CPSC 1301, Computer Science I Lab Assignment

Learning Python, 5th Edition, Lab 6a

Run the following commands in your Python interpreter. These are from *Learning Python, 5th Edition*, Chapter07. Submit a transcript of your session as your deliverable for this lab. This should be a plain text file named pythonlab-Chapter07_lastname.txt.

1 Pointers

quotation marks For all purposes, single quotes (') and double quotes (") are interchangeable. Most Python coders prefer single quotes. Some have a strong preference for double quotes because, in languages such as C and derived languages (C++, C#, Objective C, etc.), double quotes designate strings while single quotes designate characters. For most purposes, triple quotes, either '''or """, can be used interchangeably with single or double quotes.

escape sequences An excape sequence consists of the back slash (\) and another character, together representing just one character. Most common are the tab character (\t), the newline character (\n), and the back slash itself (\\).

raw strings Sometimes, you want text just to be text. You can Tell Python this by putting a *r* before the opening quote, like this: `r'This will be a raw string.'`.

string concatenation Use the plus symbol (+) to glue strings together, like this: `'Barack'+ 'Hussein'+ 'Obama'`. Note that you have to convert numbers to strings if you want to glue numbers. `str(4) + str(4) = str(8)`. The reason is that the plus symbol with numbers means addition, and you don't want addition but string concatenation.

