Python Programming

L04: Lists

March 2020

Dr. Ram P Rustagi
Professor, CSE Dept
KRP, KSGI
rprustagi@ksit.edu.in

Resources and Acknowledgements

- Intro to Programming with C++
  - Abhiram Ranade, Prof CSE, IIT Bombay
- A first course in programming
  - https://introcs.cs.princeton.edu/python/home/
  - https://introcs.cs.princeton.edu/java/home/
- Python for everybody
  - https://www.py4e.com
- Web Applications for everybody
  - https://www.wa4e.com
- https://education.pythoninstitute.org/course_datas
- https://www.w3schools.com/python/
  - Basic Python Tutorial

Overview

- Overview of Collections
- Collection operations
- Lists
- Exercises
- Summary

Collections

- Why collections?
  - variables store only 1 value, e.g
    - a=10
    - b=10.1
- Most applications need multiple data items to store in a variable
  - Such an entity is called collection
  - example: storing N prime numbers
    - e.g. 2, 3, 5,7, 11, 13, 17, 19, 23, 29
- Collections are of various kinds
  - Arrays, Lists, Stacks, Queues, Dictionaries etc
- We will look at lists
  - An **ordered** collection of elements

Lists

- Creating a List of vowels
  ```
  vowels = ['A',  'E', 'I'. 'O', 'U']
  ```
- Initializing an empty List of size `n`
  ```
  arr = [None] * n
  ```
- Extending an List by size `m`
  ```
  arr = arr + [None] * m
  ```
- Accessing elements
  - Index starts from 0
    - follows machine language programming convention
  - Index of last element is `n-1`
  - Negative index
    - `-i` implies `len(array) - i`
- `IndexError` on array bounds violation

Index Starts From 0

P1	P2	P3	P4	P5	P6	P7	P8

- Consider 8 partking lots of rental cars
- Each parking lot has a car parked in it
  - Car can be accessed at the start of parking lot
- Each parking lot width is 3 meters
- Attendant is at the beginning of the parking lot
- He needs to access car in `k`th parking lot
  - How much he needs to walk:
    - `3k` or `3(k-1)` meters
- For first car, he needs to walk 0 meters
- Computers implement access to memory same way
  - `1`st index starts from 0

Lists

- List has fixed size
  - Can be modified by adding/remove elements
  - All elements need not be of same type
  - e.g. `xyz=['A', 90, 'B', 80, 'C', 70]`
- Length of a list
  - `len(xyz)`
- Mutability
  - Any element value can be modified

    ```
    arr = [None] * 10
    arr[1] = 1
    ```
  - Array can be resized

    ```
    arr1 = [1,2]; arr2 = [3,4]
    ```
  - Adding another list or element

    ```
    arr = arr.extend(arr2)
    ```

List Traversal & Operations

- `students=["SS", "AT", "AS","SN","VA"]`
- Traversing a list

```
for student in students:
    print(student)

for i in range(len(students)):
    print (students[i])
```

- Concatenating two lists

```
phase1 = ['A', 'B', 'C', 'D', 'E', 'F']
phase2 = ['G', 'H', 'I', 'J', 'K']
tranquil = phase1 + phase2
print(tranquil)
```

Lists Operations

- Sub lists, slicing
  - Always provides a new List object

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

arr[2:6] # [3, 4 ,5]
arr[:]   # same list
arr[:-1] # excludes the last element
arr[-4:-5] # gives empty list
```

  - picking elements at an interval

```
arr[1::3] # [2, 5, 8]
arr[::-1] # reverses the list
```

List Operations

- Operations that work on the same object
  - `append`(elem)
  - `clear`()
  - `count`(arr)
  - `extend`(arr)
  - `index`(elem)
  - `insert`(index, elem)
  - `pop`() , `pop`(index)
  - `remove`(elem)
  - `reverse`()
  - `sort`()

List Operations

- Operations that work on the same object
  - Assignment operation

