CORVIT
Reaching end to end

Introducing Python

Lecture# 4

by

python™

Umair bin Mansoor

Network Programming Planner

CORVIT
Reaching end to end

python™

OBJECTIVES

After this session, students will be able to:

- Describe the difference between mutable and non-mutable data types
- Define sequences and their types
- Describe why sequences are useful in programming and how to create sequences like list, tuples and strings
- Explore common operations for sequences
- Use the `len`, `min`, `max`, `sum`, and `random.shuffle` functions with a list
- Access sequence elements by using indexed variables
- Obtain a subsequence from a larger sequence by using the slicing operator `[start : end]`
- Use the + (concatenation), * (repetition), and `in/not in` operators on sequences
- Invoke a list's `append`, `count`, `extend`, `index`, `insert`, `pop`, `remove`, `reverse`, and `sort` methods
- Split a string into a list using the str's `split` method

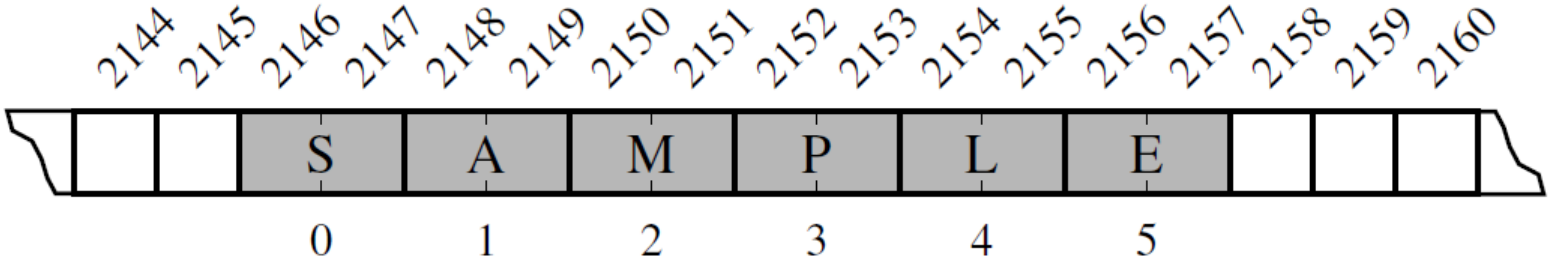
MUTABLE AND NON-MUTABLE OBJECTS

- Objects are divided into two distinct categories: mutable and immutable.
- An immutable object is one in which the state cannot be changed once it has been created.
- If the data fields of the object can be changed after the object has been created, the object is said to be mutable.

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

SEQUENCES

- There are three sequence class types: List, Tuple and String
- The name sequence relates it to the way it is stored in the memory. A sequence is stored in a sequence of memory blocks, one after the other.



- Sequences can be easily accessed via indexes similar to the way the elements of an array are accessed.
- An array has a fixed size. A Python list's size is flexible. It can grow and shrink on demand.

CORVIT
Reaching end to end

python™

LIST OBJECT

- **A list is a sequence defined by the list class. It contains the methods for creating, manipulating, and processing lists. Elements in a list can be accessed through an index.**
- **To create a list, you can use list's constructor, as follows:**
  - `list1 = list()` *# Create an empty list*
  - `list2 = list([2, 3, 4])` *# Create a list with elements 2, 3, 4*
  - `list3 = list(["red", "green", "blue"])` *# Create a list with strings*
  - `list4 = list(range(3, 6))` *# Create a list with elements 3, 4, 5*
  - `list5 = list("abcd")` *# Create a list with characters a, b, c, d*
- **You can also create a list by using the following syntax, which is a little simpler:**
  - `list1 = []` *# Same as list()*
  - `list2 = [2, 3, 4]` *# Same as list([2, 3, 4])*
  - `list3 = ["red", "green"]` *# Same as list(["red", "green"])*
  - `list4 = [2, "three", 4]` *# Mixed data-type elements*

COMMON SEQUENCE OPERATIONS

- **Common operations for sequences are:**

Operation	*Description*
`x in s`	True if element x is in sequence s.
`x not in s`	True if element x is not in sequence s.
`s1 + s2`	Concatenates two sequences s1 and s2.
`s * n, n * s`	n copies of sequence s concatenated.
`s[i]`	ith element in sequence s.
`s[i : j]`	Slice of sequence s from index i to j − 1.
`len(s)`	Length of sequence s, i.e., the number of elements in s.
`min(s)`	Smallest element in sequence s.
`max(s)`	Largest element in sequence s.
`sum(s)`	Sum of all numbers in sequence s.
`for loop`	Traverses elements from left to right in a `for` loop.
`<, <=, >, >=, =, !=`	Compares two sequences.

COMMON LIST OPERATIONS

- Common operations for list are:
  - `append(`*`element`*`)` method adds a new object as an element to the list
  - `extend(`*`index`*`)` method joins a new list to an existing list.
  - `insert(`*`index`*`)` method inserts a new element at a particular *index* location
  - `index(`*`element`*`)` method returns the index location of an *element*
  - `count(`*`element`*`)` method returns the number of times an element is present
  - `pop(`*`index`*`)` method removes and returns the element at the given *index*
  - `remove(`*`element`*`)` method removes the given *element*
  - `reverse()` method reverses all the element of the list
  - `sort()` method sorts the elements in ascending order.
  - `clear()` method removes all the elements making it an empty list
  - `copy()` method makes a copy of the list and save it in a new list
  - `split()` a string method capable of splitting string into elements of a list

Questions & Answers