Python Open Labs

Variables, Expressions and Statements

Special Thanks: Dr. Charles R. Severance(University of Michigan)

Kunal Baweja

Constants

- Fixed values - numbers, strings, letters
- Values do not change, hence the term “constants”
- Numbers - 1, 2, 3
- Character - ‘a’, “b”
- String - “Hello World”, ‘This is also a string in python’ (notice the quotes)

```
>>> print 123
123
>>> print 98.6
98.6
>>> print 'Hello world'
Hello world

>>> print “What’s up ?”
??????????
```

Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables.
- You can change the contents of a variable in a later statement

```
>>> x = 5
>>> print x
5

>>> x = 5 + 2
>>> print ‘x’
?????
```

Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables.
- You can change the contents of a variable in a later statement

```
>>> x = 5
>>> print x
5

>>> x = 5 + 2
>>> print ‘x’
x
>>>print x
7
```

Beware of imposters !!

Variable Name Rules

Suggestion: Try to follow PEP-8 Style Guide for cleaner programs

Psst: Don’t memorize the style guide, install ‘*PyLint*’ in IDE to show you the way

- Must start with a letter or underscore _
- May consist of letters and numbers and underscores, nothing else
- Case Sensitive

Good: variable, variable_new, variable1

Bad: 123coolDude, *notCool, avo!dspecialch@r

Different: word, WORD, Word

Reserved Keywords

- Sometimes, you don't get to choose

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec	in	raise	
continue	finally	is	return	
def	for	lambda	try	

Assign values to a Variable

- We assign a value to a variable using the assignment statement (=)
- An assignment statement consists of an expression on the right-hand side and a variable to store the result.
- x = 5
- y = x + 2
- Basically, a variable refers to a memory location where values are stored.
- Variables can be updated even after being assigned a value

Numeric Expressions

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
**	Power
/	Division
%	Modulus (Remainder)

```
>>> x = 5
>>> y = x*2
>>>print y
10
>>>print 10*2
??
```

- We can't use 'x' for multiplication, hence *
- ^ is too busy being the bitwise XOR operator, hence ** for raised to power operation

Order of Evaluation

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print x
???
```

- Which operator to evaluate first ?
- Order of Precedence/Evaluation

Order of Precedence (Rules)

Highest to lowest:

- Parenthesis - ()
- Exponentiation - **
- Multiplication [*], Division[/], Remainder[%]
- Addition and Subtraction
- Left to Right

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print x
???
```

Order of Precedence (Contd)

```
1 + 2 ** 3 / 4 * 5 --- exponentiation

1 + 8/4 * 5 --- left to right ( * vs / ; / wins)

1 + 2*5 --- multiplication

1 + 10 --- addition

11
```

Division can be tricky

```
>>> print 10 / 2
5
>>> print 9 / 2
4
>>>print 9.0 / 2
4.5
>>> print 5.0 / 2.0
2.5
```

- Integer division truncates, i.e rounds down values
- Floating point division produces floating point numbers (at least on of numerator or denominator should be a floating point)

Clash of the Numbers (Integers v/s Floating Point)

- Floating point always “wins” over integer in a division or multiplication.
- If one of the operands is a floating point, the results is also floating point.
- The integer is converted to a floating point, internally by python, before evaluation

“Types”

“Type” refers to the category of a constant, variable or literal.

- String - “Hello World”
- Integer - 5
- Floating point - 4.5
- List - [1, 2, 3, 4, 5]
- Python is able to differentiate between “types” independently, i.e it can tell apart an integer from a string unlike many other languages.
- Pythonistas, do not need to declare “types” of variables (How convenient !!)

Importance of Type

- Python is able to internally differentiate “types”.
- Hence cool features like:
  - print 5 + 2 gives 7 (Addition)
  - print ‘Hello ’ + ‘World’ gives Hello World (Concatenation)
  - Doesn’t allow “Hello ” + 1 - error: incompatible types
- type() function returns the type of a variable, constant, literal or expression

Type Conversion (Typecasting)

- Implicit conversion
  - 5.0/2 => 5.0/2.0 = 2.5 (implicit conversion due to one floating point operand)
  - 5/2 = 2 (no implicit conversion)
- “Typecast” - explicit conversion to another type
  - int() - cast to integer
  - float() - cast to floating point number
  - str() - cast to a string

```
>>> x = int(4.5)
>>> print x
4

>>> print float(10)
10.0

>>> y = “12345”
>>> print type(y)
<type ‘str’>
>>> z = int(y)
>>>print type(z)
<type ‘int’>
```

Type Conversion (Typecasting)

```
>>> x = "123"
>>> print x + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str'
and 'int' objects
>>> print type(int(x))
<type 'int'>
>>> print int(x) + 1
124
```

User input

- We can instruct Python to pause and read data from the user using the raw_input() function
- raw_input() returns the user input as a string literal.

```
>>> x = raw_input()
Kunal
>>> type(x)
<type 'str'>
>>> print x
Kunal
```

- Typecast user input before using as anything other than a string.

Example: Add two numbers provided by user

```
input = raw_input(‘Enter first integer: ’)
x = int(input)
input = raw_input(‘Enter second integer: ’)
y = float (input)
print x + y
```

```
kunal@baweja:~/$ python add.py
Enter first integer: 1
Enter second integer: 2.5
3.5
```

Comments in Python Code (Good Practises)

- Anything after a # is ignored by Python
- Why comment ?
  - Describe what is going to happen in a sequence of code
  - Document who wrote the code or other ancillary information
  - Turn off a line of code - perhaps temporarily (not recommended, makes ugly code over time)
  - Makes life easier after a time gap, if you revisit the code
  - Explanation in comments makes code readable (less trouble for person who takes over)

```
# Open a file for reading
handle = open('a.txt', 'r')

#read text of file
text = handle.read()

#print file contents
print text
```

Exercise: Weekly Pay

Enter Hours: 5

Enter Hourly rate: 2.95

Pay is $14.75