01-2 Python Variables and Expressions

CSI 500

Spring 2018

Note: course material adopted loosely from :
Downey, Allen B. *Python for software design: how to think like a computer scientist*. Cambridge University Press, 2009.
http://greenteapress.com/wp/think-python/

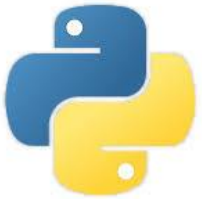
Values and Types

- Value
  - The quantity or information associated with a variable
- Type
  - The underlying category of the variable
  - Primitive types (numeric, Boolean)
  - Built-in data structures (list, string, tuple, dictionary)
  - User-defined structures (composite types, classes)
- How to determine the type of a variable?
  - Use the Python **`type( var )`** function

```
>>> type( '17' )
<type 'str'>
>>> type( 17 )
<type 'int'>
>>> type( 17.0 )
<type 'float'>
```

Variables

- Human-readable name that refers to a value
- Assignment statement uses the = operator
  - Create a new variable and assign it an initial value
  - Update the value of a previously defined variable
- State diagrams : a "map" for variable assignments



```
>>> message = 'Bring me a shrubbery'
>>> n = 17
>>> pi = 3.14159

>>> type( message ) # what happens?
>>> type( n )       # what happens?
>>> type( pi )      # what happens?
```

```
message -> 'Bring me a shrubbery'
n       -> 17
pi      -> 3.14159
```

state diagram
for some variables

Variable names and keywords

- Well chosen variable names aid in understanding the program
  - No size limit for variable names
  - Variables ARE case sensitive
  - Must begin with letter
  - Can't include special characters like '#', @, !, %
  - Can include underscore character '_' for readability
  - Can't be a reserved Python keyword

Python 3 Reserved Keywords

and	if
as	import
assert	in
break	is
class	lambda
continue	not
def	or
del	pass
elif	print
else	raise
except	return
exec (deprecated in python 3)	try
finally	while
for	with
from	yield
global	

Python Numerics

- Python supports several numeric bases (base 10, base 8, base 16)
  - Default is base 10
  - Leading '0o' indicates base 8
  - Leading '0x' indicates base 16
- Python supports scientific/engineering numeric notation
  - Floating point mantissa with integer or floating point mantissa designated by 'E' or 'e'
  - No blank space allowed between mantissa and exponent: so **1.0E3** is OK, but **1.0 E3** is not.

```
>>> num10 = 1000
>>> num8 = 0o1000
>>> num16 = 0x1000
>>> num_sci = 1.0E3

>>> num10          # what happens?
>>> num8           # what happens?
>>> num16          # what happens?
>>> num_sci        # what happens?
```

Example: Color codes

- Modern web browsers and programming environments have standardized colors
  - 140 "named" colors are widely used
  - Each color is composed of three 8-bit hex codes: red, green, and blue
  - Values range from 0x00 (0) to 0xFF (255) indicating min and max respectively

- Example: salmon = 0xFA8072
  - red component 0xFA = 250 250/255 = 98%
  - green component 0x80 = 128 123/255 = 48%
  - blue component 0x72 = 114 114/255 = 45%

Operators and Operands

- Operators
  - Symbols used to indicate operations that should be performed
- Operands
  - Things to which operators are applied, such as variables
- Precedence
  - Order in which operators are evaluated in an expression
  - Remember undergraduate algebra?
    - 3 + 2 * 4 ** 2 - 1

Answer: 34

Statements and Expressions

- Statement
  - A "statement" about the state of things
  - A bit of executable Python code
  - Does not return a value
- Expression
  - A set of instructions for what to do
  - Set of values (constants), operators, variables intended to be evaluated
  - Returns a value
  - Examples we've seen so far: print(), and assignment (=)

```
>>> freezing = 32.0          # assignment statement
>>> boiling = 212.0          # assignment statement
>>> boiling – freezing       # expression
180.0
```

Interactive mode versus script mode

- In interactive mode, you type expressions at the Python console
  - Statements executed w/o any reply
  - Expressions return the result of the computation and print to console
  - Very useful and Pythonic to have a >>> prompt running while coding in order to test small bits of code before inserting into a script file
- In script mode, you write a file of statements and expressions
  - Program output only occurs if you direct it to, such as via "print" statement
  - Execution of an expression does not result in output

Order of operations

- Operators follow precedence rules to determine order of execution
  - Python uses conventional mathematical rules
  - The mnemonic "PEMDAS" is helpful
    - Parenthesis
    - Exponentiation
    - Multiplication and Division
    - Addition and Subtraction
    - Same precedence (other) operators
- Evaluation is from Left to Right
- Parenthesis can be used to make evaluation order explicitly clear

