Comprehensions

Session 06

Session Overview

In this session, you will be able to:

- Describe list comprehensions
- Explain how to use list comprehensions, such as lambda, Map, Reduce, and Filter on files

List Comprehensions (LCs) (1-5)

Expression-oriented functions in Python are:

- `list comprehension`
- `lambda`
- `map`
- `filter`
- `reduce`

List Comprehensions (LCs) (2-5)

Create a matrix using lists:

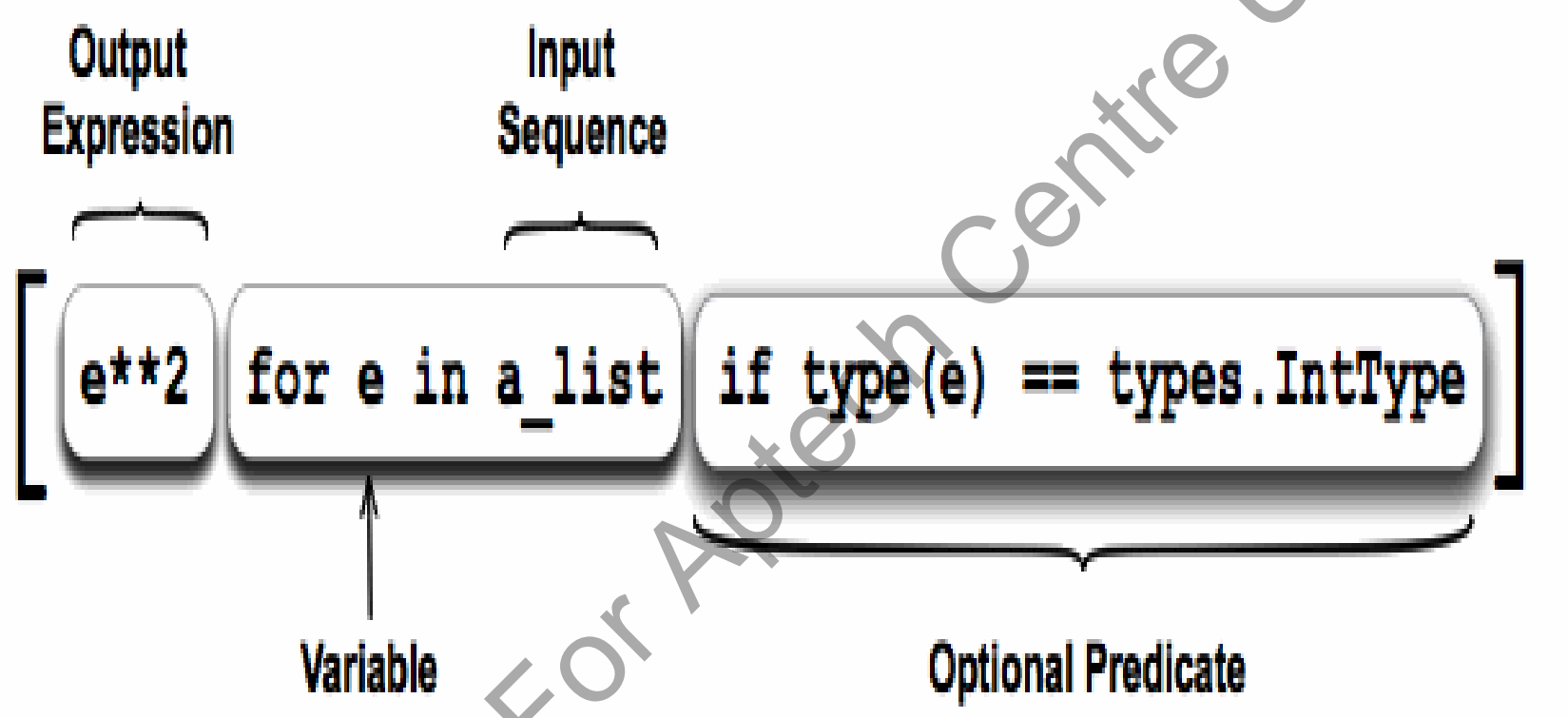
```
>>> M = [[1, 2, 3], # A 3 × 3 matrix, as nested lists
         [4, 5, 6], # Code can span lines if bracketed
         [7, 8, 9]] #To get a column from the matrix
>>> col2 = [row[1] for row in M]
# Collect the items in column 2
>>> col2
```

Output:

```
[2, 5, 8]
```

List Comprehensions (LCs) (3-5)

Working of the LC:



List Comprehensions (LCs) (4-5)

Following are the ways in which LCs work:

- For a given input sequence `a_list`, the iterator bit (`for e in a_list`) iterates through each member e.
- The predicate is responsible for checking if the member is an integer.
- In case the member is an integer, it is delivered to the output expression, where it is squared in order to be considered as a member of the output list.

List Comprehensions (LCs) (5-5)

Aspects of LCs to consider:

- It always gives a result, irrespective of whether it is used or not.
- It is possible to nest iteration and conditional expressions with many instances.
- It is also possible to nest overall LCs within another LCs.
- It allows users to iterate and manipulate many variables simultaneously.

