Python Fundamentals

Programming Lecture 1

Nicholas Dwork

Text Files

Computer code is written in text files.
A text file often has the extension *.txt.

Computer code text files often have other extensions:
These extensions indicate the language
***.py - python file**
***.c - c file**
***.cpp - c plus plus file**
***.js - javascript**
***.html - webpage**

Note: *.doc is not a text file

Comments

A comment starts with the # symbol.

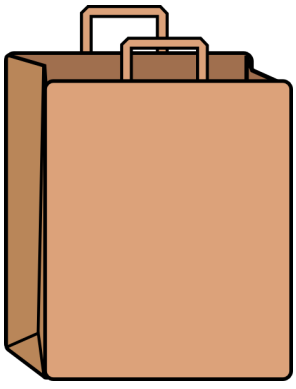
A comment does absolutely nothing. It's just a note from the programmer

Ex:

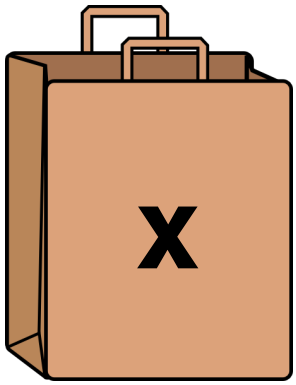
```
#  Hi fellow coder, this is a comment
```

Variables

A variable is a container that you can hold values.



Variables have names. You can put a value in the container with the equal sign.



x = 8

The thing on the left gets the value on the right.

Types of Variables

Numbers: x = 8

Lists: x = [2,3,5,7,9,11,13,17]

Characters: x = ‘a’
x = ‘x’

Strings: x = “This is a string”

Note: a string is a list of characters;

We can then use that variable in expressions

```
x = 8
y = x + 2
```

In the above example, y gets the value 10.

Lists

`x = [2,3,5,7,11,13,17]` **Make the list explicitly.**

`x = range(1:9)` **Makes the list [1,2,3,4,5,6,7,8]**

Accessing List Elements

x = [2,3,5,7,9,11,13,17]

You can get individual elements by using []

The value of x[0] **is 2.**
The value of x[2] **is 5.**
The value of x[6] **is 13.**

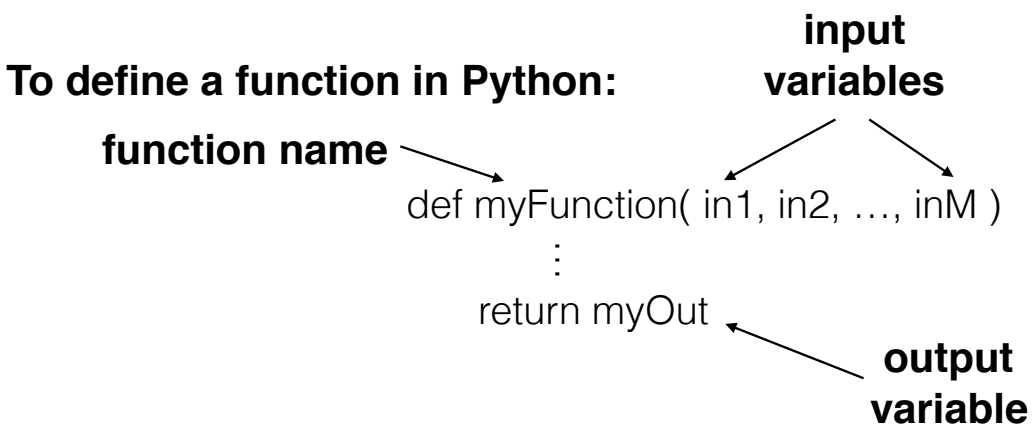
y = x[2:5] **y is the list [5,7,9]**

z = x[5:] **z is the list [11,13,17]**

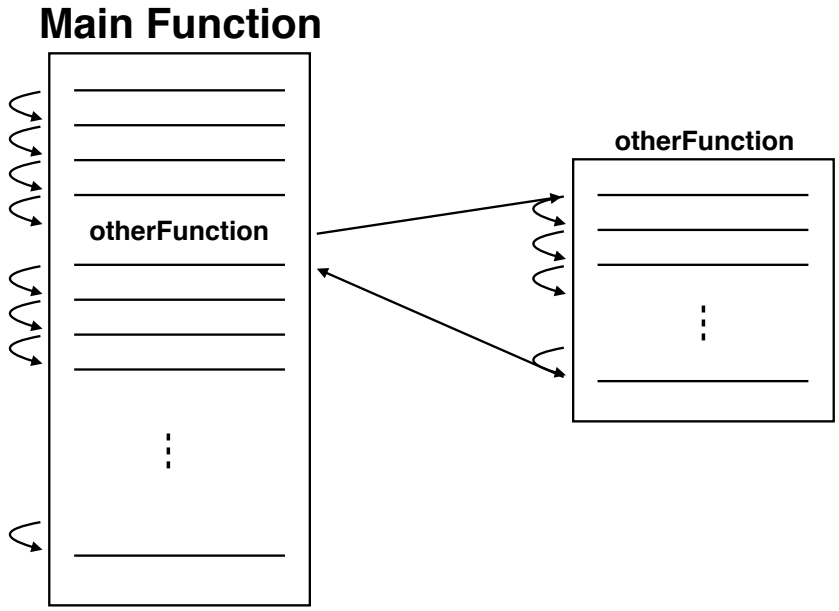
Functions

A function is a programming machine
You input stuff
You get something out

To define a function in Python:



Flow



Calling a Function Example

```
x = 8
y = x + 2
print(y)
```

Here, the function *print* is called with input argument *y*.

Scope

A variable only exists inside the function where it is created.

```
def main()
   x = 10
   y = 20
   z = myFunc( x, y )
   print(z)
```

```
def out = myFunc( a, b)
   print(x)
   out = a + b
   return out
```

The value displayed is 30.

The slashed line causes an error; x does not exist there.

Comparison Operators

Return boolean (True or False)

== **Tests to see if two expressions are equal**
Ex: a == b

!= **Tests to see if two expressions are not equal**
Ex: a != b

Return boolean (True or False)

`>` **Tests to see if the thing on the left is greater than the thing on the right.**

Ex: a > b

`>=` **Tests to see if the thing on the left is greater than or equal to the thing on the right.**

Ex: a >= b

Return boolean (True or False)

`<` **Tests to see if the thing on the left is less than the thing on the right.**

Ex: a < b

`<=` **Tests to see if the thing on the left is less than or equal to the thing on the right.**

Ex: a <= b

If … Then … Else

```
if a==1:
    print('a is equal to 1')
else:
    print('a is not equal to 1')
```

Only branch satisfying condition is executed.

For Loops

```
for i in range( first : last+1 )
    # do something here
end
```

For each value of i**, the code inside the loop gets executed.**

```
x = [ 1, 2, 3, 4, 5]

for i in range(0,3)
  print( x[i:i+3] )
end
```

This code displays [1;2;3] then [4;5;6] then [7;8;9].