In [ ]:
```
# Q2. Open a new python file and import the Module.py file.

# a. Now call the 4 methods from the Module.py file, i.e., addition(), subtraction(),multiplication(), and division().
```

In [ ]:
```
from Intellipaat import *
```

In [1]:
```
Addition(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 Addition(10,20)

NameError: name 'Addition' is not defined
```

In [ ]:
```
Substraction(10,20)
```

In [ ]:
```
Multiplication(10,20)
```

In [ ]:
```
Division(10,20)
```

In [ ]:
```
# Q3. From the Module file, import only the addition() and pass the arguments so that it can
# display the result from the method.
```

In [2]:
```
from Intellipaat import Addition
```

In [3]:
```
Addition(10,20)
```

```
30
```

In [4]:
```
Substraction(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 Substraction(10,20)

NameError: name 'Substraction' is not defined
```

In [5]:
```
Multiplication(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 Multiplication(10,20)

NameError: name 'Multiplication' is not defined
```

In [6]:
```
Division(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 Division(10,20)

NameError: name 'Division' is not defined
```

In [ ]:
```
# Q5. From the Module file, import both the multiplication() and division() and pass the arguments so that it can display
# the result from the methods.
```

In [1]:
```
from Intellipaat import Multiplication,Division
```

In [2]:
```
Addition(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 Addition(10,20)

NameError: name 'Addition' is not defined
```

In [3]:
```
Substraction(10,20)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 Substraction(10,20)

NameError: name 'Substraction' is not defined
```

In [4]:
```
Multiplication(10,20)
```

```
200
```

In [5]:
```
Division(10,20)
```

0.5

In [ ]:
```
# Q6. Create a python if-else program :

# a. check if the given numbers are greater or not
# b. check whether the given number is an armstrong number or not, and
# c. check whether the given number is a prime number or not.

# Make use of python if-else, and elif statements for the same.
```

In [18]:
```
# Armstrong Number

sum = 0
num = int(input("Enter a value to check an armstring number : "))
temp = num
print("The value of num is :", num)
stringnum = str(num) # 153 => "153" ----> stringnum
lengthofnum = len(stringnum)
while(num > 0):
    digit  = num % 10
    cube = digit**lengthofnum
    sum = sum + cube
    num = num // 10
    print("The value of num is :", num)

if(temp == sum):
    print("Armstrong")
else:
    print("Not Armstrong")


# while(num > 0)   digit    cube    sum    num
#    True            3       27      27     15
#    True            5       125     152    1
#    True            1       1       153    Garbage Value(-ve)
#    False ------> Exit Condition

# 153 == 153

# Armstrong
```

```
Enter a value to check an armstring number : 1234
The value of num is : 1234
The value of num is : 123
The value of num is : 12
The value of num is : 1
The value of num is : 0
Not Armstrong
```

In [ ]:

```python
# 153 = 1**3 + 5**3 + 3**3 == 153 ------> Armstrong number
# 1234 = 1**4 + 2**4 + 3**4 + 4**4 == 1234  ----------> Not Armstrong number

# Find the number of digits => num -> str(num) ----> stringnum ------> len(stringnum) ------> lengthstringnum
# To keep a copy of num so that we can compare it later. temp = num
# Until and Unless the number is less than 0 zero run the loop
    # 1. I have to extract the indivisual digits. num % 10 -------> last digit
    # 2. I have to cube each digits (indiviual_digit to the power of total number of digits) last_digit**lengthstringnum ------> power
    # 3. I have to add all of it together. sum = 0 -----> sum + power -----> sum
    # 4. I have to remove the last digit. num // 10 ------> num
# Compare the original number with sum to check if armstrong number or not. temp == sum

# 153
# 15
# 1
# 0
# 3 -----> 3**3 -----> 27 -------> 27 ------> 15
# 5 -----> 5**3 -----> 125 ------> 152 -----> 1
# 1 -----> 1**3 -----> 1 --------> 153 -----> 0
# 0 ------> Stop the loop

# 123 => "123" # Chnaging the data type
```