Programming using Python – I

Syllabus Overview

1. Core Python
2. Python GUI
3. Django
4. Flask

Core Python

Python used in :-

- AI and machine learning
- Data analytics and visualisation
- Web and game development
- SEO
- Design
- Etc.

1

Map

Map in Python is a function that works as an iterator to return a result after applying a function to every item of an iterable (tuple, lists, etc.). It is used when you want to apply a single transformation function to all the iterable elements.

Syntax of Map :

map(function, iterables)

1. **function**: It is the transformation function through which all the items of the iterable will be passed.
2. **iterables**: It is the iterable (sequence, collection like list or tuple) that you want to map.

Example 1:

```
def multiply(i):
    return i * i
# Using the map function
x = map(multiply, (3, 5, 7, 11, 13))
print(list(x))
```

OUTPUT : [9,25,49,121,169]

Map and for loop difference

```
num = [3, 5, 7, 11, 13]

mul = []

for n in num:

    mul.append(n ** 2)

print (mul)
```

```
def mul(i):

    return i * i

num = (3, 5, 7, 11, 13)

resu = map(mul, num)

print(resu)

mul_output = list(resu)

print(mul_output)
```

Example 2: Map with built in function

```
example = ["Welcome", "to", "Sem 3"]

x = list(map(len, example))

print(x)
```

OUTPUT : [7,2,4]

Example 3: Map with lambda

```
num = (6, 9, 21, 44)

resu = map(lambda i: i + i, num)

print(list(resu))
```

OUTPUT : [12,18,42,88]

Example 4: Map with tuple

```
def example(s):
    return s.upper()
tuple_exm = ('this','is','map')
upd_tup = map(example, tuple_exm)
print(tuple(upd_tup))
```

OUTPUT : (‘THIS’,’IS,’MAP’)

Example 5: Map with dictionary

```
car_dict ={'a': 'Mercedes-Benz', 'b': 'BMW', 'c': 'Ferrari', 'd': 'Lamborghini', 'e': 'Jeep'}

# adding an '_' to the end of each value
car_dict = dict(map(lambda x: (x[0], x[1] + '_'), car_dict.items() ))

print('The modified dictionary is : ')

print(car_dict)
```

Example 6: Map with set

```
def example(i):
    return i%3
set_exm = {33, 102, 62, 96, 44, 28, 227}
upd_itms = map(example, set_exm) # divisible by 3 or not
print(set(upd_itms))
```

Assignment

1. Triple all the numbers given in list
2. Separate even odd number from given list
3. Print all string in lower case from given tuple
4. Print square root of numbers given in list
5. Eliminate duplicate letter from given string
6. Print table of any number
7. Add two list
8. Convert all float digits into integer using built in function from given list.
9. Reverse given set
10. Add ‘@gmail.com’ to all the values of given dictionary and convert it to lower case.

2

Filters

The filter() function extracts elements from an iterable (list, tuple etc.) for which a function returns True.

Syntax of Filter :

map(function, iterables)

1. **function**
2. **iterables**: It is the iterable (sequence, collection like list or tuple) that you want to map.

Example 1:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# returns True if number is even
def check_even(number):
    if number % 2 == 0:
        return True
    return False

even_numbers_iterator = filter(check_even, numbers)

# converting to list
even_numbers = list(even_numbers_iterator)
print(even_numbers)

# Output: [2, 4, 6, 8, 10]
```

Example 2:

```
letters = ['a', 'b', 'd', 'e', 'i', 'j', 'o']

def filter_vowels(letter):
    vowels = ['a', 'e', 'i', 'o', 'u']
    return True if letter in vowels else False

filtered_vowels = filter(filter_vowels, letters)

vowels = tuple(filtered_vowels)
print(vowels)
```

Example 3: Using lambda

```
numbers = [1, 2, 3, 4, 5, 6, 7]

even_numbers_iterator = filter(lambda x: (x%2 == 0), numbers)

even_numbers = list(even_numbers_iterator)

print(even_numbers)
```

Assignment

1. Using filter() function filter the list so that only negative numbers are left.
2. Using filter function, filter the even numbers so that only odd numbers are passed to the new list.
3. Using filter() and list() functions and .lower() method filter all the vowels in a given string.
4. This time using filter() and list() functions filter all the positive integers in the string.
5. Convert a number to positive if it's negative in the list. Only pass those that are converted from negative to positive to the new list.[Try using map inside filter]

3

Exception Handling

When a Python program meets an error, it stops the execution of the rest of the program. An error in Python might be either an error in the syntax of an expression or a Python exception. Using try,catch we can handle them.

Syntax:

try:

Code block

These statements are those which can probably have some error

except:

This block is optional.

If the try block encounters an exception, this block will handle it.

else:

If there is no exception, this code block will be executed by the Python interpreter

finally:

Python interpreter will always execute this code.

Example:

```
try:
    print('try block')
    x = int(input("Enter number"))
    y = int(input("Enter another number"))
    z = x/y
except ZeroDivisionError:
    print("except ZeroDivisionError block")
    print("Division by 0 not accepted")
else:
    print("else block")
    print("Division = ", z)
finally:
    print("finally block")
    x=0
    y=0
print ("Out of try, except, else and finally blocks." )
```

Example : Raise an Exception

```
try:
    x=int(input('Enter a number upto 100: '))
    if x > 100:
        raise ValueError(x)
except ValueError:
    print(x, "is out of allowed range")
else:
    print(x, "is within the allowed range")
```

Assignment

Below, we have buggy code. Add a try/except clause so the code runs without errors. If a blog post didn’t get any likes, a ‘Likes’ key should be added to that dictionary with a value of 0.

CODE :

```
blog_posts = [{'Photos': 3, 'Likes': 21, 'Comments': 2}, {'Likes': 13, 'Comments': 2, 'Shares': 1}, {'Photos': 5, 'Likes': 33, 'Comments': 8, 'Shares': 3}, {'Comments': 4, 'Shares': 2}, {'Photos': 8, 'Comments': 1, 'Shares': 1}, {'Photos': 3, 'Likes': 19, 'Comments': 3}]

total_likes = 0

for post in blog_posts:
    total_likes = total_likes + post['Likes']
```