COMP 141

Strings II

Rhodes College

1

Quiz 7

```
def main():
    file = open('tncities.txt', 'r')

    largestPop = 0
    largestPopCity = ''

    for line in file:
        line = line.rstrip()
        city, population = line.split(" ")
        population = int(population)

        if population > largestPop:
            largestPop = population
            largestPopCity = city

    file.close()

    print(largestPopCity)
    print(largestPop)

main()
```

tncities.txt - Notepad

File Edit Format View Help

```
Chattanooga 181624
Memphis 647506
Nashville 679318
Knoxville 189368
Clarksville 154250
Murfreesboro 149605
```

Output

```
Nashville
679318
```

2

Number 5 in Billboard File Reading Lab

Write a program to find all pairs of consecutively-ranked songs on the chart where their relative popularities are reversed from the prior week. In other words, find all back-to-back songs in the file where --- in the prior week --- the currently less-popular song was ranked higher than the currently more-popular song. Note that the songs need not be consecutively-ranked in the prior week, just the current week.

Hint: Use the sliding window technique.

3

Announcements

Reminders:

Program 6 - due Sunday, April 5th

4

Using len function

Prints 1 letter of city on each line

```
city = 'Boston'
index = 0
while index < len(city):
     print(city[index])
     index += 1
```

Equivalent Code

```
city = 'Boston'
for index in range(0, len(city)):
     print(city[index])
```

5

5

Accessing Characters Review

Strings are stored character by character.
Each character in a string is numbered by its position:

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

The numbers shown here above the characters are called ***indices*** (singular: index) or ***positions***.

6

6

Negative Indices

Negative indexing can be used.
Particularly useful for getting characters near the end of a string.

0	1	2	3	4	5	6	7
-8	-7	-6	-5	-4	-3	-2	-1
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

```
s[2] is the same as s[-6] both refer to "m"

To find last letter in string use:
s[-1]
```

7

7

String Indices

- Two ways to use square brackets
  - 1 number inside -> gives you 1 character of a string
    - s[0] gives you the first character in s
    - If s = "Computer", s[0] gives you 'C'
  - 2 numbers inside (separated by a colon) -> gives you a ***substring*** or string ***slice***

8

8

String Slicing

- **Slice:** span of items taken from a sequence, known as *substring*
  - Slicing format: `string[start : end]`
    - Expression will return a string containing a copy of the characters from `start` up to, but not including, `end`
    - If `start` not specified, 0 is used for start index
    - If `end` not specified, `len(string)` is used for end index
  - Slicing expressions can include a step value and negative indexes relative to end of string

9

9

String Slicing

`s[a:b]` gives you a substring of s starting from index a and ending at index b-1.

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

```
s[0:1] -> "C" just like s[0]
s[0:2] -> "Co"
s[0:7] -> "Compute"
s[3:6] -> "put"
s[0:8] -> "Computer"
```

10

10

Indices Don't have to be Literal Numbers

Say we have this code:

```
s = input("Type in a string: ")
x = int(len(s) / 2)
print s[0:x])
```

What does this print?

11

11

More Fun with Indices

- Examples using negative indices
- A negative index counts from the right side of the string, rather than from the left

```
s = "Computer"
print(s[-1])          #prints r
print(s[-3:len(s)])   #prints ter
print(s[1:-1])        #prints ompute
```

12

12

More Fun with Indices

- Slices don't need both left and right indices
- Missing left -> use 0 [far left of string]
- Missing right -> use len(s) [far right of string]

```
s = "Computer"
print(s[1:])        #prints omputer
print(s[:5])        #prints Compu
print(s[-2:])       #prints er
```

13

13

Class Practice

- Write a function called **`total_seconds`** that takes one string argument. This argument will be a string of the form "M:SS" where M is a number of minutes (a single digit) and SS is a number of seconds (2 digits). This function should calculate the total number of seconds in this amount of time and **return** it as an integer. (Hint: Use string slicing/indices)
- Write a function called **`count_digits`** that returns the number of digits in a string.
  - `count_digits("abc123def5") returns 4`
- Write a function called sum_digits that returns the sum of all the digits in a string.
  - `sum_digits("abc123def5") returns 11`
  `(because 1 + 2 + 3 + 5 = 11)`

14

14