«««< HEAD ======= »»»> upstream/master

THE ONLY SHIBBOLETH THE WEST HAS IS SCIENCE. IT IS THE PREMISE OF MODERNITY AND IT DEFINES ITSELF AS A RATIONALITY CAPABLE OF, INDEED REQUIRING SEPARATION FROM POLITICS, RELIGION AND REALLY, SOCIETY. MODERNISATION IS TO WORK TOWARDS THIS.

BRUNO LATOUR

THE BOUNDARY BETWEEN SCIENCE FICTION AND SOCIAL REALITY IS AN OPTICAL ILLUSION.

DONNA HARAWAY

CSEF

INTRODUCTION TO PYTHON

THE STUDENT ACADEMY

Contents

List of Figures

List of Tables

The longest snake ever held captive is Medusa,
a reticulated python (python reticulatus).
On 12 October 2011, she was measured at
7.67 m long.

Note

This physics text is an OpenSource academic project developed in abstraction at The Academy. The manuscript is written in LaTeX and makes use of the `tufte-book` and `tufte-handout` document classes.

http://latex-project.org/ftp.html
https://git-scm.com/downloads

«««««< HEAD

Introduction to OSX Terminal

Terminal (Terminal.app) is the terminal emulator included in the OS X operating system by Apple. (Wikipedia)

A terminal emulator is an application that allows access to the operating system. This emulator can be used to perform multiple tasks; it can perform from simple tasks like printing out text, to performing more complex tasks like uploading files to the cloud.

Basic Commands:

break : Exit from a For, While, Until or Select loop. cd : Change Directory chmod : Change access permissions clear : Clears terminal screen curl : Transfer data from or to a server

======= «««« HEAD

Array 1-D, 2-D, 3-D

Intro

Array is like a storage, it can fill with string or integer. In 1-D, 2-D it can also represents the x-axis and y axis.

Creating arrays

Arrays is created buy blanket.

Example:

```
a=[ ]        *a is the array name that you want.
```

The things in the [ ] and be store and when you want to access it you will need its position in the array and type like a[0]

Example:

```
a=["apple","orange","banana"]
```

If you want to print banana form the array, you may want to type

```
print a[2]
```

Filling arrays

Everything can be store in the array, strings, integers, arrays. When you create an array you can fill things in it as the default things that the array have.

Example:

```
a=["Billy","Bud",90,60,50]
b=["Anne","Chow",90,95,100]
c=["Jen","Bo",60,80,90]
```

If you want to add things into the array that u create already, you can use

```
array_name=array_name+[Things you want to add]
```

Example:

```
a=[apple]
```

and now I want to add orange into it, so we add

```
a=a+[orange]
```

To create 2-D or more array we need to create array in the nested for-loop.

Example:

```
a=[]
for i in range(N): *N how long you want the array to be
    b=[]    *This is a temporary array to generate every array inside the main array.
    for j in range(N):
        *Things you want to put in the array by b=b+[ ]
    a=a+[b]     *Here put the temporary array back to the main array.
```

Traversing array

Traversing array is visiting each element in the array and do something. In 1-D we can do it with for loop to identify things in array.
Example:

```
a=[1,2,3]
for i in range(len(a))    *len(a) =  Numbers of elements in the array
    *Things put here can edit the specific element a[i]
```

In 2-D we start using nested for-loop to identify the x-axis and y-axis.
So we use nested for-loop to traversing it too.

Example:

```
a=[[0,1],[0,0],[0,1]]
for i in range(len(a)):
    for j in range(len(a)):
        *Things put here can edit the specific element a[i][j]
```

In 3-D we use more for-loop to identify the more dimension.

Example:

```
a=[[[0,0],[0,0]],[[0,0],[0,0]]]
for x in range(len(a)):
    for y in range(len(a)):
        for z in range(len(a)):
            *Things put here can edit the specific element a[x][y][z]
```

======= »»»»> upstream/master »»»»> upstream/master

While Loops

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.When the condition becomes false, program control passes to the line immediately following the loop.In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

whileloop.png

Figure 1: Flow diagram about how the while loop works

Python code example

```
count = 0
while (count < 9):
   print 'The count is:', count
   count = count + 1

print "Good bye!"
```

Output

```
>>>
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
>>>
```

The code will produce the following output.

Infinite Loop

A loop becomes infinite loop if a condition never becomes **FALSE**. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as and when required.

```
var = 1
while var == 1:
   num = raw_input("Enter a number  :")
   print "You entered: ", num

print "Good bye!"
```

This python code is an example of how infinite loop can be created.

```
Enter a number  :X
You entered:  x
Enter a number  :Y
You entered:  Y
Enter a number  :Z
You entered:  Z
Enter a number between :
```

This code creates an infinite loop where it will need your input of any number.Once you input any number, it will output it like if you input "X" it will show back "X".

To break the loop you will either need to add the "break" command in your code OR press **CTRL+C** to exit the program.

Using else statements with while loops

Python supports to have an else statement associated with a loop statement.

If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

```
count = 0
while count < 5:
   print count, " is  less than 5"
   count = count + 1
else:
   print count, " is not less than 5"
```

The following code illustrates the combination of an else statement with a while statement that prints a number as long as it is less than 5, otherwise else statement gets executed.

```
>>>
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
>>>
```

This is the output in python when the code above is executed.