COMP 10280
Programming I (Conversion)

John Dunnion

School of Computer Science
University College Dublin

COMP 10280 Programming I (Conversion)/Lecture 17

Outline

More on List Comprehensions

- As we have seen, we can use a list comprehension to apply an operation to the values in a sequence in order to create a new list
- `L = [x ** 2 for x in range(7)]`
- ```
  mixedList = [1, 2, 3.0, 'four', 5]
  squaredList = [x ** 2 for x in mixedList
       if type(x) == int or type(x) == float]
  ```
- `countList = [0 for x in range(4)]`
- While the last example is correct, there is a simpler solution to initialising a simple (one-dimensional) list:
- `countList = [0] * 4`
  ```

More on variables (1)

```
# Program to demonstrate variables

a = 1234
print('id(a) is', id(a))

b = a
print('id(b) is', id(b))

c = 'A random string'
print('id(c) is', id(c))

d = c
print('id(d) is', id(d))

e = 'A random string'
print('id(e) is', id(e))
```

More on variables (2)

```
if c == d:
    print('c and d are the same')
else:
    print('c and d are not the same')

if c == e:
    print('c and e are the same')
else:
    print('c and e are not the same')

c = c * 2
print('c is', c)
print('id(c) is', id(c))
print('d is', d)
print('id(d) is', id(d))

print('Finished!')
```

More on variables (3)

```
id(a) is 139704141068240
id(b) is 139704141068240
id(c) is 139704097995760
id(d) is 139704097995760
id(e) is 139704097995760
c and d are the same
c and e are the same
c is A random stringA random string
id(c) is 139704098042544
d is A random string
id(d) is 139704097995760
Finished!
```

Operations on Strings, Tuples and Lists

- We have now seen three difference sequence types: `str`, `tuple` and `list`
- All these sequence types have the following operations in common:
- `seq[i]` returns the ith element in the sequence
- `len(i)` returns the length of the sequence
- `seq1 + seq2` returns the concatenation of the two sequences
- `n * seq2` returns a sequence that repeats `seq2` `n` times
- `seq[start:end]` returns a slice of the sequence
- `e in seq` is `True` if `e` is contained in the sequence and `False` otherwise
- `e not in seq` is `True` if `e` is not in the sequence and `False` otherwise
- `for x in seq` iterates over the sequence

A puzzle!

- What is `[100,200,300,400,500][2:4][1]`?
- `400`

Methods

- In Object-Oriented Programming (OOP), a method can be thought of as a function associated with a given class
- A method invocation can be thought of as the call/invocation of such a function to an object of that class
- We use dot notation to place the object to which the method is to be applied before the function name
- `o.m(args)`
- We will talk more about object-oriented programming, classes, objects and methods later in the course

Methods associated with lists

- The following are some of the methods associated with lists
- All of them, except `count` and `index`, mutate the list
- `L.append(e)` adds the object `e` to the end of the list `L`
- `L.count(e)` returns the number of times that `e` occurs in `L`
- `L.insert(i, e)` inserts the object `e` occurs into `L` at index `i`
- `L.extend(L1)` adds the items in list `L1` to the end of `L`
- `L.remove(e)` deletes the first occurrence of `e` from `L`
  (This methods raises an exception if `e` is not in `L`)
- `L.index(e)` returns the index of the first occurrence of `e` in `L`
  (This methods raises an exception if `e` is not in `L`)
- `L.pop(i)` removes and returns the item at index `i` in `L`
  If `i` is omitted, it defaults to -1, to remove and return the last element of `L`

Demonstrating methods on lists (1)

```
# Program to demonstrate methods on lists
a = [0, 1234, 2345, 77.96, 0, 2]
print('a is', a)

print(a.count(77.96), a.count(100), a.count(0))

a.insert(2, 100)
a.append(0)
print('a is', a

print('First occurence of 100 is at index', a.index(100))

a.remove(0)
print('a is', a)
a.reverse()
print('a reversed is', a)

a.sort()
print('a sorted is', a)

a.pop()
print('a, having popped the last element, is', a)

print('Finished!')
```

Demonstrating methods on lists (2)

