PRG1

W
E
E
K

3

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Lists

Programming I (PRG1)

Diploma in Information Technology

Diploma in Financial Informatics

Diploma in Information Security & Forensics

Year 1 (2018/19), Semester 1

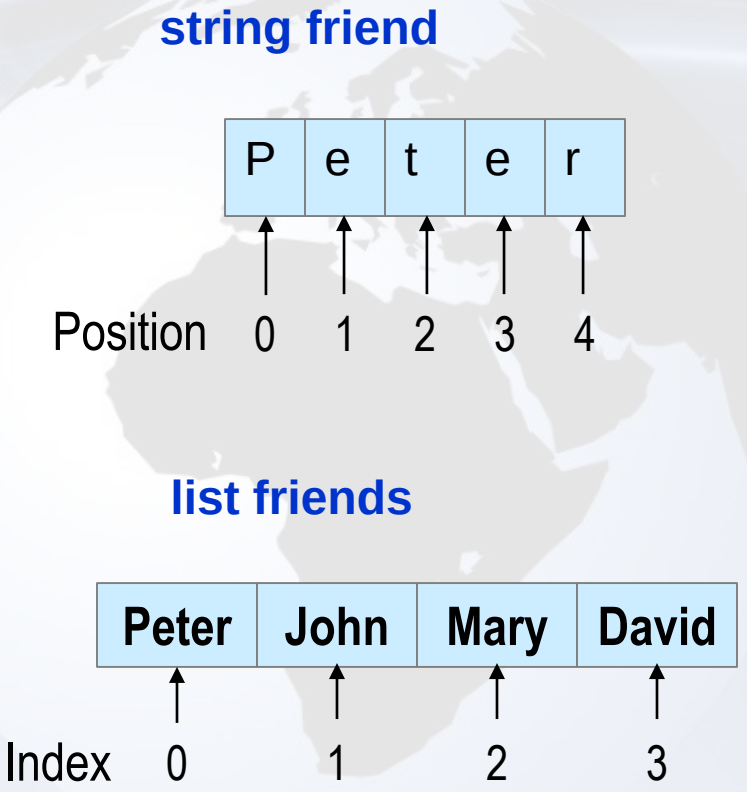
Objectives

At the end of this lecture, you will learn how to:

- Create Lists
- Process Lists using List operators and methods

What is List

- A ***list*** is considered a sequence type in Python, similar to Strings.
- A list is a sequence of values.
- In a string, the values are characters.
- In a list, the values can be any type.
- Values in a list are called ***elements*** or items.



Examples of List

- The names of your friends
- A list of students' marks
- A shopping list
- A recipe – a list of instructions
- A text document – a list of lines

Peter	84.7
David	71
Vincent	55
Hafiz	80
Janet	63
Albert	….
….	

S9099885K;Ang PS;98765432
S9177885P;David Davis;89778899
S9267890A;Lim Vincent;91990099
S9111199Q;Tan SS;88995566
….

Creating a List

- Enclose the elements in square brackets, separate each element by commas.

  E.g. `[ ]`

  `[ 'Peter', 'John', 'Mary', 'David' ]`

  `[ 89, 77, 55, 69 ]`

- Usually we assign the list to a variable name so that we can refer to the list subsequently:

  E.g. `emptyList = [ ]`

  `friendsList = [ 'Peter', 'John', 'Mary', 'David' ]`

  `marksList = [ 89, 77, 55, 69 ]`

Creating a List

- An element in a list can be another list – ***nested list***.

  E.g. friendsList = ['Peter', ['John', 'Mary'], 'David']

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

- A list that contains no elements is called an empty list.

  E.g. emptyList = [ ]

- A list may contain elements of different types.

