INDEX :

Importing Source Code :

```
def greet(name) :
    print(f"Good morning, {name}")

print(__name__)  #-> Prints the name of source file, if this file is imported in another file

if __name__ == "__main__" :  # -> after this no any data going to print/use in another file.
    n = input("Enter your name here :  ")
    greet(n)
```

Below is Another Source Code :

```
import V_ImportFileOfCode
V_ImportFileOfCode.greet("Aman")
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\V_ImportFileOfCode.py"
__main__
Enter your name here :  Aman Verma
Good morning, Aman Verma
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\VV__ImportFileOfCode.py"
V_ImportFileOfCode
Good morning, Aman
PS D:\Tutorial Of Python>
```

Enumerate Function And List Comprehension :

```python
# Enumerate function -> This function adds counter to an iterable and returns it.
list1 = [1, 2, 3, False, 6.2, "It's me"]

for index, item in enumerate(list1) :
    print(index, item)

# List Comprehension
list2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# list3 = []
# for item in list2 :
#     if item%2 == 0 :
#         list3.append(item)

# print("Printing the elements in list3 after performing operation :  ", list3)

# For same logic that we use in above line can handle by 'list comprehension' as follow :-
list3 = [i for i in list2 if i%2 == 0]
print("Printing the elements in list3 by using 'list comprehension' :  ", list3)

list4 = [1, 2, 3, 2, 1, 4, 5, 4, 2]
set1 = {i for i in list4}  # -> take the data and convert it into set.
print(f"Printing the elements in the set1 :  {set1}")
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\W_EnumerateFunction.py"
0 1
1 2
2 3
3 False
4 6.2
5 It's me
Printing the elements in list3 by using 'list comprehension' :  [2, 4, 6, 8, 10]
Printing the elements in the set1 :  {1, 2, 3, 4, 5}
PS D:\Tutorial Of Python>
```

Lambda, Join And Format :

```
# Lambda function ->
func1 = lambda a : a+5
x = 123
print("Printing the value by the use of lambda function :  ", func1(x))

func2 = lambda a, b, c : a+b+c
print(f"Printing the value after performing operation in lambda function
:  {func2(1, 2, 3)}")

# Join function ->
list = ["Camera", "Laptop", "Phone", "iPad", "Hard Disk", "Nvidia Graphic 3080
card"]

sentence = " and ".join(list)  # -> type of 'sentence' is 'string'
# sentence = " ~ ".join(list)  # -> we can give any thing in string to print b/w
values of list.
print(f"Printing sentence :  {sentence}")
print("Printing the type of 'sentence' :  ", type(sentence))

# Format keyword -> before fString format is used to sink data in string.
name = "It's me"
channel = "MyChannel"
type = "Coding"
# data = "Here is :  {}".format(name)
# data = "Here is :  {} and his channel is {}".format(name, channel)
a = "type of channel is :  {2} and his channel is {1} and Here is
:  {0}".format(name, channel, type)
print(a)
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\X_Lambda_Join_Format.py"
Printing the value by the use of lambda function :  128
Printing the value after performing operation in lambda function :  6
Printing sentence :  Camera and Laptop and Phone and iPad and Hard Disk and Nvidia Graphic 3080 card
Printing the type of 'sentence' :  <class 'str'>
type of channel is :  Coding and his channel is MyChannel and Here is :  It's me
PS D:\Tutorial Of Python>
```

Map, Filter And Reduce :

```
from functools import reduce  # -> we need to import it to use 'Reduce'

# Map -> Maps applies a function to all the items in an input_list.
# Syntax : map(function, input_list)
# list(map(function, input_list)) -> type casting into list and we can type cast
into any format.

def square(num) :
    return num*num

l1 = [1, 2, 4, 5]

# Method 1 to get the list of square of elements in the list l1
l2 = []
for i in l1 :
    l2.append(square(i))

print(f"Printing the list of square by the normal logic :  {l2}")

# Method 2 for the same as above ->
print(f"Printing the list of square by the use of 'map' :  {list(map(square,
l1))}")


# Filter
def greater_than_5(num) :
    if num>5 :
        return True
    else :
        return False

l3 = [1, 2, 5, 6, 3, 9, 8, 55, 44, 2, 3, 96]
print("Printing the list of elements greater than 5 :  ",
list(filter(greater_than_5, l3)))
```

```python
# We can use 'lambda' function also.
print("We can also use 'lambda' as a function in 'filter' :  ", list(filter(lambda
num : num>10, l3)))


# Reduce -> Reduce applies a rolling computation to sequential pair of elements.
# We need to import reduce form functools.
sum = lambda a, b : a+b
l4 = [1, 2, 3, 4]
val = reduce(sum, l4)
print(f"Printing the value after using 'reduce' function :  {val}")

# For normal web dev learn 'flask' on you tube by code with harry.
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\Y_Map_Filter_Reduce.py"
Printing the list of square by the normal logic :  [1, 4, 16, 25]
Printing the list of square by the use of 'map' :  [1, 4, 16, 25]
Printing the list of elements greater than 5 :   [6, 9, 8, 55, 44, 96]
We can also use 'lambda' as a function in 'filter' :   [55, 44, 96]
Printing the value after using 'reduce' function :  10
PS D:\Tutorial Of Python>
```