```
lectures = ['L1', 'L2', 'L3']
updated = lectures # same object.
```

    - any operation affects both

```
updated.append('L4')
print(lectures)
```

      - ['L1', 'L2', 'L3', 'L4']

- Check if something is in list

```
if 'L2' in lectures:
  print('found')
if 'L5' not in lectures:
  print('not found')
```

List Operations

- Operations that return a new list object
  - List slicing e.g. arr[m:n]
  - `copy`() # returns a new copy of array
- Operations return a value
  - `index`(elem, [start, [end]])
  - `count`()

Built-In Functions on List

```
lectures = ['L1', 'L2', 'L3']
```

- print(lectures)
- max(lectures)
- min(lectures)

- ages=[61,17,18,16,20,61]
- sum(ages)
- min(ages)
- max(ages)
- sum(ages)/len(ages) # gets average

Strings

- Strings are lists with some additional functions and characteristics

```
name='python'
len(name)
name[2]
name[3:5]
name[::-1]
name[2::2]
name[2::-1]
names='abhiraj srujna aditya anuj vedant'
names.split('a')
x=names.split()
x[1].split('u')
```

Strings

- Strings are immutable

```
name='python'
name[0]='P' # gives error
```

- Concatenation

```
course = 'python' + 'programming'
```

- Multiply string by a number

```
name * 3 #gives 'pythonpythonpython'
```

- Iterating over a string

```
for letter in name:
  print(letter)
for index in range(len(name)):
  print(letter[index])
```

Strings Operations

- Comparing two string

```
name='python'
lang='python'
lang2='Python'
name == lang # True
name == lang2 # False
```

- Case conversion: returns new string

```
name.upper()
lang2.lower()
name.capitalize()
```

- Get list of all methods on string

```
dir(name)
```

Strings Operations

- Stripping spaces at beginning and end

```
name=' python  '
name.strip()
name.lstrip()
name.rstrip()
```

- Replacing a part of string

```
name.replace('py', 'Py')
name.replace('on', 'language')
```

- Get prefix and suffix

```
name.startswith('py')
name.endswitdh(‘language')
```

Programming Exercises:

- Ex 01: Deck of Playing Cards
  - Create a deck of cards
    - Rank: `2,3,4,5,6,7,8,9,10, J,Q,K,A`
    - Suite: `Club, Diamond, Heart, Spade`
  - Pick 2 random cards with replacing.
    - Find the probabilities of picking same suite card
      - By conducting 100000 trials
  - pick 4 random cards without replacing
    - Count the number of trials till you get the cards of same rank
- Note: to pick a random number between 1 and 10

```
import random
random.randrange(1,11)
```

Programming Exercises:

- Ex 02: Birthday Probability
  - Let students enter a class one by one.
  - The entry stops the moment the birthday of new incoming student is same as one of those already in the class.
    - Birthday can be taken as a number between `1` & `365`
    - Conduct 10000 trials and find the average class size when two students have same birthday.
- Note: to pick a random number between 1 and 10

```
import random
random.randrange(1,11)
```

Programming Exercises:

- Ex 03: Finding a duplicate
  - Given an array of numbers
    - Elements of array have value less than array size
  - Find if a duplicate number exists
  - Do not create a new array
    - You can not modify the elements of the array
  - Do not use a dictionary object
  - Do not check membership existence of element
    - e.g. do not use `if elem in arr:`
- Simple implementation in $O(n^2)$ time
- Can you find an efficient implementation: $O(n)$

Programming Exercises:

- Ex 04a: create multiple lists
  - Tranquil as a list of 11 towers
  - Each tower as a list of 14 floors
  - Each floor as a list of 8 apts
  - Iterate over these 3 lists and
    - Display apartment numbers
- Ex 04b: List of lists
  - Tranquil as a list of 11 towers
  - Each tower as a list of 14 floors
  - Each floor as a list of 8 apts
  - Iterate over these list of lists of lists and
    - Display apartment numbers

Questions