

Introduction to Python

- Welcome
- IDLE
- Expressions
- Statements
- Scripts
- Modules
- Defining functions

Welcome

- Your instructor: Jonathan Ellis
 - NOT the “spiritual director”
 - Former instructor at Neumont
 - Speaker, PyCon 06 and 07
 - President, Utah Python User Group
- Resources
 - <http://utahpython.org>
 - <http://groups.google.com/group/utahpython>
 - irc

IDLE

```
>>> 1 + 1
2
>>> x = 1 + 1
>>> x
2
>>> x + x
4
>>>
```

Scripts

- print

Function calls

```
>>> input('gimme a number: ')
gimme a number: 4
4
>>> x = input('gimme a number: ')
gimme a number: 4
>>> x
4
>>>
```

Exercise: squares

- Write a script that asks for a number, then prints its square

More function calls

```
>>> L = [1, 2, 3, 4, 5]
>>> sum(L) / len(L)
3
```

Lists

```
>>> 1 in L
True
>>> 1.5 in L
False
>>> L.sort(reverse=True)
>>> L
[5, 4, 3, 2, 1]
>>> L[0]
5
>>> L.pop()
1
>>> L
[5, 4, 3, 2]
>>> L.append(0)
>>> L
[5, 4, 3, 2, 0]
>>> L + [7, 8]
[5, 4, 3, 2, 0, 7, 8]
>>> L
[5, 4, 3, 2, 0]
```

Help

```
>>> help(L)
```

```
...
```

```
>>> help(L.append)
```

```
Help on built-in function append:
```

```
append(...)
```

```
    L.append(object) -- append object to end
```

```
>>> help(list.append)
```

```
Help on method_descriptor:
```

```
append(...)
```

```
    L.append(object) -- append object to end
```

Modules

- Reusable functions packaged up together
- Namespaces

```
>>> asctime()  
Traceback ...  
>>> from time import asctime  
>>> asctime()  
'Tue Apr 17 21:38:15 2007'  
>>> import time  
>>> time.asctime()  
'Tue Apr 17 21:39:31 2007'  
>>> time.time()  
1176867576.8280001  
>>> help(time.asctime)  
>>> help(time.localtime)  
>>> time.asctime(time.localtime(0))  
'Wed Dec 31 17:00:00 1969'
```

Tuples

- Bundle (possibly unrelated) data
- Immutable

```
>>> foo = (1, 2)
>>> bar = (1, 'asdf')
>>> foo + bar
(1, 2, 1, 'asdf')
>>> foo.extend(bar)
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#64>", line 1, in <module>
```

```
    foo.extend(bar)
```

```
AttributeError: 'tuple' object has no attribute 'extend'
```

```
>>> x, y = (1, 2)
```

```
>>> x, y = foo
```

```
>>> x
```

```
1
```

```
>>> y
```

```
2
```

if

```
>>> if x > 10: print "bingo"
```

```
>>> if x < 10 and x > 0: print "bingo"
```

```
bingo
```

Boolean expressions

```
>>> True
True
>>> 1 < 2
True
>>> 2 < 2
False
>>> 2 <= 2
True
>>> 2 in [1, 2]
True
>>> 2 == 2
True
>>> 2 = 2
SyntaxError: can't assign to literal
>>> if 2 = 2: print 1
SyntaxError: invalid syntax
```

for

```
>>> for i in range(5): print i
```

```
0  
1  
2  
3  
4
```

```
>>> range(5)
```

```
[0, 1, 2, 3, 4]
```

```
>>> L = ['foo', 'bar', 'baz']
```

```
>>> for st in L: print st
```

```
foo  
bar  
baz
```

Blocks

```
for i in range(10):  
    j = i * i  
    print '%d squared is %d' % (i, j)  
  
print 'the last value was', i
```

Ending loops early

```
for i in range(10):  
    j = i * i  
    print '%d squared is %d' % (i, j)  
    if i > 5:  
        break  
  
print 'the last value was', i
```

Exercise: guessing game

- Write a guessing game: give the player 5 tries to guess a number
 - Tell him if he guesses too low or too high
 - Congratulate him (and quit) if he gets it right

The Zen of functions

- Functions allow you to break up your problem into smaller problems, then combine the solutions
- Keep breaking them up until they are *easy* to solve / understand
 - Brian Kernighan: "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Defining functions

- (the “else:” is redundant)

```
def zero_to_ten(i):  
    if i >= 0 and i <= 10:  
        return True  
    else:  
        return False
```

Exercise: refactor guessing

- Make it look like this

```
for i in range(5):  
    guess = input("what's your guess? ")  
    if handle_guess(guess):  
        break
```

Strings

```
>>> st = "the quick brown fox jumped over the lazy dog"
>>> st = 'the quick brown fox jumped over the lazy dog'
>>> 'fox' in st
True
>>> st.split()
['the', 'quick', 'brown', 'fox', 'jumped', 'over', 'the',
'lazy', 'dog']
>>> "Madam, I'm adam"
"Madam, I'm adam"
>>> '''"Madam, I'm adam," he said'''
'"Madam, I\'m adam," he said'
```

Slicing

- All of these work on lists too

```
>>> st = "the quick brown fox jumped over the lazy dog"
>>> st[0]
't'
>>> st[0:3]
'the'
>>> st[3:]
' quick brown fox jumped over the lazy dog'
>>> st[-3:]
'dog'
>>> st[:-3]
'the quick brown fox jumped over the lazy '
```

Converting strings

```
>>> int('1')
1
>>> int('1.5')

Traceback (most recent call last):
  File "<pyshell#110>", line 1, in <module>
    int('1.5')
ValueError: invalid literal for int() with base 10: '1.5'
>>> float('1.5')
1.5
>>> eval('[1, 2]')
[1, 2]

# input() uses eval
```

Dicts

```
>>> children = {}
>>> children['jonathan'] = 2
>>> children['frank'] = 6
>>> children
{'frank': 6, 'jonathan': 2}
>>> d = {'jonathan': 2, 'frank': 6}
>>> d == children
True
>>> d is children
False
```

Dict operations

```
>>> d['jonathan'] = d['jonathan'] + 1
>>> d['jonathan'] += 1
>>> for name in d.keys(): print d[name]

6
4
>>> for name, kids in d.items(): print '%s has %d kids' %
(name, kids)

frank has 6 kids
jonathan has 4 kids

>>> help(dict)
```

Dealing with missing keys

```
>>> children['asdf']

Traceback (most recent call last):
  File "<pyshell#153>", line 1, in <module>
    children['asdf']
KeyError: 'asdf'
>>> 'asdf' in children
False
>>> children.get('asdf', 0)
0
```

Files

```
>>> f = file(r'c:\python25\README.txt')
>>> L = []
>>> for line in f: L.append(line)

>>> len(L)
1291
>>> L[0]
'This is Python version 2.5\n'
>>> L[0].rstrip()
'This is Python version 2.5'
```

Final exam (hard!)

- Print out how many times each “word” occurs in README.txt
 - A “word” is anything separated by spaces or newlines