```
a is [0, 1234, 2345, 77.96, 0, 2]
Number of occurences of 77.96, 100 and 0:  1 0 2
a is [0, 1234, 100, 2345, 77.96, 0, 2, 0]
First occurence of 100 is at index 2
a is [1234, 100, 2345, 77.96, 0, 2, 0]
a reversed is [0, 2, 0, 77.96, 2345, 100, 1234]
a sorted is [0, 0, 2, 77.96, 100, 1234, 2345]
a, having popped the last element, is [0, 0, 2, 77.96, 100, 1234]
Finished!
```

Methods on strings (1)

- The following are some of the methods on strings
- Note that, since strings are immutable, all of them return values and have no side-effects
- `s.count(s1)` returns the number of times that the string `s1` occurs in `s`
- `s.find(s1)` returns the index of the first occurrence of the substring `s1` in `s`, and returns -1 if `s1` does not occur in `s`
- `s.rfind(s1)` the same as `find`, but starts from the end of `s` (the "`r`" in `rfind` stands for "reverse")
- `s.index(s1)` the same as `find`, but raises an exception if `s1` does not occur in `s`
- `s.rindex(s1)` the same as `index`, but starts from the end of `s`

Methods on strings (2)

- `s.lower()` converts all uppercase letters in `s` to lowercase
- `s.replace(old, new)` replaces all occurrences of the string `old` in `s` with the strings `new`
- `s.rstrip(s1)` removes trailing whitespace from `s`
- `s.split(d)` splits `s` using `d` as a delimiter
  Returns a list of substrings of `s`
  If `d` is omitted, the substrings are speated by arbitrary strings of whitespace characters (space, tab, newline, return and formfeed)

Demonstrating methods on strings (1)

```
# Program to demonstrate methods on strings

a = 'Cristiano Ronaldo plays soccer with Portugal!'
print('a is:', a)
print('The length of a is:', len(a))

print('Number of occurences of o:', a.count('o'))

print('First occurrence of o:', a.find('o'))
print('First occurrence of o, searching backwards:', a.rfind('o'))

print('String with all uppercase letters changed to lowercase:',
        a.lower())
print('a is:', a)

a = a.replace('Portugal', 'Real Madrid')
print('a is:', a)

print('The words in a:', a.split(' '))

print('Finished!')
```

Demonstrating methods on strings (2)

```
a is:  Cristiano Ronaldo plays soccer with Portugal!
The length of a is:  45
Number of occurences of o:  5
First occurrence of o:  8
First occurrence of o, searching backwards:  37
String with all uppercase letters changed to lowercase:
        cristiano ronaldo plays soccer with portugal!
a is:  Cristiano Ronaldo plays soccer with Portugal!
a is:  Cristiano Ronaldo plays soccer with Real Madrid!
The words in a:  ['Cristiano', 'Ronaldo', 'plays', 'soccer',
        'with', 'Real', 'Madrid!']
Finished!
```

Palindromes

- A palindrome is a string that reads the same way backwards as forwards
- The strings “abba” and “aba” are palindromes
- The strings “xx” and “x” are palindromes
- The strings “abc” and “ab” are not palindromes
- Normally case is ignored
- Spaces and other non-letter characters are also ignored
- “Abba” is a palindrome
- “Madam, I’m Adam!” is a palindrome

Program to check whether a string is a palindrome (1)

```
# Program to check whether a string is a palindromes
# Prompts the user for strings and checks each one
# Exits when an empty string is entered

def isPalindrome(s):
    """Checks whether the string s is a palindrome

    Assumes s is a str.
    Returns True if the letters in s form a palindrome;
    Returns False otherwise.
    Case and non-letters are ignored."""
```

Program to check whether a string is a palindrome (2)

```
    def toChars(s):
        """Converts a string to lowercase and removes non-letters

        Assumes s is a str.
        Converts uppercase letters to lowercase and removes non-letter
# First of all, convert uppercase letters to lowercase
        s = s.lower()
# Start with an empty string
        letterstring = ''
# Go through s...
        for c in s:
# ...  and add the character to the string if it is a letter
            if c in 'abcdefghijklmnopqrstuvwxyz':
                letterstring += c
        return letterstring
```

Program to check whether a string is a palindrome (3)

```
    def isPal(s):
    """Checks whether the string s is a palindrome

        Assumes that s is a str with only lowercase letters and no
            non-letters.
        Returns True if s is a palindrome;
        Returns False otherwise.
        Recursive function."""
        if len(s) <= 1:
# A palindrome of length 0 or 1 is a palindrome
            return True
        else:
# Compare the first and the last letters and check the remainder
#       of the string
            return s[0] == s[-1] and isPal(s[1:-1])

    return isPal(toChars(s))
```

Program to check whether a string is a palindrome (4)

```
str = input('Enter a string (empty string to exit):  ')

while str != '':
    if isPalindrome(str):
        print(str, 'is a palindrome')
    else:
        print(str, 'is not a palindrome')

    str = input('Enter a string (empty string to exit):  ')

print('Finished!')
```