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

# a. check if the given numbers are greater or not, also
# 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 [8]:
```
# Prime Number :

# -> A whole number greater than 1 that can't be divided by any other whole number other than 1 and itself.

# 13 -----> 2,3,4,5,6,7,8,9,10,11,12 ------->range(2,num)
# 5 ------> 2,3,4 --------> range(2,num)
# 14 -----> 2,3,4,5,6,7,8,9,10,11,12,13  -------> range(2,num)

num = int(input("Enter a number to check if its a Prime number or not :"))
if(num > 1):
    for i in range(2,num):
        if(num % i == 0):
            print("Its not a Prime Number")
            break
    else:
        print("Its a Prime Number")
else:
    print("Its not a Prime Number")
```

```
Enter a number to check if its a Prime number or not :14
Its not a Prime Number
```

In [19]:
```
#  Create a fibonacci sequence using python if-else statements for n terms.

# 0,1,1,2,3,5,8,...........

nterms = int(input("Enter the number of terms :"))
n1 = 0
n2 = 1
count = 0

if(nterms <= 0):
    print("ERROR : You are giving a wrong input, Please enter a positive number greater than zero")
else:
    while(count <= nterms):
        print(n1)
        nth = n1 + n2
        n1 = n2
        n2 = nth
        count = count + 1

# nterms = 5
# while    print   nth    n1   n2    count
#  True    0        1     1    1       1
#  True    1        2     1    2       2
#  True    1        3     2    3       3
#  True    2        5     3    5       4
#  True    3        8     5    8       5
#  True    5        13    8    13      6
#  False--------> Exit Condition

# edge cases
```

```
Enter the number of terms :-7
ERROR : You are giving a wrong input, Please enter a positive number greater than zero
```

In [ ]:
```
# Create a nested dictionary with values as a nested list for each key in the dictionary

d2 = {"Names": ["Harry", "Avinash", "Adi"], "Post" : ["CEO","Trainer","Intern"]}
print(d2)
```

In [21]:
```
# Create two sets and perform the following:

# a. Union of the two sets
# b. Intersection of the two sets

set1 = {2,3,4,5,6,"Avinash",45.35}
set2 = {1,2,3,4,"Python", "Try"}

print(set1)
print(set2)
```

```
{2, 3, 4, 5, 6, 'Avinash', 45.35}
{1, 2, 3, 4, 'Python', 'Try'}
```

In [22]:
```
# All the elements
set1.union(set2)
```

Out[22]:
```
{1, 2, 3, 4, 45.35, 5, 6, 'Avinash', 'Python', 'Try'}
```

In [23]:
```
# Common elements between both the sets
set1.intersection(set2)
```

```
Out[23]: {2, 3, 4}
```

```
In [30]: # Create a nested tuple from the dictionary, with each item in the tuple as a key value pair from the dictionary.

         d1 = {'k1' : 23, 'k2': "Avinash", 'k3' : True}
         print(d1)

         # items(): return all the key,value pair of a dictionary in a tuple
         d1.items()

         tuple(d1.items())
```

```
         {'k1': 23, 'k2': 'Avinash', 'k3': True}
Out[30]: (('k1', 23), ('k2', 'Avinash'), ('k3', True))
```

```
In [39]: # Create a list of the first 50 even natural numbers, and perform the following operations.

         # a. Count the number of elements in the list.
         # b. Reverse the sequence of the list.
         # c. Sort the list in ascending and descending order.
         # d. Get the index value for the element 44, and update the element with the number 100.
         # e. Return a copy of the list, with the resulting list containing the square of each element.

         l1=[]
         for i in range(1,101):
             if(i%2 == 0):
                 l1.append(i)
         print(l1)
```

```
         [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 7
         2, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

```
In [40]: len(l1)
```

```
Out[40]: 50
```

```
In [43]: # variable_name[start_index : end_index + 1 : steps]

