

**Tab 1**

## **NUMBER POWER**

```
class py_pow:  
    def powr(self, x, n):  
        if x==0 or x==1 or n==1:  
            return x  
  
        if x==-1:  
            if n%2 ==0:  
                return 1  
            else:  
                return -1  
        if n==0:  
            return 1  
        if n<0:  
            return 1/self.powr(x,-n)  
        val = self.powr(x,n//2)  
        if n%2 ==0:  
            return val*val  
        return val*val*x  
  
x=int(input("Enter x value :"))  
n=int(input("Enter n value :"))  
print("pow(x,n) value is:",py_pow().powr(x,n));
```

## **Reverse**

```
def rev_words(string):  
    words = string.split(' ')  
    rev = ' '.join(reversed(words))  
    return rev  
  
s= "Python is good"  
print ("reverse: ",rev_words(s))  
  
s2= "Hello from Study tonight"  
print ("reverse: ",rev_words(s2))
```

## **SUB STRING**

```
# all of the following are equivalent
my_string = 'Hello'
print(my_string)
my_string = "Hello"
print(my_string)
my_string = ""Hello"""
print(my_string)
# triple quotes string can extend multiple lines
my_string = """Hello, welcome to the world of Python"""
print(my_string)
c=" mlritm"
print(my_string+c)
# substring function

print(my_string[5:11])
```

## **ARITHMETIC OPERATIONS**

```
x = 15
y = 4
# Output: x + y = 19
print('x + y =',x+y)

# Output: x - y = 11
print('x - y =',x-y)

# Output: x * y = 60
print('x * y =',x*y)

# Output: x / y = 3.75
print('x / y =',x/y)

# Output: x // y = 3
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```

## DICT / DICTIONARY

```
my_dict = {'name':'Jack', 'age': 26}

# Output: Jack
print(my_dict['name'])

# Output: 26
print(my_dict.get('age'))

# Trying to access keys which doesn't exist throws error
# my_dict.get('address')
# my_dict['address']
```

## Celsius

```
# Python Program to convert temperature in celsius to fahrenheit
# change this value for a different result
celsius = 37.5

# calculate fahrenheit
fahrenheit = (celsius * 1.8) + 32
print('%0.1f degree Celsius is equal to %0.1f degree Fahrenheit' %(celsius,fahrenheit))
```

## TRAINGLE

```
# Python Program to find the area of triangle
```

```
a = 5
```

```
b = 6
```

```
c = 7
```

```
# Uncomment below to take inputs from the user
```

```
# a = float(input('Enter first side: '))
```

```
# b = float(input('Enter second side: '))
```

```
# c = float(input('Enter third side: '))
```

```
# calculate the semi-perimeter
```

```
s = (a + b + c) / 2
```

```
# calculate the area
```

```
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
```

```
print('The area of the triangle is %0.2f %area)
```

## **TUPLE**

```
# empty tuple
my_tuple = ()
print(my_tuple)

# tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)

# tuple can be created without parentheses
# also called tuple packing
my_tuple = 3, 4.6, "dog"
print(my_tuple)

# tuple unpacking is also possible
a, b, c = my_tuple
print(a)
print(b)
print(c)
```

## **PATTERN**

```
n=5;
for i in range(n):
    for j in range(i):
        print('*', end="")
    print()

for i in range(n,0,-1):
    for j in range(i):
        print('*', end="")
    print()
```

## **FIBONACCI**

```
n_terms = int(input ("How many terms the user wants to print? "))

# First two terms
n_1 = 0
n_2 = 1
count = 0

# Now, we will check if the number of terms is valid or not
if n_terms <= 0:
    print ("Please enter a positive integer, the given number is not valid")
# if there is only one term, it will return n_1
elif n_terms == 1:
    print ("The Fibonacci sequence of the numbers up to", n_terms, ": ")
    print(n_1)
# Then we will generate Fibonacci sequence of number
else:
    print ("The fibonacci sequence of the numbers is:")
    while count < n_terms:
        print(n_1)
        nth = n_1 + n_2
        # At last, we will update values
        n_1 = n_2
        n_2 = nth
        count += 1
```

**Tab 2**

