



Learning to Program With Python – Part 2



Letters, Numbers, and Math (yes, math!!)

Based on the book:

Snake Wrangling for Kids, Learning to Program with Python
by Jason R. Briggs

(Version 0.7.7-python2.7, modified by SJL)

Presented by

Steve Arnold, Principal Scientist VCT Labs*

Stephanie Lockwood-Childs, President VCT Labs*

Nick Lockwood, Senior Engineer, VCT Labs

(also open source Gentoo Linux / Yocto developers)*



Numbers and Math



- 8 multiplied by 3.57 equals. . .

```
>>> number = 8 * 3.57 <enter>
```

```
>>> print(number)
```

```
28.56
```

```
>>> type(number)
```



Try it in pycrust!

- Basic arithmetic operators in Python:

+	Addition
-	Subtraction
*	Multiplication
/	Division

- Order of operations is important!
- When in doubt, use parentheses

```
>>> print((5 + 30) * 20)
```



Variables and Types



- A 'variable' in programming is just a place to put something
 - Can be like a variable in algebra
 - Can also be almost anything else
 - Really just a memory location with a name
- Python example:

```
>>> fred = 100
>>> print(fred)
100
```
- Upper and lower case letters are *different*
- Integers and floating point numbers are *different*
 - Fred != fred
 - 2 and 2.0 are *different types*



Numbers and Strings



- A variable is created when you name it
 - `fred = 200` (not a meaningful name)
 - `number_of_students = 200` (much better)
 - Variable names can use letters, numbers, underscore (but can't start with a number or have spaces)
- Basic “types” of variables common across most languages
 - Integers – fixed range, signed or unsigned
 - Floating point numbers – anything with a decimal
 - Strings – a string of characters (even just one)
 - All can be considered different kinds of “objects”
- Python infers the type from how it's used
 - `fred = "200"` is a string, but `fred = 200` is an integer



More With Strings



- Your first program (remember part 1?) prints a string

```
>>> print ('Hello, world!')
Hello, world!
>>> fred = 'this is a string'
>>> len(fred)
16
```



Try it in your
python shell!

- If $10 * 5 = 50$, what is $10 * 'a'$?

```
>>> print(10 * 'a')
aaaaaaaaaa
```

- Using '%s' as a placeholder for a value

```
>>> mytext = 'I am %s years old'
>>> print(mytext % 12)
I am 12 years old
```

- Formatting printed output

```
>>> print 'The value of PI is approximately
{0}.'.format(math.pi)
```





More Useful Data Types



- A shopping list (sort of) using a string:

```
>>> shopping_list = 'eggs, milk, cheese, celery, peanut  
butter, baking soda'  
>>> print(shopping_list)  
eggs, milk, cheese, celery, peanut butter, baking soda
```

- A shopping list (sort of) using a list:

```
>>> shopping_list = [ 'eggs', 'milk', 'cheese', 'celery',  
... 'peanut butter', 'baking soda' ]  
>>> print(shopping_list)  
['eggs', 'milk', 'cheese', 'celery', 'peanut butter',  
'baking soda']
```



Lists and Objects



Creating a list in Python:

```
>>> colors = ['red', 'blue', 'green']
>>> print colors[0]    ## red
red
>>> print colors[2]    ## green
green
>>> print len(colors)  ## 3
3
```

#-marks are
comments

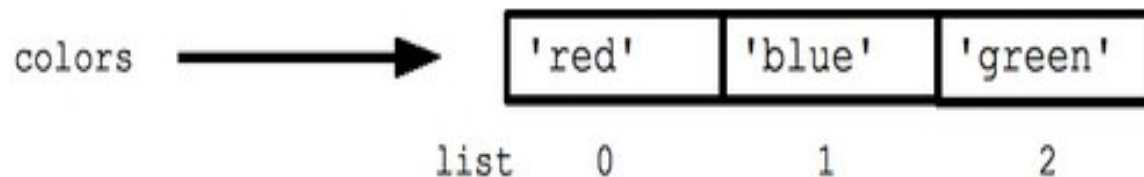
Q: What happens if you need a variable to hold several different things?

A: Make a list!

The native higher-level “container” types in Python include:

- Lists
- Dictionaries
- Tuples

(also known as
Abstract Data Types)





More About Lists and Objects



Almost everything in Python is an “object”. Objects have built-in “methods” (or functions) which are invoked via dot notation. Since the list type is an object, some common list methods are shown below:

- `list.append(elem)` -- adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.
- `list.insert(index, elem)` -- inserts the element at the given index, shifting elements to the right.
- `list.extend(list2)` adds the elements in list2 to the end of the list. Using `+` or `+=` on a list is similar to using `extend()`.
- `list.index(elem)` -- searches for the given element from the start of the list and returns its index. Throws a `ValueError` if the element does not appear (use `"in"` to check without a `ValueError`).
- `list.remove(elem)` -- searches for the first instance of the given element and removes it (throws `ValueError` if not present)
- `list.sort()` -- sorts the list in place (does not return it)
- `list.reverse()` -- reverses the list in place (does not return it)
- `list.pop(index)` -- removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of `append()`).



Note that these are **methods** on a list object, `"in"` is a keyword, and `len()` is a function that takes the list (or other object) as an argument.



A Word or Two About Language Keywords



Depending on the version, there are 20 – 30 or so words reserved as language keywords, meaning you cannot use them as variable names, etc (see Appendix A for a list).

Although other words used as function or other object names are not “reserved”, they can still cause a name clash if used without full dot notation.

One way to minimize or avoid name clashes is to import the top-level library object and use dot notation to call the function:

```
>>>> import math
>>>> print math.pi
3.14159265359
```



Hands-On



Task 1:

Make a list of your favorite toys and name it toys.
Make a list of your favorite foods and name it foods.
Join these two lists and name the result favorites.
Then print the variable favorites.

Task 2:

Create variables for your first and last name. Now create a string and use place-holders to add your name and print it.



This work is an original work by Stephen Arnold
<stephen.arnold@acm.org>

<<http://www.vctlabs.com>>

Portions copyright 2014 Stephen L Arnold. Some rights reserved.

The Gentoo Linux logo is Copyright 2006 Gentoo Foundation, used with permission.



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit <<http://creativecommons.org/licenses/by-nc-sa/1.0>> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Please contact Stephen Arnold <stephen.arnold@acm.org> for commercial uses of this work.