



Lab 9

Python Intro

- What stays the same and what changes

More advanced Python

Python Intro

- Main Differences:
 - True and False are capitals
 - Python floors (always down) with int division (matters with negatives): $-3 / 2 = -2$
 - No variable declarations (automatically interprets based on what you assign it to)
 - `my_string = "hello"`
 - Python has no ++ operator
 - Review lecture slides for more details
 - **Use `str[-1]` to access last element in string**

New things in Python

- `raw_input()` is how you extract from the input stream
- Raise to a power using `**`
- Concatenation using `+`
- Multiplication using `*`
 - Works on strings (see lecture slides!)
- **`print`** is like `cout` but no `'<<'` needed
 - No `'endl'` needed to print a new line
 - Add a comma to force output on the same line
 - This comma adds a space between the two outputs

Python: common syntax

- No semicolons!
- No curly braces!
- No parentheses for loops and conditionals!
- Colons are used in python
- **Everything is decided by indentation**
 - determines scope as well

```
if x == 35:  
    return True
```

```
for index in my_str:  
    #do something iterating through my_str
```

if, elif, else

- Very similar, all logic is the same
- ! → not
- && → and
- || → or

```
name = "Castiel"
if name == "Dean" or name == "Sam":
    print "hunter"
elif name == "Castiel":
    print "angel"
else:
    print "demon"
```

raw_input()

- Used for reading in from the user
- Reads until it hits an <enter> (**like getline**)
- Ignores leading and trailing white space
- Can directly store into a variable (remember: no type declaration!)

```
print 'Please enter your name: ' # prompt  
your_name = raw_input()
```

~or~

```
your_name = raw_input('Please enter your name: ')
```

Printing out variables

```
name = 'Fred'  
print 'Hi there %s!' %name
```

%s is for string

%d is for int (double will get truncated)

%f is for floats (if you want to save the decimals)

print Examples

```
print 'One'
print 'Two'
print 'Three',
print 'Four'
print 'Five' 'Six', 'Seven'
```

Console

```
One
Two
Three Four
Five Six Seven
```

- Note that print goes to a new line by default
- A comma adds a space and can also specify staying on the same line

Loops

```
while condition:  
    #do something
```

```
for <variable> in <container>:  
    #do something
```

- variable can be anything (make a new one)
 - Takes on the value of, or refers to, each successive member in the container
- Container is any type that holds other values
- You can't modify the elements of the container

```
for index in my_str: #loops through each char in my_str
```

Looping through a container

```
name_list = ['Erin', 'Leah', 'Austin', 'Maxim']  
for name in name_list:  
    name = 'bestTA'
```

What will name_list look like after this code runs?

Looping through a container

```
name_list = ['Erin', 'Leah', 'Austin', 'Maxim']  
for name in name_list:  
    name = 'bestTA'
```

What will name_list look like after this code runs?

```
['Erin', 'Leah', 'Austin', 'Maxim']
```

range()

- This function creates a list of values in the requested range (creates indices for your use)
- `range(n)` creates a list of values from 0 to $n-1$
 - It can also accept up to 3 parameters
 - 1 param represents (stop)
 - 2 params = (start, stop)

`range(start, stop, step)`

start: Starting number of the sequence.

stop: Generate numbers up to, **but not including** this number.

step: Difference between each number in the sequence.

<http://pythoncentral.io/pythons-range-function-explained/>

len()

- This function returns the number of elements in a list
- Use `range()` and `len()` to loop over a list when you want to **change values in the container**

```
my_list = ['hello', 'world' , 'it', 'is', 'me']  
for k in range(len(my_list)):  
    my_list[k] = 'x'  
    print my_list[k]  
#will make every word in the list an 'x'
```

User-defined functions

```
def add(a, b):  
    sum = a + b  
    return sum
```

No return type
necessary!

```
def main():  
    x = 3  
    y = 5  
    sum = add(x, y)  
    print sum
```

← function call

User-defined functions

```
def add(a, b):  
    sum = a + b  
    return sum
```

