

Python Basics

Whitespace matters! Your code will not run correctly if you use improper indentation.

String Formatting

Similar to printf() in C, uses the % operator to add elements of a tuple into a string

```
this_string = "there"
print "Hello %s!"%this_string returns "Hello there!"
```

Tuples

A tuple consists of a number of values separated by commas. They are useful for ordered pairs and returning several values from a function.

```
emptyTuple = ()
singleItemTuple = ("spam",) # note the comma!
thistuple = 12, 89, 'a'
thistuple = (12, 89, 'a')
thistuple[0] # returns 12
```

Ranges

```
>>> range(5)
[0, 1, 2, 3, 4]

>>> range(3,7)
[3, 4, 5, 6]

>>> range(3,-3,-1)
[3, 2, 1, 0, -1, -2]
```

List Comprehension

A special expression enclosed in square brackets that returns a new list.

[expression for expr in sequence if condition] The condition is optional.

```
>>>[x*5 for x in range(5)]
[0, 5, 10, 15, 20]

>>>[x for x in range(5) if x%2 == 0]
[0, 2, 4]
```

(With materials from Marc Destefano, Ph.D. RPI)

PRIMITIVE DATA TYPES

Data Type Example

```
NoneType None
int() 12
long() 12L
float() 12.0
bool() True False
str() "Hello" 'Hi'
```

OPERATORS

Arithmetic

```
+ add
- subtract
* multiply
/ divide
% modulo
```

Logical

```
and logical AND
or logical OR
not logical NOT
```

Comparison

```
< less than
<= less / equal
== equal
!= not equal
>= greater / equal
> greater than
```

INPUT / OUTPUT

```
raw_input() user enters text
print prints text
```

COMMENTS

```
# this is a comment
```

CLASSES

```
class Person(object):
    def __init__(self, n):
        self.name = n
p = Person("Steve Jones")
```

STRINGS

Ordered sequences of characters

```
s = 'Welcome'
s[3] # indexable
s[4:7] # slice notation
e = "" # empty is okay
len(s) # returns length of string
s + " sir!" # concatenation
s.find('e') # returns first index of 'e'
s.lower() # converts s to lowercase
s[3] = 'c' # ERROR. Immutable
```

LISTS

Ordered collections of items

```
colours = ['red', 'green', 'blue']
empty = []
mixed = [1, 1.4, 'haha']
nested = [[1,2], 2]
colours[2] # indexable
colours[2] = 'yellow' # mutable
colours.append('pink') # add to end
colours.sort() # alphanumeric sort
colours[1:] # slicing notation
```

DICTIONARIES

Contain key-value pairs. Unique, immutable keys

```
empty_dict = {}
scores = {"Abe": 10, "Bob": 5}
scores['Bob'] # 5 (indexable)
scores['Cody'] = 15 # add new pairs
scores.keys() # list of keys
scores.items() # list of key-value pairs
```

MEMBERSHIP

How to check if something is in a list:
item in list # returns True or False
In a dictionary, this only works on keys:
key in dict
For values:
value in dict.values()

CONTROL STRUCTURES

For Loop

```
loop over items in a list
for element in list:
    print element
for num in range(5):
    print num # prints 0 1 2 3 4
for num in range(3,6):
    print num # prints 3 4 5
```

While Loop

Repeat while a condition is true

```
while True:
    print "Infinite loop!"
while x < 5:
    x += 1 # increment

Conditional

if condition:
    print "condition is True"
elif condition2: # optional
    print "condition2 is True"
elif condition3: # optional
    print "condition3 is True"
else: # optional
    print "Nothing is True!"
```

FUNCTIONS

```
def fname(param1, param2): # definition
    print "this is the function body"
    return True # optional
fname(arg1, arg2) # function call
Function calls are expressions.
```

MODULES

import these to use their functions

```
import random
random.randint(1,6) # roll a die
import math
math.sqrt(8) # square root
math.cos(1) # cosine
from datetime import date
```