Using LCs on Files

LCs help to navigate through a folder and obtain the type of files:

Code Snippet:

```
>>> import os
>>> files = []
>>> for file in
os.listdir('/my_dir'):
        if file.endswith('.txt'):
          files.append(file)
>>> print (files)
```

Output:

```
['data.txt',
'data_new.txt',
'datafile.txt',
'json.txt']
```

Reading a CSV file into Dictionary (1-3)

- CSV stands for comma separated values file that stores data in a table structured format.
- CSVs appear almost similar to a garden-variety spreadsheet, but it has a .csv extension.
- Traditionally, they are similar to a text file having data separated by commas, thus, the name.
- There is a repeated request to read data from a csv file and process the same.
- One of the most helpful approach to process csv data is to change it into a list of dictionaries.

Reading a CSV file into Dictionary (2-3)

Create a CSV file:

Code Snippet:

```
>>> import csv
>>> data = []
>>> for x in csv.DictReader(open('file.csv',
'rU')):
                        data.append(x)
>>> print (data)
```

Reading a CSV file into Dictionary (3-3)

Sample Output:

```
[OrderedDict([('1', '2'), ('Eldon Base for stackable
storage shelf, platinum', '1.7 Cubic Foot Compact "Cube"
Office Refrigerators'), ('Muhammed MacIntyre', 'Barry
French'), ('3', '293'), ('-213.25', '457.81'), ('38.94',
'208.16'), ('35', '68.02'), ('Nunavut', 'Nunavut'),
('Storage &
Organization', 'Appliances'), ('0.8', '0.58')]),
OrderedDict([('1', '3'), ('Eldon Base for stackable
storage shelf, platinum', 'Cardinal Slant-D Ring Binder,
Heavy Gauge Vinyl'), ('Muhammed MacIntyre', 'Barry
French'), ('3', '293'), ('-213.25', '46.71'), ('38.94',
'8.69'), ('35', '2.99'),('Nunavut', 'Nunavut'),
('Storage & Organization', 'Binders and Binder
Accessories'), ('0.8', '0.39')])]
```

Set and Dictionary Comprehensions (1-3)

- Consider there is a list of names. The list can have names differing only in the case used to represent them, duplicates and names having only one character.
- Assume a user wants the list of names longer than one character and prefers to use the same format for all the names.
- It can be achieved by using the list as follows: `[ 'Sam', 'Paul', 'Nemo', 'sam', 'PAUL', 'J', 'memo' ]`
- User needs the list in the format: `{ 'Sam', 'Paul', 'Nemo', 'Memo' }`

Set and Dictionary Comprehensions (2-3)

How set comprehension is accomplished?

Code Snippet:

```
>>> names = [ 'Sam', 'Paul', 'Nemo', 'sam',
'PAUL', 'J', 'memo' ]
>>> result_set = { name[0].upper() +
name[1:].lower() for nam in names if
len(nam) > 1 }
>>> print (result_set)
```

Output:

```
{ 'Sam', 'Paul', 'Nemo', 'Memo' }
```

Set and Dictionary Comprehensions (3-3)

Total count of character instances:

Code Snippet:

```
>>> test_dic = {'l':10, 'b': 34, 'Z': 2,
'N':4, 'L':4, 'z':5}
>>> mcase_frequency = { k.lower() :
test_dic.get(k.lower(), 0) +
test_dic.get(k.upper(), 0) for k in
test_dic.keys()}
>>> mcase_frequency
```

Output:

```
{'l': 14, 'b': 34, 'z': 7, 'n': 4}
```

lambda, map, reduce, and filter (1-3)

lambda:

lambda is not a statement, it is considered as an expression. Due to this, lambda can appear in locations that a def is not permitted.

lambda's body is considered as a single expression and not as a block of statements. Its body is similar to what is generally given as a def body's return statement.

lambda, map, reduce, and filter (2-3)

map:

The map function applies a passed-in function to each item in an iterable object and generates a list with all the function call results:

Code Snippet:

```
>>> items = [2, 1, 3, 4]
>>> def sqrmaker(x): return x **
2
>>> list(map(sqrmaker, items))
```

```
Output:
[4, 1, 9, 16]
```

lambda, map, reduce, and filter (3-3)

filter and reduce:

The filter function is used to extract each element in a given sequence for which the function returns True.

The reduce function is used to reduce a list to a single value by combining elements through a provided function.

The filter and reduce are used to apply functions to sequences and other iterable.

The filter function filters elements depending on a test function and applies functions to pairs of elements.

Summary (1-4)

- Functional programming is all about expressions, that is, it is an expression-oriented programming.
- Expression-oriented functions in Python are `list comprehension`, `lambda`, `map`, `filter`, and `reduce`.
- `List Comprehension` (LC) is a powerful way to process structures, such as matrix. It is an easy and effective technique and can help users to complete a range of tasks easily.

Summary (2-4)

- LCs are also useful when working on files. They help to navigate through a folder and obtain the type of files.
- CSV stands for comma separated values file that stores data in a table structured format. CSVs appear almost similar to a garden-variety spreadsheet but have a .csv extension.
- Set comprehensions are similar to LCs. They allow sets to be created using the same principles as LCs, but the output sequence will be in the form of a set.

Summary (3-4)

- `lambda` operator or `lambda` function helps to create small, one-time, and anonymous function objects in Python.
- The `map` function applies a passed-in function to each item in an iterable object and generates a list with all the function call results.
- The `filter` function is used to extract each element in a given sequence for which the function returns True.

Summary (4-4)

- The reduce function is used to reduce a list to a single value by combining elements through a provided function.