COMP 141

Lists

Rhodes College

Announcements

Reminders:

- Program 6 – due tomorrow by 11:55pm
- Midterm 2 on Nov. 8th

Rhodes College

String Methods

Table 9-3 Search and replace methods

Method	Description
`endswith(substring)`	The *substring* argument is a string. The method returns true if the string ends with *substring*.
`find(substring)`	The *substring* argument is a string. The method returns the lowest index in the string where *substring* is found. If *substring* is not found, the method returns −1.
`replace(old, new)`	The *old* and *new* arguments are both strings. The method returns a copy of the string with all instances of *old* replaced by *new*.
`startswith(substring)`	The *substring* argument is a string. The method returns true if the string starts with *substring*.

Rhodes College

Using the `find` method

```
def main():
    filename = "First Last_assignsubmission_file_lastname_firstname_prg6.py"
    print(renameFile(filename))

def renameFile(fileName):
    ind = fileName.find("file_")
    fileName = fileName[ind+5:]
    return fileName

main()
```

Output:

```
lastname_firstname_prg6.py
```

Rhodes College

Testing, Searching, and Manipulating Strings

- **You can use the `in` operator to determine whether one string is contained in another string**
  - General format: *`string1`* `in` *`string2`*
    - *`string1`* and *`string2`* can be string literals or variables referencing strings
- **Similarly you can use the `not in` operator to determine whether one string is not contained in another string**

Rhodes College

Class Practice

- Write a function called `count_unique` that counts the number of unique characters in a string.
  - `count_unique("abracadabra") returns 5.`
- Write a function called `count_dups` that counts the number of back-to-back duplicated characters in a string.
  - `count_dups("balloon") returns 2`

Introduction to Lists

- **List:** an object that contains multiple data items
  - Element: An item in a list
  - Format: *`list`* `= [`*`item1`*`,` *`item2`*`, etc.]`
  - Can hold items of different types
- **`print` function can be used to display an entire list**
- **`list()` function can convert certain types of objects to lists**

Rhodes College

Introduction to Lists

A list of integers

```
even_numbers = [2, 4, 6, 8, 10]
```

A list of strings:

```
names = ['Molly', 'Steven', 'Will', 'Alicia']
```

A list holding different types:

```
info = ['Alicia', 27, 1550.87]
```

Rhodes College

Example Using Lists

```
def main():
    # Create a list with some items.
    food = ['Pizza', 'Burgers', 'Chips']

    # Display the list.
    print('Here are the items in the food list:')
    print(food)


# Call the main function.
main()
```

Program Output

```
Here are the items in the food list:
['Pizza', 'Burgers', 'Chips']
```

Rhodes College

9

Why use lists?

- Lists exist so programmers can store multiple related variables together.
- Useful when we don't know ahead of time how many items we are going to store.
  - Lists solve this problem because a single list can hold from zero to practically any number of items in it.

Basic list operations

- Lists are created using square brackets around items separated by commas.

```
mylist = [1, 2, 3]
numbers = [-9.1, 4.77, 3.14]
fred = ["happy", "fun", "joy"]
```

- Lists are accessed using indices/positions just like strings.
- Most (but not all) string functions also exist for lists.

Strings	Lists
string_var = "abc123"	list_var = [item1, item2, ...]
string_var = ""	list_var = [ ]
len("abc123") len(string_var)	len([3, 5, 7, 9]) len(list_var)
string_var[p] string_var[p:q]	list_var[p] list_var[p:q]
str3 = str1 + str2 str3 = "abc" + "def"	list3 = list1 + list2 list3 = [1, 2, 3] + [4, 5, 6]
"i" in "team" -> False	7 in [2, 4, 6, 8] -> False

One important difference

Strings are *immutable*

- **You can't change a string without making a copy of it.**

```
s = "abc"
s[0] = "A"       # illegal!
s = "A" + s[1:] # legal
```

Lists are *mutable*

- **Can be changed "in-place" (without explicit copying)**

```
L = [2, 4, 6, 8, 10]
L[0] = 15        # legal
L.append(26)     # legal
```

13

Compare Immutable and Mutable

- **How can we switch the first and last letter in a string?**
- **How can we switch the first and last items in a list?**

Rhodes College

14

Three common ways to make a list

- Make a list that already has stuff in it:
  `lst = [4, 7, 3, 8]`
- Make a list of a certain length that has the same element in all positions:
  `lst = [0] * 4  #makes the list [0,0,0,0]`
  - Common when you need a list of a certain length ahead of time.
  - Uses the repetition operator, similarly to strings
- Make an empty list:
  `lst = []`
  - Common when you're going to put things in the list coming from the user or a file.