Python Overview
You're flying! How?

I dunno... Dynamic typing? White space?
Come join us! Programming is fun again! It's a whole new world up here!

But how are you flying?

I just typed import antigravity
That's it?

... I also sampled everything in the medicine cabinet for comparison.

But I think this is the python.
Python?

Write down all the questions you have about python.

(Your response)
The essence of Python

Everything is an object.*

Every object has

- a value.
- a type.
- an identity.
- a namespace.

*but some objects (e.g., numeric and boolean literals) are “special”.

```python
>>> 1  # value
1
>>> type(1)  # type
<type 'int'>
>>> id(1)  # identity
140686900921016
>>> dir(1)  # namespace
['__abs__', ..., 'real']
```
### Some Python types and values

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>int</td>
<td>1, 10000</td>
</tr>
<tr>
<td>floating point</td>
<td>float</td>
<td>3.14, 1.</td>
</tr>
<tr>
<td>boolean</td>
<td>bool</td>
<td>True, False</td>
</tr>
<tr>
<td>string</td>
<td>str</td>
<td>'hi', 'hello', '''hola''', '''howdy'''</td>
</tr>
<tr>
<td>list</td>
<td>list</td>
<td>[], [0], [1, 1, 2, 3, 5, 8, 13]</td>
</tr>
<tr>
<td>tuple</td>
<td>tuple</td>
<td>(), (0,), ('Ben', 'Olin', 1279)</td>
</tr>
<tr>
<td>dictionary</td>
<td>dict</td>
<td>{}, {'a': 1}, {'a': 1, 'b': 2}</td>
</tr>
<tr>
<td>set</td>
<td>set</td>
<td>set(), {0}, {2, 3, 5, 7, 11}</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>function</td>
<td>(varies)</td>
<td>type, help, len, dir, locals</td>
</tr>
</tbody>
</table>
Reading from files

```
0,1,2,3,4,5,6,7,8,9
0,2,4,6,8
100,200,300,400,500,600,700,800,900
10,17,24,31,38,45,52,59,66,73,80,87,94
```

don('data.txt').read()
'0,1,2,3,4,5,6,7,8,9
0,2,4,6,8
100,200,300,400,500,600,700,800,900
10,17,24,31,38,45,52,59,66,73,80,87,94'

don('data.txt').readlines()
['0,1,2,3,4,5,6,7,8,9\n',
 '0,2,4,6,8\n',
 '100,200,300,400,500,600,700,800,900\n',
 '10,17,24,31,38,45,52,59,66,73,80,87,94\n']
Slicing a sequence

\[ \text{seq[start : end : step]} \]

think: \([start, end)\)

\[
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
\]

>>> range(10)[3:]
[3, 4, 5, 6, 7, 8, 9]

>>> range(10)[3:5]
[3, 4]

>>> range(10)[::2]
[0, 2, 4, 6, 8]
Functional programming in Python
List comprehensions
are syntactic sugar for functional programming concepts (e.g., map)

```python
lines = open('data.txt').readlines()

data = []
for line in lines:
    data.append(line[:-1])

data = map(lambda line: line[:-1], lines)

data = [line[:-1] for line in lines]
```
List comprehensions
are syntactic sugar for functional programming concepts (e.g., filter)

```python
positiveValues = []
for value in values:
    if value > 0:
        positiveValues.append(value)

data = filter(lambda value: value > 0, values)

data = [value for value in values if value > 0]
```
What does this code do?

```python
lines = open('data.txt').readlines()
data = [line[:-1] for line in lines]
values = [map(int, row.split(',')) for row in data]
results = map(sum, values)
```

```
0,1,2,3,4,5,6,7,8,9
0,2,4,6,8
100,200,300,400,500,600,700,800,900
10,17,24,31,38,45,52,59,66,73,80,87,94
data.txt
```
Assignment
What tha What?!
What tha what?! (guess, check, explain)

Tuple unpacking

```python
x = 0
y = 1
y, x = x, y
print x
print y
```
What tha what?! (guess, check, explain)

(Im)mutability

```
# list
l = [1,2,3]
l[0] = 0
print l

# tuple
l1 = [1,2]
l2 = l1
l1 += [3]
print l1
print l2

# tuple
l1 = (1,2)
l2 = l1
l1 += (3,)
print l1
print l2
```
def giveRaise(salaries, factor):
    for i in range(len(salaries)):
        salaries[i] *= factor

payGrades = range(50000, 100000, 10000)
print payGrades

result = giveRaise(payGrades, 1.5)
print result
print payGrades
def f(elem, l=[]):
    l.append(elem)
    return l

f(1)
f(2)
f(3)
Mutable default argument values, part 2

```python
def f(elem, l=None):
    if not l:
        l = []
        l.append(elem)
    return l

f(1)
f(2)
f(3)
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>int</td>
<td>1, 10000</td>
</tr>
<tr>
<td>floating point</td>
<td>float</td>
<td>3.14, 1.</td>
</tr>
<tr>
<td>boolean</td>
<td>bool</td>
<td>True, False</td>
</tr>
<tr>
<td>string</td>
<td>str</td>
<td>'hi', &quot;hello&quot;, ''hola'', &quot;&quot;&quot;howdy&quot;&quot;&quot;&quot;</td>
</tr>
<tr>
<td>list *</td>
<td>list</td>
<td>[], [0], [1, 1, 2, 3, 5, 8, 13]</td>
</tr>
<tr>
<td>tuple</td>
<td>tuple</td>
<td>(), (0,), ('Ben', 'Olin', 1279)</td>
</tr>
<tr>
<td>dictionary *</td>
<td>dict</td>
<td>{}, {'a': 1}, {'a': 1, 'b': 2}</td>
</tr>
<tr>
<td>set *</td>
<td>set</td>
<td>set(), {0}, {2, 3, 5, 7, 11}</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>function</td>
<td>(varies)</td>
<td>type, help, len, dir, locals</td>
</tr>
</tbody>
</table>

* mutable
The essence of Python

Everything is an object.*

Every object has

- a value.
- a type.
- an identity.
- a namespace.

```
>>> 1                 # value
    1

>>> type(1)          # type
    <type 'int'>

>>> id(1)            # identity
    140686900921016

>>> dir(1)           # namespace
    ['__abs__', ..., 'real']
```

*but some objects (e.g., numeric and boolean literals) are “special”.*
1st midterm

[Bar chart showing scores distribution with a box plot indicating the interquartile range]