         # [::-1] : start from the end of the list, till the 0 index element and move at -1 step

         reversed = l1[::-1]
         print(reversed)
```

```
[100, 98, 96, 94, 92, 90, 88, 86, 84, 82, 80, 78, 76, 74, 72, 70, 68, 66, 64, 62, 60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 3
2, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]
```

In [44]:
```
# By default it sorts it in ascending order
ascending = sorted(l1)
print(ascending)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 7
2, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

In [45]:
```
# reverse = True is used with the sorted() in order to reverse the list
descending = sorted(l1, reverse = True)
print(descending)
```

```
[100, 98, 96, 94, 92, 90, 88, 86, 84, 82, 80, 78, 76, 74, 72, 70, 68, 66, 64, 62, 60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 3
2, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]
```

In [46]:
```
# index() : Return the index number of a element.
l1.index(44)
```

Out[46]:
```
21
```

In [47]:
```
l1[21] = 100
print(l1)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 100, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 7
2, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

In [48]:
```
l2 = []
for i in l1:
    l2.append(i*i)

print(l2)
```

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 900, 1024, 1156, 1296, 1444, 1600, 1764, 10000, 2116, 2304, 2500, 2704, 2
916, 3136, 3364, 3600, 3844, 4096, 4356, 4624, 4900, 5184, 5476, 5776, 6084, 6400, 6724, 7056, 7396, 7744, 8100, 8464, 8836, 9216, 9604, 10
000]
```

In [ ]:
```
# Create a nested dictionary and perform the following operations.

# a. Return a list with the key value pairs from the dictionary.
# b. Return a list with just the keys from the dictionary.
# c. Remove the first and last key value from the dictionary.
# d. Update the last key value pair in the dictionary after removing in the 3rd step.
```

In [9]:
```
d1 = {"key1": {"key1a" : "Value1"},
      "key2": {"key2a" : "Value2"},
      "key3": {"key3a" : "Value3"}}
print(d1)
```

```
{'key1': {'key1a': 'Value1'}, 'key2': {'key2a': 'Value2'}, 'key3': {'key3a': 'Value3'}}
```

In [10]:
```
list(d1.items())
```

Out[10]:
```
[('key1', {'key1a': 'Value1'}),
 ('key2', {'key2a': 'Value2'}),
 ('key3', {'key3a': 'Value3'})]
```

In [11]:
```
d1key = list(d1.keys())
print(d1key)
```

```
['key1', 'key2', 'key3']
```

In [12]:
```
d1.pop(d1key[0])
print(d1)
```

```
{'key2': {'key2a': 'Value2'}, 'key3': {'key3a': 'Value3'}}
```

In [13]:
```
d1.pop(d1key[-1])
print(d1)
```

```
{'key2': {'key2a': 'Value2'}}
```

In [14]:
```
d1['key2'] = {"NewKey" : "Avinash"}
print(d1)
```

```
{'key2': {'NewKey': 'Avinash'}}
```

In [15]:
```
# For the given strings A = “Python Programming Language”, B = “Best in the World” , perform the following operations.

# a. Using indexing operations, get the text “gram” from the string A.
# b. Using indexing operations, get the text “World” from the string B.
# c. Change the letters in both strings to Uppercase letters.
# d. Concatenate the two strings.

A = "Python Programming Language"
B = "Best in the World"
```

In [16]:
```
A[10:14]
```

Out[16]:
```
'gram'
```

```
In [17]: B[12:17]

Out[17]: 'World'
```

```
In [18]: A.upper()

Out[18]: 'PYTHON PROGRAMMING LANGUAGE'
```

```
In [19]: B.upper()

Out[19]: 'BEST IN THE WORLD'
```

```
In [22]: print(A + " " + B)

Python Programming Language Best in the World
```

```
In [26]: # Create a list with n numbers, and using negative indexing return the list starting from the 5th index to the n-2th index.

l2 = []
n = int(input("Enter the value of n"))
for i in range(n):
    l2.append(i)

print(l2)

Enter the value of n10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [29]: # Using range(), create a list with numbers ranging from 5-100, and the number of elements should be exactly 19.

l3 = []
for i in range(5,100,5):
    l3.append(i)

print(l3)
print(len(l3))

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
19
```

```
In [ ]:
```