2 Lab

```
    # string introduction
>>> 'shrubbery', "shrubbery"
>>> 'knight"s', "knight's"
>>> title = "Meaning " 'of' " Life" # Implicit concatenation
>>> title
>>> 'knight\'s', "knight\"s"
>>> s = 'a\nb\tc'
>>> s
>>> print(s)
>>> len(s)
>>> s = 'a\0b\0c'
>>> s
>>> len(s)
>>> mantra = """Always look
>>> on the bright
>>> side of life."""
>>> mantra
>>> print(mantra)
>>> menu = """spam # comments here added to string!
>>> eggs # ditto
>>> """
>>> menu
>>> menu = (
>>> "spam\n" # comments here ignored
>>> "eggs\n" # but newlines not automatic
>>> )
>>> menu
    # concatenation
>>> len('abc') # Length: number of items
>>> 'abc' + 'def' # Concatenation: a new string
>>> 'Ni!' * 4 # Repetition: like "Ni!" + "Ni!" + >>>
>>> len('abc') # Length: number of items
>>> 'abc' + 'def' # Concatenation: a new string
>>> 'Ni!' * 4 # Repetition: like "Ni!" + "Ni!" + >>>
>>> print('-------
>>>    more... ---') # 80 dashes, the hard way
>>> print('-' * 80)
>>> myjob = "hacker"
>>> for c in myjob: print(c, end=' ') # Step through items, print each (3.X form)
>>> "k" in myjob # Found
>>> "z" in myjob # Not found
>>> 'spam' in 'abcspamdef' # Substring search, no position returned
>>> S = 'spam'
    # slicing, important for later
>>> S[0], S[-2] # Indexing from front or end
>>> S[1:3], S[1:], S[:-1] # Slicing: extract a section
>>> S = 'abcdefghijklmnop'
>>> S[1:10:2] # Skipping items
>>> S[::2]
>>> S = 'hello'
>>> S[::-1] # Reversing items
>>> S = 'abcedfg'
>>> S[5:
>>> 'spam'[1:3] # Slicing syntax
>>> 'spam'[slice(1, 3)] # Slice objects with index syntax + object
>>> 'spam'[::-1]
>>> 'spam'[slice(None, None, -1)]
    # string conversion
>>> "42" + 1
>>> int("42"), str(42) # Convert from/to string
>>> print(str('spam'), repr('spam')) # 2.X: print str('spam'), repr('spam')
>>> S = "42"
>>> I = 1
>>> S + I
>>> int(S) + I # Force addition
>>> S + str(I)
>>> str(3.1415), float("1.5")
>>> text = "1.234E-10"
>>> float(text)
>>> ord('s')
>>> chr(115)
>>> S = '5'
>>> S = chr(ord(S) + 1)
>>> S
>>> S = chr(ord(S) + 1)
>>> S
>>> int('5')
>>> ord('5') - ord('0')
>>> int('1101', 2) # Convert binary to integer: built-in
>>> bin(13) # Convert integer to binary: built-in
    # changing strings
>>> S = 'spam'
>>> S[0] = 'x' # Raises an error!
>>> S = S + 'SPAM!' # To change a string, make a new one
>>> S
>>> S = S[:4] + 'Burger' + S[-1]
>>> S
>>> S = 'splot'
>>> S = S.replace('pl', 'pamal')
>>> S
>>> 'That is %d %s bird!' % (1, 'dead') # Format expression: all Pythons
>>> 'That is {0} {1} bird!'.format(1, 'dead') # Format method in 2.6, 2.7, 3.X
    # string methods
>>> S = 'spam'
>>> result = S.find('pa')
>>> S = 'spammy'
>>> S = S[:3] + 'xx' + S[5:] # Slice sections from S
>>> S
>>> S = 'spammy'
>>> S = S.replace('mm', 'xx') # Replace all mm with xx in S
>>> S
>>> S = 'xxxxSPAMxxxxSPAMxxxx'
>>> where = S.find('SPAM') # Search for position
>>> where # Occurs at offset 4
>>> S = S[:where] + 'EGGS' + S[(where+4):]
>>> S
>>> S = 'xxxxSPAMxxxxSPAMxxxx'
>>> S.replace('SPAM', 'EGGS') # Replace all
>>> S.replace('SPAM', 'EGGS', 1)
>>> S = 'spammy'
>>> L = list(S)
>>> L
>>> L[3] = 'x' # Works for lists, not strings
>>> L[4] = 'x'
>>> L
>>> S = ''.join(L)
>>> S
>>> 'SPAM'.join(['eggs', 'sausage', 'ham', 'toast'])
    # parsing text
>>> line = 'aaa bbb ccc'
>>> col1 = line[0:3]
>>> col3 = line[8:]
>>> col1
>>> col3
>>> cols = line.split()
>>> cols
>>> line = 'bob,hacker,40'
>>> line.split(',')
>>> line = "i'mSPAMaSPAMlumberjack"
>>> line.split("SPAM")
>>> line = "The knights who say Ni!\n"
>>> line.rstrip()
>>> line.upper()
>>> line.isalpha()
>>> line.endswith('Ni!\n')
>>> line.startswith('The')
>>> line
>>> line.find('Ni') != -1 # Search via method call or expression
>>> 'Ni' in line
>>> sub = 'Ni!\n'
>>> line.endswith(sub) # End test via method call or slice
>>> line[-len(sub):] == sub
    # string formatting
>>> 'That is %d %s bird!' % (1, 'dead') # Format expression
>>> exclamation = 'Ni'
>>> 'The knights who say %s!' % exclamation # String substitution
>>> '%d %s %g you' % (1, 'spam', 4.0) # Type-specific substitutions
>>> '%s -- %s -- %s' % (42, 3.14159, [1, 2, 3]) # All types match a %s target
>>> x = 1234
>>> res = 'integers: ...%d...%-6d...%06d' % (x, x, x)
>>> res
>>> x = 1.23456789
>>> x # Shows more digits before 2.7 and 3.1
>>> '%e | %f | %g' % (x, x, x)
>>> '%E' % x
>>> '%-6.2f | %05.2f | %+06.1f' % (x, x, x)
>>> '%s' % x, str(x)
>>> x = 1234
>>> res = 'integers: ...%d...%-6d...%06d' % (x, x, x)
>>> res
>>> x = 1.23456789
>>> x # Shows more digits before 2.7 and 3.1
>>> '%e | %f | %g' % (x, x, x)
>>> '%E' % x
    # skipping string formatting methods
```