MANIPULATING STRINGS AND LAMBDA FUNCTIONS

CS 3080: Python Programming

UCCS University of Colorado Colorado Springs

String literals

A String literal is a sequence of characters used by Java programmers to populate String objects or display text to a user. The characters could be letters, numbers or symbols and are enclosed within two quotation marks

- Double quotes
  - *Using double quotes allows the use of the single quote inside the string*
- Escape characters
  - *Allows the use of characters that are otherwise impossible to put into a string.*
  - *Consists of a backslash (\) followed by the character you want to add to the string*
    - Example: escape character for a single quote is \’
- Raw string
  - *Completely ignores all escape characters and prints any backslash that appears in the string.*
  - *Place an* **r** *before the beginning quotation mark of a string to make it a raw string*

Table 6-1: Escape Characters

Escape character	Prints as
\'	Single quote
\"	Double quote
\t	Tab
\n	Newline (line break)
\\	Backslash

Full list of escape characters: https://www.quackit.com/python/reference/python_3_escape_sequences.cfm

String literals

Be careful of copy/paste from Powerpoint…quotes are special characters in Microsoft products…

- Multiline string
  - *A multiline string in Python begins and ends with either three single quotes or three double quotes.*
  - *Any quotes, tabs, or newlines in between the “triple quotes” are considered part of the string.*
  - *Python’s indentation rules for blocks do not apply to lines inside a multiline string*
  - *Escaping single and double quotes is optional in multiline strings.*

Indexing and slicing strings

- Strings use indexes and slices the same way lists do
- If you specify an index, you will get the character at that position in the string
- If you specify a range from one index to another, the starting index is included and the ending index is not
- You can think of the string 'Hello world!' as a list and each character in the string as an item with a corresponding index.

'	H	e	l	l	o		w	o	r	l	d	!	'
	0	1	2	3	4	5	6	7	8	9	10	11	

The in and not in Operators with Strings

- The **in** and **not in** operators can be used with strings just like with list values.
- Will evaluate to a Boolean True or False

Putting Strings Inside Other Strings

- Simpler approach is to use string interpolation.
  - *Use the %s operator inside the string which acts as a marker to be replaced by values following the string*
- Don’t have to call str() to convert values to strings
- Python 3.6 introduced *f-strings*

Useful string methods

Used to analyze strings or create transformed string values.

- *spam = 'Hello world!'*
- *spam**.upper()* *# these methods do not change the string itself*
- *spam**.lower()* *# but return new string values*
- *spam**.capitalize()* *# Capitalize first letter, 'hello world' to 'Hello world'*
- *spam.**islower()* *# False*
- *spam.**isupper()* *# False, 'HELLO WORLD!'.isupper() -> True*
- *'HELLO'.**lower().islower()*** *# True*
- ***isalpha()*** *# True if only letters and is not blank*
- ***isalnum()*** *# True if only letters and numbers and is not blank.*
- ***isdecimal()*** *# True if only numeric characters and is not blank.*
- ***isspace()*** *# True if only spaces, tabs, and new lines and is not blank.*
- ***istitle()*** *# True if only all words begin with an uppercase letter/number*
  # followed by only lowercase letters or number.

Useful string methods

- ***.startswith()***
- ***.endswith()***
- *', '**.join([**'cats', 'rats', 'bats'**])*** *# 'cats, rats, bats'*
- *' '.join(['My', 'name', 'is', 'Simon'])* *# 'My name is Simon'*
- *'ABC'.join(['My', 'name', 'is', 'Simon'])* *# 'MyABCnameABCisABCSimon'*
- *'My name is Simon'**.split()*** *# ['My', 'name', 'is', 'Simon']*
- *'MyABCnameABCisABCSimon'.split('ABC')* *# ['My', 'name', 'is', 'Simon']*
- *'My name is Simon'.split('m')* *# ['My na', 'e is Si', 'on']*

Useful string methods

- rjust(), ljust(), center()
  - *Especially useful when you need to print tabular data that has correct spacing*

- strip(), rstrip(), lstrip()
  - *Useful if you want to strip off whitespace characters (space, tab, and newline) from the left side, right side, or both sides of a string*

pyperclip module

- The pyperclip module has copy() and paste() functions that can send text to and receive text from your computer's clipboard.
- Does not come with Python.
- Sending the output of your program to the clipboard will make it easy to paste it to an email, word processor, or some other software.

  - *import pyperclip*
  - *pyperclip.copy('Hello world!')*
  - *pyperclip.paste() # 'Hello world!'*

LAMBDA FUNCTIONS (NOT IN TEXTBOOK)

Lambda functions

- The **lambda** keyword in Python provides a shortcut for declaring small anonymous functions.
- Lambda functions behave just like regular functions declared with the def keyword.
- They can be used whenever function objects are required.

Syntax:
lambda arguments : expression

- It can only contain expressions, not statements
- It is written as a single line of execution
- It can be immediately invoked

Lambda functions

```
def add(x, y):
    return x + y

print(add(5, 3))                    # 8

add = lambda x, y: x + y
print(add(5, 3))                    # 8
```

Lambda functions

```
(lambda x, y: x + y)(5, 3)
```

- *The difference is we didn't bind it to a name like add before we used it.*
- *We simply stated the expression we wanted to compute and then immediately evaluated it by calling it like a regular function*
- Unlike lambda forms in other languages, where they add functionality, Python lambdas are only a shorthand notation if you're too lazy to define a function (says the official Python documentation https://docs.python.org/3/faq/design.html#why-can-t-lambda-expressions-contain-statements)