Mod 01 Python Flow Control

SCRIPTING ESSENTIALS

DR. BURKMAN

Learning Objectives

In this module you will learn the basic Python components to control the flow of your script.

- Booleans
- IF
- WHILE
- FOR
- BREAK/CONTINUE
- Functions

Python Basics

Values with an operator make up an expression

- 2 + 2 is an expression

Basic Python math operators (and they adhere to PEMDAS):

Table 1-1: Math Operators from Highest to Lowest Precedence

Operator	Operation	Example	Evaluates to...
**	Exponent	`2 ** 3`	8
%	Modulus/remainder	`22 % 8`	6
//	Integer division/floored quotient	`22 // 8`	2
/	Division	`22 / 8`	2.75
*	Multiplication	`3 * 5`	15
-	Subtraction	`5 - 2`	3
+	Addition	`2 + 2`	4

Python Basics

Python common data types are about what you expect.

Table 1-2: Common Data Types

Data type	Examples
Integers	`-2, -1, 0, 1, 2, 3, 4, 5`
Floating-point numbers	`-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25`
Strings	`'a', 'aa', 'aaa', 'Hello!', '11 cats'`

You can use “ or ‘ in matching pairs around text, but be consistent. A nice approach is to use “ if there is a ‘ in your text (“Jim’s Garage”) otherwise stick with ‘ (‘The Garage of Jim’)

Python Basics

Using + means two things:

- It adds numbers (5 + 2)
- It concatenates strings (‘Jim ‘ + ‘and ‘ + ‘Nita’)

Using * means two things:

- It multiplies numbers (5 * 2)
- It replicates a string (‘ha’ * 5)

Python Basics

Variables don’t need initialized before use, they are created when assigned a value

- Assigning a new value to a variable overwrites the old value

```
❶ >>> spam = 40
  >>> spam
  40
  >>> eggs = 2
❷ >>> spam + eggs
  42
  >>> spam + eggs + spam
  82
❸ >>> spam = spam + 2
  >>> spam
  42
```

```
>>> spam = 'Hello'
>>> spam
'Hello'
>>> spam = 'Goodbye'
>>> spam
'Goodbye'
```

Python Basics

Variables must:

- Be only one word
- Use only letters, numbers and _
- Cannot begin with a number

Variables are case sensitive!

Python convention now is to write functions and variable names in lowercase, with words separated by underscores

- dog_name, alert_output
  - mixedCase should only be used in existing code to preserve the style. CamelCase isn't used.

Python Basics

You cannot concatenate strings and number, so you have to convert the number to a string

- Str(29)
- Input always inputs a string!

You can also convert numbers stored as text to int or float

```
>>> spam = input()
101
>>> spam
'101'
```

Understanding Modules

Python comes with a standard library that provides many modules

- https://docs.python.org/2/library/

For example, to make a random number we'd need to import the random module before using it (we generally do all imports at the top of the script):

```
import random
print(random.randint(1,10))
```

Understanding Modules

To get the documents on all the functions at once:

- help(module_name)

To list all available functions in the module:

- dir(module_name)

Python Basics

Give this a try in a script

```
#This program says hello and asks for my name


print('Hello world!')
print('What is your name?')  #ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
print('What is your age?')  #ask for their age
myAge = input()
print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

Boolean Variables

Boolean variables can be True or False (capitalization just like that)

Other comparison Operators:

- Remember, = assigns a value to a variable, == checks to see if two things are equal in value

Table 2-1: Comparison Operators

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

```
>>> 42 == 42.0
True
>>> 42 == '42'
False
```

Boolean Operators

Boolean operators work like you'd expect

- AND
- OR
- NOT

In order of precedence: NOT, AND, OR

```
>>> (4 < 5) and (5 < 6)
True
>>> (4 < 5) and (9 < 6)
False
>>> (1 == 2) or (2 == 2)
True
```

IF Statements

The general form of IF statements is:

```
If <condition>:
   <clause>
elif <condition>:
   <clause>
else:
   <clause>
```

A condition is anything that evaluates to True or False. A clause is the action to be performed.
A variable MUST be initialized prior to using it in an evaluation. Meaning it must have been given some value.
The order of statements matter as they are processed top to bottom. elifs may not execute but else will if nothing else does.

Python Blocks

Python uses blocks

- Blocks begin when indentation increases
- Blocks can contain other blocks
- Blocks end when indentation decreases to zero or to a containing block's indentation

```
if name == 'Mary':
    print('Hello Mary')
if password == 'swordfish':
    print('Access granted.')
else:
    print('Wrong password.')
```

Try this as a script

```
my_var = input('Enter a number greater than zero: ') #Nicer way of doing input
if int(my_var) < 5:
    print ('Less than five')
elif int(my_var) < 10:
    print ('Less than ten but greater than five')
elif int(my_var) <= 20:
    print ('Between 10 and 20')
else:
    print ('Larger than 20')
```

WHILE Loops

The general form of While statements is:

while <condition>:
  <clause>

```
spam = 0
while spam < 5:
    print('Hello, world.')
    spam = spam + 1
```

Spam here is being used as a counter

A Cleaner Way of Using Counters

Table 4-1: The Augmented Assignment Operators

Augmented assignment statement	Equivalent assignment statement
`spam = spam + 1`	`spam += 1`
`spam = spam - 1`	`spam -= 1`
`spam = spam * 1`	`spam *= 1`
`spam = spam / 1`	`spam /= 1`
`spam = spam % 1`	`spam %= 1`

While True

This code will run forever (or until you ctrl-C stop it)

```
while True:
    name = input('Please type your name: ')
    print(name)
```

Note that True has to be capitalized. You could do the same thing with while 1==1, or any statement that will always evaluate to True

Break

Break stops the loop

```
while 1==1:
    name = input('Please type your name: ')
    print(name)
    break
```

Continue

Continue forces program flow to the condition check at the top of the block

```
while True:
    name=input('Who are you? ')
    if name != 'Joe':
        continue
    else:
        print('Hello Joe.  What is the password?')
        password=input()
        if password == 'swordfish':
            print('Access Granted')
            break
        else:
            print('Access Denied')
```

FOR Loops

BREAK and CONTINUE work in FOR loops, where continue goes to the next value and break stops the loop. Break and continue only work in While loops and For loops.

```
for i in range(5):
    print (i, i*2)
    if i*2 > 4:
        break
```

FOR Loops

Note: Range can take three values: (start, stop, step). Negative values can be used.

```
for i in range(5,-5, -1):
    print (i)
```

Program Termination

If you import sys, you can stop a program with sys.exit()

```
import sys

while True:
    print('Type exit to exit')
    response = input()
    if response == 'exit':
        sys.exit()
    print('You typed ' + response + '.')
```