())
print("Factorial=",SC.Factorial())
print("X power Y=",SC.XpowerY())
print("Log=",SC.Log())
print("Natural Log=",SC.ln())
print("Sin=",SC.sin())
print("Cos=",SC.cos())
print("Tan=",SC.tan())
print("Square Root=",SC.sqrt())

```

## OUTPUT:



```

Enter First Number: 5
Enter Second Number: 7
Sum= 12
Sub= -2
Multiplication= 35
Division= 0.7142857142857143
Factorial= (120, 5040)
X power Y= 78125
Log= (1.6094379124341003, 1.9459101490553132)
Natural Log= (1.6094379124341003, 1.9459101490553132)
Sin= -0.9589242746631385
Cos= 0.28366218546322625
Tan= -3.380515006246586
Square Root= 2.23606797749979

Process finished with exit code 0

```

