CMPT 120
Intro to CS & Programming I

WEEK 4 (Jan. 27-31)

— Jérémie O. Lumbroso —

Lecture 9:
All About Strings

http://www.sfu.ca/~jlumbros/Courses/CMPT120/

Notion central to many useful algorithms

MANIPULATING STRINGS

What Are Strings?

- Strings store text in Python
- `"a string"` and `'also a string'`
- Single and double quotes can be used interchangeably

- **Avoid confusion with variables:** If a word is not contained in quotes it is a variable

  `>>> print hello` (prints contents of variable hello)

  `>>> print "hello"` (prints the string "hello")
- Be careful with this distinction!

- **Finally:** the function `len` gives the length of a string, i.e., `len("hello")` gives 5

Iterating Over Strings I

- Strings can be iterated, this means that they can be used in a for loop (instead of a range)

- Print every character in the string `"hello"`

```
for ch in "hello":
   print ch
```

- Also works with string variables

```
somestring = "hello"
for ch in somestring:
   print ch
```

Counting Occurrences of Letter

```
# Counts the number of occurrences of letter in phrase.

def countLetter(phrase, letter):
  total = 0
  for ch in phrase:
    if ch == letter:
      total = total + 1
  return total
```

- Can you use this function count the number of words in a sentence?
- Can you modify this to count vowels (the letters "a", "e", "i", "o", "u") in a phrase?

Slicing Strings (Again)

- String can be sliced: a character or range of character can be accessed using the following syntax
  - `mystring[k]` gives character #k of my string
  - `mystring[a:b]` gives range from character #a to #b
  - `mystring[a:]` gives range from character #a to end
  - `mystring[:b]` gives range from beginning to #a

- **Important:** strings (and everything else in Python) are indexed in 0; this means that the first character of a `mystring` is `mystring[0]` **not** `mystring[1]`

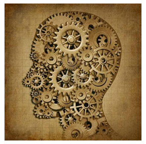
Iterating Over Strings II

- Using slicing and the function `len`, there is another way to iterate over strings

```
mystring = "hello"
for i in range(len(mystring)):
   print mystring[i]
```

- Idea:
  - we do a for loop on the range 0… len(mystring)-1
  - we access each character of string using its index
- Less practical than other way, but useful when the position of a character is important

Other Example

- Here this “decryption” function takes a string and only prints one of every character
- What is the secret message? 

```
def easyDecrypt(word):
  for i in range(len(word)):
    if i % 2 == 0:
      print word[i]

easyDecrypt("wyhbaftk fajmu did tshanyuisnfg")
```

A Got it!

B Nope

(Code available at `http://goo.gl/IlYWDm`)

Concatenation of strings

- Concatenation (gluing two strings together) is done using the addition operator

```
>>> first_string = "hel"
>>> second_string = "lo there"
>>> final_string = first_string + 
second_string
>>> print final_string
```

- Concatenation is useful to create strings in loops

Reverse String

This shows how concatenation can be used to create a new string

```
def reverseString(string):
  result = ""
  for ch in string:
    # By concatenating the character in front
    # we are reversing the string
    result = ch + result
  return result

print reverseString("sdrawkcab si sgnirts siht")
```

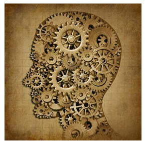
String Multiplication

- Finally the operation * multiplication can be used to repeat several copies of a string

```
my_string = "hello "
print my_string * 5
```

- Can be useful when you want to display a histogram?

Example



```
def numberDiagram(first, last):
  num_divbytwo = 0
  num_divbythree = 0
  num_divbyfive = 0
  for i in range(first, last+1):
    if i % 2 == 0:
      num_divbytwo = num_divbytwo + 1
    if i % 3 == 0:
      num_divbythree = num_divbythree + 1
    if i % 5 == 0:
      num_divbyfive = num_divbyfive + 1
  print "Bar graph of divisibility from", first, "to", last
  print "2", "=" * num_divbytwo
  print "3", "=" * num_divbythree
  print "5", "=" * num_divbyfive

numberDiagrams(63, 104)
```

```
Bar graph of divisibility from 63 to 104
2 =====================
3 ==============
5 ========
```

A Tried it!

B Nope

(Code available at `http://goo.gl/Azkt20`)

Type Conversion

- A string **cannot** be used as a number; this gives an error:

```
>>> "3" + 4
```

- But you can use the type conversion functions
  - `str(xxx)` converts `xxx` into a string
  - `int(xxx)` converts `xxx` into an integer
  - `float(xxx)` converts `xxx` into a float
- This works:

```
>>> int("3") + 4
>>> somestring = "123"
>>> int(somestring)*4
```

help(str)

STRING FUNCTIONS

Bounty of Function on Strings

- The type `str` of strings has a lot of built-in methods that can be used

- They are all used by applying a period, and then typing the name of the function: `"hello".isdigit()`

Get their list

- In Python interactive shell: `help(str)`
- On Internet: http://docs.python.org/2/library/stdtypes.html#string-methods

Some Functions to Test a String

- These functions return True if a property is verified and False if not
- For instance
  - "123".isdigit() tests whether "123" is an integer
  - "some word".islower() tests whether all alphabetical characters of the string are lowercase
  - "some word".isupper() tests if all alphabetical characters of the string are uppercase
  - …

Some Functions to Modify a String

- `"   edde   ".strip()` removes trailing and leading space from a string
- `"ALL UPPER".lower()` transforms all alphabetical characters to lowercase
- `"all lower".upper()` does opposite
- `"11".zfill(k)` returns the string with k leading zeroes
- many other functions…

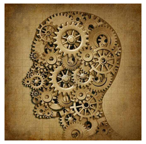
Try It Up Yourself

What does `raw_input()` do?

? ? ? ? ?

Pacing and Understanding

How well did you understand today?



- A Too easy, this lecture is way below my abilities
- B Everything went at a good pace, and I am fine
- C Too fast, but I will catch up on my own
- D Too fast, and I need you to slow down
- E I really do not think I can handle this

Course Exercise 3

- Using what you have learned in this lecture, define a function with two parameters
  `first_pos_of_char(astring, achar)`
- Takes a string `astring` and a character `achar` and returns the first position in which the character appears in `astring`, and -1 if the character does not appear in the string
- Submission on Coursys: http://courses.cs.sfu.ca