Introduction to Python

Overview

- Variables
- Numbers
- Strings
- Lists
- Tuples
- Dictionaries
- Conditionals
- Loops
- Functions
- Resources

Variables

Values are assigned using an equal sign.

```
>>> a = 5
>>> b = 3
```

Variables

Must start with an alpha or underscore character.

```
>>> a1 = 5
>>> 1b = 5
File "<stdin>", line 1
1b = 5
 ^
SyntaxError: invalid syntax
```

Variables

Variables reference objects.

```
>>> a = 1
>>> b = []
>>> type(a)
<type 'int'>
>>> type(b)
<type 'list'>
```

Variables

Multiple variables can reference the same object.

```
>>> a = [1, 2]
>>> b = a
>>> print b
[1, 2]
>>> b[0] = 2
>>> print a
[2, 2]
```

Variables

Objects exist as long as there are references to it.

```
>>> a = [1, 2]
>>> b = a
>>> del a
>>> print b
[1, 2]
```

Numbers

Arbitrary Length Integers

```
>>> a = 1
>>> print a
1
>>>
>>> 10 ** 20
100000000000000000000L
```

Numbers

Floats

```
>>> b = 2.0
>>> print b
2.0
```

Operators

Mathematics

```
+ - * / ( ) ** %
```

Comparison

```
== != <= >= < >
```

Operators

Integer division is default

```
>>> 2/3
0
>>> 2.0/3
0.66666666666666666
>>> 2/3.0
0.66666666666666666
```

Strings

Quotes

```
a = 'He said, "Python is great."'
b = "Isn't Python great."
c = '''This has a "new line."
Isn't that awesome.'''
```

Strings

Unicode Strings

```
>>> print 'ASCII string.'
ASCII string.

>>> print 'Unicode\xffstring.'
Unicode?string.

>>> print u'Unicode\xffstring.'
Unicodeÿstring.
```

Strings

Raw Strings

```
>>> print 'Escape backslashes\n.'
Escape backslashes
.

>>> print r"Don't escape backslashes
\n."
Don't escape backslashes\n.
```

Strings

Immutable

```
>>> a = 'This is a string.'
>>> a[0] = 'Y'
TypeError: 'str' object does not
support item assignment
```

Lists

Lists are a numerically indexed collection of objects, with indices starting at zero.

```
>>> a = [0, 1, 2, 3, 4]
>>> a[0]
0
>>> a[1]
1
```

Lists

A single list can contain many different objects, and can be edited.

```
>>> a = [1, 'string1', 2.0]
>>> print a
[1, 'string1', 2.0]
>>> a[1] = 'new string'
>>> print a
[1, 'new string', 2.0]
```

Tuples

Very similar to lists except that it cannot be edited.

```
>>> a = (1, 'string1', 2.0)
>>> print a
(1, 'string1', 2.0)
>>> a[1] = 'new string'

TypeError: 'tuple' object does not
support item assignment
```

Dictionaries

Dictionaries are a mapping of key value pairs and are indexed on the keys.

```
>>> name = {'first': 'Stephen',
 'last': 'Haywood'}
>>> name['first']
 'Stephen'
>>> name['last']
 'Haywood'
```

Dictionaries

Keys must be immutable objects. Strings or tuples are typically used.

```
>>> l = [1, 2]
>>> d = {l: 'test'}
TypeError: unhashable type: 'list'
```

Dictionaries

A single dictionary can map to many different objects.

```
>>> d = {'str': 'string', 'int': 1,
'float': 2.0}
```

Conditionals

If ... else

```
a = 10
if a == 10:
    print a
else:
    print 'Not 10'
```

Conditionals

If ... elif ... else

```
if a % 2 == 0:
    print 'Not prime'
elif a % 3 == 0:
    print 'Not prime'
else:
    print 'Maybe prime'
```

Conditionals

Pass - Does nothing.

```
if a is True:
    pass
else:
    print a
```

Loops

For ... in

```
for w in 'word':
    print w

for i in xrange(4):
    print i

for item in ['list', 'of', 'items']:
    print item
```

Loops

For ... in

```
for item in ('list', 'of', 'items'):
    print item

name = {'first': 'Stephen', 'last':
'Haywood'}
for k in name:
    print k
    print name[k]
```

Loops

While

```
i = 0
while i < 10:
    i += 1

while True:
    print a
```

Loops

Break - Exits the current loop

```
i = 10
while True:
    i -= 1
    if i == 5:
        break
```

Loops

Continue - Moves to the next iteration of the loop.

```
for w in 'word':
    if w == 'r':
        continue
    print w
```

Functions

Use the keyword def.

```
def mod(a, b):
    s = a / b
    return a - b * s

mod(10, 2)
```

Functions

Variables are passed by reference.

```
a = 10
def mod(a, b):
    s = a / b
    return a - b * s

print mod(7)
```

Functions

Global variables can be read but not assigned to in a function.

```
a = 10
def mod(b):
    while a >= b:
        a -= b
    return a

print mod(7)
```

Functions

Global variables can be read but not assigned to in a function.

```
a = {'f': 'Stephen', 'l': 'Haywood'}
def update_name():
    a['f'] = 'Clyde'

update_name()
print a
```

Functions

All functions return a value.

```
>>> def do_nothing():
...     pass
...
>>> print do_nothing()
None
```

Resources

docs.python.org/2/tutorial/

docs.python.org/2/library/index.html

learnpythonthehardway.org/

www.codeacademy.com/tracks/python