

Introduction to Python

Genome 373

Genomic Informatics

Elhanan Borenstein

Why Python?

- **Python** is
 - easy to learn
 - fast enough
 - object-oriented
 - widely used
 - fairly portable
- **C** is much faster but much harder to learn and use.
- **Java** is somewhat faster but harder to learn and use.
- **Perl** is a little slower and a little harder to learn.

Your first program

- Create a file containing one line:

```
print "hello, world!"
```

- Be sure that you end the line with enter.
- Save the file as "hello.py" in your present working directory.
- In your terminal, type "python hello.py"

```
>python hello.py  
hello, world!
```

(This tells the computer "use python to run the program hello.py".
Yes, the result is somewhat anticlimactic.)

Objects and types

- We use the term object to refer to any entity in a python program.
- Every object has an associated type, which determines the properties of the object.
- Python defines six main types of built-in objects:

Number	10 or 2.71828	a list of numbers	a list of strings
String	"hello"		
List	[1, 17, 44] or ["pickle", "apple", "scallop"]		
Dictionary	{"food" : "something you eat", "lobster" : "an arthropod"}		
Tuple	(4, 5) or ("homework", "exam")		
File	we'll talk about this one later...		

notice the different symbols used to define each type

- Each type of object has its own properties, which we will learn about in the next couple of weeks.

Variables

- A variable is a name for an object.
- For example, we can assign the name "pi" to the Number object 3.14159, as follows:

```
pi = 3.14159  
print pi
```

```
>python prog.py  
3.14159
```

- You can also think of variables as boxes. Here, we put the object 3.14159 in a box named pi.

Assignment operator

```
pi = 3.14159
```

The '=' means assign the value 3.14159 to the variable `pi`. (it does NOT assert that `pi` equals 3.14159)

```
pi = 3.14159  
pi = -7.2  
print pi
```

you can see where
"variable" comes from -
pi can be changed

```
>python prog.py  
-7.2
```

The `import` command

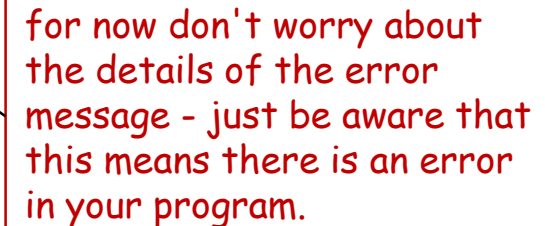
- Many python functions are available only via "packages" that must be imported (other functions are always available - called "built-in").

```
>>> print log(10)
Traceback (most recent call last):
  File <pyshell#0>", line 1, in <module>
NameError: name 'log' is not defined
```

```
>>> import math
>>> print math.log(10)
2.30258509299
```

```
>>> print log(10)
Traceback (most recent call last):
  File <pyshell#0>", line 1, in <module>
    print log(10)
NameError: name 'log' is not defined
```

for now don't worry about the details of the error message - just be aware that this means there is an error in your program.



The command line

- To get information into a program, we can use the command line.
- The command line is the text you enter after the word "python" when you run a program.

```
> python my-program.py 17
```

- The zeroth argument is the name of the program file.
- Arguments larger than zero are subsequent elements of the command line, separated by spaces.

zeroth
argument

first
argument

Reading command line arguments

Access in your program like this:

```
import sys
print sys.argv[0]
print sys.argv[1]
```

zeroth
argument

first
argument

```
> python my-program.py 17
my-program.py
17
```

There can be any number of arguments, accessed by sequential numbers (`sys.argv[2]` etc).

Class problem #1

- Write a program called "print-two-args.py" that reads the first two command line arguments after the program name, stores their values as variables, and then prints them on the same line with a colon between.

```
> python print-two-args.py hello world  
hello : world
```

Hint - to print multiple things on one line, separate them by commas:

```
>>> print 7, "pickles"  
7 pickles
```

Solution #1

```
import sys
arg1 = sys.argv[1]
arg2 = sys.argv[2]
print arg1, ":", arg2
```

assign the first
command line argument
to the variable `arg1`

print the value of
this variable

print the literal
string

