Welcome to Google Docs
Quickly create new documents and open existing ones.
Share with others to edit together.
TAKE A TOUR
Lab 2

CSE101 : Introduction to Programming

(Monsoon 2019)

August 14, 2019

Name : Mihir Chaturvedi
Roll Number : 2019061

The purpose of this lab is to get you comfortable with using assignment statements, strings, and their associated methods. This lab is very similar to the previous one, in that you will be typing commands into the Python interactive shell and recording the results.

- **Variables and Assignment Statements**
  As we saw in class, assignment statements are different from expressions. A statement like a = 3 < 5 is a command to do something. In particular, this command

  1. Evaluates the expression on the right-hand side of the = (in this case, 3 < 5), and
  2. Stores its value in the variable on the left-hand side of the =, in this case, a.

  Because it is not an expression, Python will not actually output a result when you type it in; it will just perform the command silently. It is important that you understand the difference. In the table on the next page, the first column contains either an expression or a command. If it is an expression, write the value. If it is a command, you should just write "None" (we have done the first one for you). Because some of the entries are commands, it is important that you enter the expressions or commands in exactly the order they are given.

Statement or Expression	Expected Value	Calculated Value	Reason for Calculated Value
i = 6	None	None	Assignment, so nothing returned
i	6	6	Expression, so returns value

j	Undefined error	Undefined error	j variable has not been assigned any value
j = 1	None	None	Assignment, so nothing returned
j + 4	5	5	Expression, so returns value
j = j + i	None	None	Assignment
j	7	7	Expression
i	6	6	Expression
w = ‘Hello’	None	None	Assignment
i + w	TypeError	TypeError	int+string

- **String Expressions**
  Throughout this section, pay close attention to spaces and to the different types of quotation marks being used. We use both ‘ (single quote) and “ (double quote).

Expression	Expected Value	Calculated Value	Reason for Calculated Value
‘Truth ’ + ‘is ’ + ‘always ’ + ‘best’	Truth is always best	Truth is always best	simple concatenation of strings
“Truth ” + “is ” + “best”	Truth is best	Truth is best	string concat
“Truth” + (‘is ’ + “best”)	Truth is best	Truth is best	string concat
‘A double quote: ” ’	A double quote "	A double quote: ”	string concat without + operator
“A single quote: ’ ”	A single quote: '	A single quote: '	expression output
‘A single quote: ’ ’	Error	Error	There is an extra single quote at the end that is not matched and so it errors
‘’ + ‘lol’	lol	lol	string concat
‘’ + ‘4 / 2’	4 / 2	4 / 2	string concat
‘’ + 4 / 2	TypeError	TypeError	string+float is not allowed
‘’ + str(4 / 2)	2.0	2.0	conversion to string and concat

- **Functions**
  Built-in functions are those that do not require you to import a module to use them. You can find a list of them in the Python documentation: https://docs.python.org/3/library/functions.html
  Note that the casting and typing operations are listed as functions. While this is true in Python, this is not always the case in other programming languages.

Expression	Expected Value	Calculated Value	Reason for Calculated Value
min(25, 6)	6	6	6 < 25
max(27, 4)	27	27	27 > 4
min(25, max(28, 4))	25	25	28>4, 25<28
abs(26)	26	26	absolute val
abs(-26)	26	26	absolute val
round(23.6)	24	24	rounds to nearest int
round(-23.6)	-24	-24	rounds to nearest int
round(23.64, 0)	24	24.0	round to 0 decimal values of precision, returns float
round(23.64, 1)	23.6	23.6	round to 1 decimal value of precision
round(23.64, 2)	23.64	23.64	round to 2 decimal values of precision
len('Truth')	5	5	number of characters in the string

- **Using a Python Module**
  One of the more important Python library modules is the math module. It contains essential mathematical functions like sin and cos. It also contains variables for mathematical constants such as pi. To learn more, look at its online documentation: https://docs.python.org/3/library/math.html
  To use a module, you must import it. Type the following into the Python interactive shell:

  import math

  You can now access all of the functions and variables in math. However, to use any of them, you have to put “math.” Before the function or variable name. For example, to access the variable pi, you must type **math.pi**. Keep this in mind as you fill out that table below.

Expression	Expected Value	Calculated Value	Reason for Calculated Value

math.sqrt(16)	4.0	4.0	returns float
math.sqrt(-16)	error	error	square root of negative number
math.floor(6.7)	6	6	greatest integer function
math.ceil(6.7)	7	7	rounds to greater int
math.ceil(-6.7)	-6	-6	rounds up
math.copysign(2,-6.7)	-2	-2.0	copies sign of second arg to first, converts to float
math.trunc(6.7)	6	6	similar to int()
math.trunc(-6.7)	-6	-6	similar to int()
math.pi	3.14159...	3.141592653589793	value of pi, greater precision
math.cos(math.pi)	-1.0	-1.0	cos(pi)

- **String Methods**
  Now that you understand strings and (calling) functions, you can put the two together. Strings have many handy methods, whose specifications can be found at the following URL:
  https://docs.python.org/2/library/stdtypes.html#string-methods
  Before starting with the table below, enter the following statement into the Python shell:
  S = ‘Hello World!’
  Once you have done that, use the string stored in s to fill out the table, just as before.

Expression	Expected Value	Calculated Value	Reason for Calculated Value
s[1]	e	e	strings are zero-indexed, [1] returns second char
s[15]	Error	Error: string index out of range	There are only 12 characters, accessing the 15th errors out
s[1:5]	ello	ello	substring from 1 to 5
s[:5]	Hello	Hello	substring from 0 to 5
s[5:]	World!	World!	substring from 6 to end
‘H’ in s	True	True	Whether ‘H’ is present in the string
‘X’ in s	False	False	‘x’ is not present

s.index(‘w’)	-1	Error	‘w’ is not present
s.index(‘x’)	Error	Error	‘x’ is not present
s.index(‘l’, 5)	9	9	returns index of the ‘l’ after the 5th character
s.find(‘e’)	1	1	returns index of ‘e’
s.find(‘x’)	-1	-1	returns -1 when substring not found

Let q1 is a variable assigned as in the below given statement.

q1 = "The phrase, “Don\'t panic!” is frequently uttered by consultants.’

You could also write a string like this using double quotes as the delimiters. Rewrite the assignment statement for q1 above using a double-quoted string literal:

```
q1 = “The phrase, \"Don't panic!\" is frequently uttered by consultants.”
```

For your last exercise, we want you to write Python code to extract the substring inside the double quotes (which is "Don't panic!"). But we want to do it in a way that is independent of q1. That means, even if you change the contents of q1, your answer should still work, provided that q1 still has a pair of double-quote characters somewhere.

In the box below, write a sequence of one or more assignment statements, ending with an assignment to a variable called inner (the other variables can be named whatever you want). The assignment statements should use string slicing to remove the unwanted parts of q1. When you are done, the contents of inner should be the substring inside the double-quote characters (but not including the double quotes themselves).

```
i = q1.find('"') + 1
inner = q1[i:q1.find('"',i)]
```

To test that your statements are correct, do the following. First, type in

q1 = ‘The phrase, “Don\'t panic!” is frequently uttered by consultants.’

Then type in your statements from the box above. Finally, print the value of inner. You should see Don't panic, without the quotes (printing always removes the quotes from a string value).

Now, try the process again with

q1 = ‘The question “Can you help me?” is often asked in consulting hours.’

Type in the assignment statement above, then type in your statements from the box, and finally print the value of inner. You should see Can you help me?, without the quotes. If you had to modify your answer in the box for the second q1, you have done it incorrectly.