Operators on Strings

- Strings are immutable arrays of characters
  - You can't do in-place assignment on a string
  - You can make a new string that does what you want based on an old string
- Strings can be combined with the + operator
- Strings can be replicated with the * operator

```
>>> msg = 'hello, how are you '
>>> name = 'Bob'
>>> greeting = msg + name
>>> print( greeting )
hello, how are you Bob

>>> lunch = 'Spam '
>>> print( lunch * 4 )
Spam Spam Spam Spam
```

immutable means it can't be changed once it's created

Comments in Python

- Comments are non-executable documentation embedded in your Python code scripts
  - Not so useful at the command prompt interactively
  - Identify script name, purpose, date of creation
  - Identify key functions and data types
  - Indicate important features, don't just narrate what the code is doing
- Single line comments designated by # character
  - All text from # to end of line is ignored
- Comment blocks begin and end with triple-quotes
  - You may use the double quote character " or the single quote character '
  - Any number of lines allowed inside a triple quote block

Summary

- Variables have an associated type (primitive, built-in, or user-defined)
  - Variables are assigned values using the assignment operator =
- Python numeric types include decimal, real, imaginary, hex, and octal
  - Python provides a wide set of mathematical and logical operators
- A statement is an assertion about the state of things (e.g. assignment)
  - does not return a value
- An expression is an executable instruction (e.g. math operation)
  - returns a value
- Comments are expressed using triple quotes or '#' for single line

Appendix: Python Operators

- Logical, Comparison
- Bitwise
- Arithmetic
- Negation and Index
- Structures

Operators and Operands: Logical, Compare

Operator	Function	Example

or
lambda args: expression
	Logical (lazy) OR
	Lambda expression
		>>> True or False
		True
		>>> is_divisible = lambda x, y : (x % y) == 0
		>>> is_divisible(6, 3)
		True

and	Logical (lazy) AND	>>> True and False False
not	Logical negation	>>> not False True

<, <=, >, >=,
==, <>, !=
is, is not
 	
in, not in
	Comparison operators
	
	Identity operator
	
	Sequence membership
		>>> 5 >= 3
		True
		>>> 5 is 5
		True
		>>> 5 in (3, 5, 7)
		True

Operators and Operands: Bitwise

Operator	Function	Example (Note: using 0x notation to emphasize bit-level ops; Not required by Python)
x | y	Bitwise OR	>>> 0x08 | 0x01 9 OR 0000 1000 / 0000 0001 / 0000 1001
x ^ y	Bitwise Exclusive-OR	>>> 0x0F ^ 0x01 14 XOR 0000 1111 / 0000 0001 / 0000 1110
x & y	Bitwise AND	>>> 0x07 & 0x02 2 AND 0000 0111 / 0000 0010 / 0000 0010
x << y	Bitwise shift X left by Y bits	>>> 0x01 << 3 8 << 3 0000 0001 / 0000 1000
x >> y	Bitwise shift X right by Y bits	>>> 0x08 >> 1 4 << 1 0000 1000 / 0000 0100

Operators and Operands: Arithmetic

Operator	Function	Example

+	Addition for numerics
	
	Concatenation for strings and lists
		>>> 3 + 3
		6
		>>> 'fee ' + 'fie'
		'fee fie'

-	Subtraction	>>> 5 – 3 2

*	Multiplication for numerics
	
	Replication for strings
		>>> 3 * 3
		9
		>>> 'foo ' * 3
		"foo foo foo"

/
 	
%
	Division for numerics
	(note: integer vs floating point evaluation)
	Modulus for integers
		>>> 5 / 3
		1
		>>> 8 % 3
		2

Operators and Operands: Arithmetic

Operator	Function	Example

**	Exponent	>>> 3 ** 2
		9
		>>> 3.0 ** 2.0
		9.0

%	Modulus
	
	Old print format operator (2.x only)
		>>> 8 % 3
		2
		>>> print ("value of pi to 2 digits = %4.2f") % (3.14159)
		3.14

Operators and Operands: Negation, Index

Operator	Function	Example

-x, +x, ~x	Unary negation
	
	Identity
	
	bitwise complement
		>>> -3
		-3
		>>> +5
		5
		>>> ~ 2
		-3

x[i]
 	
x[i:j]
 	
x.y
 	
x(…)
	Indexing
	
	Slicing
	
	Qualification (used in OO and classes)
	
	Function invocation
		>>> x = (1, 2, 4, 8)
		>>> x[0]
		1
		>>> x[1:3]
		(2, 4)
		>>> x.count(2)
		1
		>>> sum(x)
		15

Operators and Operands: Structures

Operator	Function	Example

(…)	Tuple creation	>>> x = (1, 3, 5)
		>>> x
		(1, 3, 5)

[…]	List creation	>>> x = [1, 3, 5]
		>>> x
		[1, 3, 5]

{ … }	Dictionary creation	>>> x = { "IBM" : "computers", "Dow" :
		"chemicals",\
		"Pfizer" : "pharmaceuticals" }
		>>> x["IBM"]
		"Computers"

` … `	String conversion	>>> x = 123
		>>> `x`
		'123'