  E.g. mixedList = ['Peter', 100, 23.5, [10, 20]]

Basic Operators for List

- Use the **bracket operator [n]** to access an element in the list.
- Use the **[n:m] operator** to access part of the list from index n to m.
- Usage is similar to Strings

```
>>> list1 = [1,2,3,4,5]
>>> list1
[1, 2, 3, 4, 5]
```

```
>>> list1[0]
1
>>> list1[1]
2
>>> list1[-1]
5
>>> list1[1:3]
[2, 3]
>>> list1[3:]
[4, 5]
>>> list1[:3]
[1, 2, 3]
```

```
>>> friends = ['Peter', 'John', 'Mary', 'David']
>>> friends[1:3]
['John', 'Mary']
```

Basic Operators for List

- The **+ operator** concatenates lists.

```
>>> list1 = [1,2,3,4,5]
>>> list2 = ['a','b','c']
>>> list1 + list2
[1, 2, 3, 4, 5, 'a', 'b', 'c']
```

- The **in operator** detects the presence of an element in the list.

```
>>> 1 in list1
True
>>> 1 in list2
False
```

- The **== operator** compares the equality of two lists

```
>>> list3 = [1,2,3,4,5]
>>> list1 == list2
False
>>> list1 == list3
True
```

Basic Functions for List

- The function **len()** returns the number of elements in the list.

```
>>> list1 = [1,2,3,4,5]
>>> len(list1)
5
```

- The function **min()** returns the smallest element in the list.

```
>>> min(list1)
1
```

- The function **max()** returns the largest element in the list.

```
>>> max(list1)
5
```

Activity 1

- Create a list called marksList that contains 10 elements.
- Display the value in the first element of marksList.
- Add the values in the last two elements of marksList and assign the result to the variable sum.
- Double the value in the second element of marksList.

Built-in List methods

Methods	Description	Example letters=['a', 'b']
append(x)	Add an element, x, to the end of the list.	letters.append('c') letters ['a', 'b', 'c']
extend(L)	Extend the list by appending all the items in the given list, L.	letters.extend(letters) letters ['a', 'b', 'c', 'a', 'b', 'c']
insert(i, x)	Insert an item, x, before the given position i in the list.	letters.insert(3,'z') letters ['a', 'b', 'c', 'z', 'a', 'b', 'c']

Built-in List methods

Methods	Description	Example letters is the list ['a', 'b', 'c', 'z', 'a', 'b', 'c']
remove(x)	Remove the first item from the list whose value is x. Error occurs if x is not in the list.	**letters.remove('c')** **letters** **['a', 'b', 'z', 'a', 'b', 'c']** **letters.remove('d')** **ValueError:** **list.remove(x): x not in list**
pop([i])	Remove the item at the given position in the list and return it. Removes and returns the last item in the list if the argument is not stated.	**letters.pop(2) ➔ 'z'** **letters** **['a', 'b', 'a', 'b', 'c']** **letters.pop() ➔ 'c'** **letters** **['a', 'b', 'a', 'b']**

Built-in List methods

Methods	Description	Example Letters is the list ['a', 'b', 'a', 'b']
index(x)	Return the index in the list of the first item whose value is x. Error occurs if x is not in the list.	letters.index('a') ➔ 0 letters.index('c') ValueError: 'c' is not in list
count(x)	Return the number of times x appears in the list.	letters.count('a') ➔ 2
reverse()	Reverse the elements of the list in place.	letters.reverse() letters ['b', 'a', 'b', 'a']
sort()	Sort the items of the list in place.	letters.sort() letters ['a', 'a', 'b', 'b']
clear()	Remove all items from the list.	letters.clear()

Try-Outs

- Given that the list firstList and secondList are created as follows:
  - **firstList = [2, 4, 6, 8, 10]**
  - **secondList = [2, 4, 6, 8, 10]**

Evaluate the outputs in the following pages

Evaluate

Program Text	Output
print(firstList[1])	
print(firstList[-1])	
print(firstList[5])	
print(firstList[1:3])	
print(firstList[:3])	
print(firstList[3:])	
print(firstList)	

Evaluate

Program Text	Output
print(len(firstList))	
print(secondList)	
print(firstList == secondList)	
for each in firstList: print(each)	
prifor i in range(len(secondList)): secondList[i] = secondList[i] + 1 print(secondList)	
thirdList = firstList + secondList print(thirdList)	

Evaluate

Program Text	Output
print(5 in firstList) print(5 in secondList)	

fourthList = firstList
for i in range(len(fourthList),2):
  fourthList[i] = fourthList[i] * 2
  print(fourthList)
print(firstList)	

Activity 2

- You will have some practice with processing and manipulating strings and lists, via Coursemology.

Reading Reference

- How to Think Like a Computer Scientist: Learning with Python 3
  - Chapter 11

http://openbookproject.net/thinkcs/python/english3e/index.html

Summary

- Creating Lists
- List operators and methods