# Python Programming

# L06: Arrays

Aug/Sep, 2019

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 Arrays
- Array operations
- Multi dimentional arrays
- Exercises
- Summary

# Arrays

- Creating an array
  ```
  vowels = [‘A’,  ‘E’, ‘I’. ‘O’, ‘U’]
  ```
- Initializing an array with size `n`
  ```
  arr = [None] * n
  ```
- Extending an array 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

## Arrays

- Array implementation
  - Implemented as **List** object
- Mutability
  - Any element value can be modified

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

    ```
    arr1 = [1,2]; arr2 = [3,4]
    arr = arr.extend(arr2)
    arr.append(5)
    ```

## Array Operations

- Sub arrays, slicing
  - always provides a new array object

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

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

  - picking elements at an interval

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

# Array Operations

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

    `arr = [1,2,3,4]`

    `narr = arr` # same object.

    - any operation affects both

# Array Operations

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

# Array as Other Data Structure

- Stack:
  - `append()` and `pop()` provides the stack operations
- Queue

```
from collections import deque
q = deque([1,2,3,4,5])
a.popleft()
a.pop()
a.append()
```

# Multi Dimensional Array

- By default list is one dimensional
  - Each element can be an array itself
  - Each element array need not be same size e,g.

```
N=10
arr = [None] * N
for i in range(N):
  arr[i] = [None]* i
```

- For matrix, each element should be same size, e.g.

```
N=10
arr = [None] * N
for i in range(N):
  arr[i] = [None]* N
```

## Module `numpy`

- Language design has a tradeoff
  - Between simplicity and efficiency
  - List data structure has simplicity in design
    - For large array sizes, it performance is slow
- Module numpy
  - Designed for large array for processing numbers
  - Uses lower level implementation to overcome inefficiencies

# 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 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 04: Finding Longest Plateau
  - Given an array of integers
  - Find the length and location of plateau i.e.
    - Longest contiguous sequence of equal values, where
      - values of elements just before and just after this sequence are smaller.

# Questions