No return type
necessary!

```
def main():  
    x = 3  
    y = 5  
    sum = add(x, y)  
    print sum
```

← function call

Output:
8

Python: passed by object reference

- ◉ In python, variables are **neither** passed by value nor passed by reference
- ◉ Variables are just names that refer to objects
 - ◉ Anytime a variable is set to an object, it is said to refer to that object (in memory)
 - ◉ Look at lecture slides for more details!
 - ◉ <http://robertheaton.com/2014/02/09/pythons-pass-by-object-reference-as-explained-by-philip-k-dick/>

Lists

- You can think of a list as an array that can also:
 - Have different data types in one list
 - Add elements to it, increasing length
 - Start out with any number of elements (no need to declare or decide on a size)
- You can access elements with brackets `[]` just like with an array

Lists: which are valid?

```
my_list = []
```

```
my_list = [73, 1, 33]
```

```
my_list = ["name", 'hi', 3, 4.0]
```

Lists: which are valid?

```
my_list = [] #empty list
```

```
my_list = [73, 1, 33]
```

```
my_list = ['name', "hi", 3, 4.0]
```

ALL ARE VALID!

- Notice use of "" and ', and different types in the same list

Slicing

- A slice is a way to specify a portion of a container (list, tuple, string)
- Format:
 object[start:end]
- Other format options:
 - object[1:] #means from index 1 to the end
 - object[:5] #means from first index up to (but not including), index 5 (or through index 4)
 - Negative index = relative to the back end
 object[-3:-1] #means third to last index up to but not including last index

Slicing question

```
text = 'review discussion!'
print text[-4:]
```

What does this print?

Slicing question

```
text = 'review discussion!'
print text[-4:]
```

What does this print?

ion!

Summary

- The while loop is familiar
- The for loop is always over some container
- You can use range() to create a list of indices
- Lists and strings can be looped over in the same way
- You can slice strings and arrays into pieces
- A list can contain different types
- A list of lists can work as a 2D array

Answer to a good question

In Python if you type:

```
print "Leah" print "me"
```

All on one line what prints?

This is a compile error.

Answer to a good question (2)

If you pass in variables to a function in Python but they aren't the types the function is expecting, for instance:

```
def sum(x,y):  
    sum = x + y  
    return sum
```

And you pass it 3 and "yes"

What will happen?

You will get runtime type error which fires inside sum when it tries to apply the + operator to objects that are not the same type.

TypeError: cannot concatenate 'str' and 'int' objects

Answer to a good question (3)

- ◉ Using `range` and `len` together allows you to modify the container
- ◉ `range()` gives you indices to use to loop over a container
- ◉ `len()` gives you the number of elements in the container
- ◉ But this isn't true with a string: strings are **immutable** (cannot be changed)
- ◉ To modify a string you need to make a new string variable and append what you want from the original string to it

Notes on a good question (4)

- You cannot modify the container at all in a normal for **var** in **container** loop
- You CAN modify the variable
- But modifying the variable that is holding each value of the container **WILL NOT** modify the container

Review Lecture Slides for Python!

- For python:
 - you should become familiar with the new things you can do with it
 - If you are working on CreativeAI you will be writing python code
 - Otherwise, it's great to be familiar with another coding language
- Feel free to ask any questions
 - It can be confusing at first, but it comes very easily as you keep working with it

Python practice question

- define a main function
- Read into a variable `pswr` from the user, printing the prompt: "password please: "
- Concatenate a '2' and an '&' to the end of `pswr`
- Print out each letter of the password, separated by a space using a loop
- Create a variable `hide_pswr`
- Set each letter in the `pswr` to x for `hide_pswr` and print this new hidden `pswr`

Python practice question solution

```
def main():  
    pswrd = raw_input("password please: ")  
  
    pswrd += "2&"  
  
    for k in pswrd:  
        print k,  
  
    hide_pswrd = ""  
  
    for j in pswrd:  
        hide_pswrd += 'x'  
  
    print hide_pswrd
```