Python 3

I learned it last night! Everything is so simple!

Hello world is just print("Hello, world!")

I dunno... Dynamic typing? Whitespace?

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.
There’s so much more to Racket

```racket
#lang web-server/insta
(define (start request)
  (response/xexpr
    '(html
      (head)
      (body
        (h1 "Behold, the platypus!")
        (img ((src "http://bit.ly/2dpeUzo"))))))
```

racket-lang.org  clojure.org
Racket is syntactic sugar for the λ-calculus.
\[ x \in \text{Identifiers} \]

\[ \text{expr ::= x} \]

\[ | \ (\text{lambda } (x) \ \text{expr}) \]

\[ | \ (\text{expr expr}) \]
Preface:
What follows will never be on a CS 42 assignment or exam.
(define (☐ x)
  (if x false true))

(define (☐ x y)
  (if x y false))

(define (☐ x y)
  (if x true y))
(define (NOT x)
  (if x false true))

(define (AND x y)
  (if x y false))

(define (OR x y)
  (if x true y))
(define NOT (λ (x) (if x false true)))

(define AND (λ (x y) (if x y false)))

(define OR (λ (x y) (if x true y)))
(define NOT (λ (x) 
  (if x FALSE TRUE)))

(define AND (λ (x y) 
  (if x y FALSE)))

(define OR (λ (x y) 
  (if x TRUE y)))

(define TRUE (λ (x y) x))
(define FALSE (λ (x y) y))
(define NOT (λ (x) (x FALSE TRUE)))

(define AND (λ (x y) (x y FALSE)))

(define OR (λ (x y) (x TRUE y)))

(define TRUE (λ (x y) x))
(define FALSE (λ (x y) y))
Python Overview
Prior experience: programming languages
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>
</tbody>
</table>
| string          | str    | 'hi', "hello", '''hola''', """howdy""
| list            | list   | [], [0], [1, 1, 2, 3, 5, 8, 13]                                            |
| tuple           | tuple  | (), (0,), ('Ben', 'Olin', 1279)                                           |
| dictionary      | dict   | {}, {'a': 1}, {'a': 1, 'b': 2}                                            |
| set             | set    | set(), {0}, {2, 3, 5, 7, 11}                                              |
| None            | None   | None                                                                       |
| function        | (varies)| type, help, len, dir                                                      |
What tha What?!
x = 0
y = 1
y, x = x, y
print(x)
print(y)
What tha what?! (guess, check, explain)

(Im)mutability

```python
l = [1,2,3]
l[0] = 0
print(l)

t = (1,2,3)
t[0] = 0
print(t)

l1 = [1,2]  # List is mutable
l2 = l1
l1 += [3]
print(l1)
print(l2)

l1 = [1,2]  # List is mutable
l1 += [3]
print(l1)

l1 = [1,2]  # List is mutable
l2 = l1
print(l2)

l1 = [1,2]  # List is mutable
l1 += [3]
print(l1)

l1 = [1,2]  # List is mutable
l2 = l1
print(l2)

l1 = [1,2]  # List is mutable
l1 += [3]
print(l1)

l1 = [1,2]  # List is mutable
l2 = l1
print(l2)
```
## 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>

* mutable
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”.

```
>>> 1  # value
1
>>> type(1)  # type
<intype 'int'>
>>> id(1)  # identity
140686900921016
>>> dir(1)  # namespace
['__abs__', ..., 'real']
```
Great Shakeout
Drop, cover, and hold on
## Evacuation Sites

### HMC Building and Residence Hall Evacuation Sites

- **Foothill Blvd.**
- **North**
- **Evacuate to nearest location**

<table>
<thead>
<tr>
<th>BUILDING/LOCATION</th>
<th>EVACUATION SITE</th>
<th>BUILDING/LOCATION</th>
<th>EVACUATION SITE</th>
<th>BUILDING/LOCATION</th>
<th>EVACUATION SITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Olin</td>
<td>1</td>
<td>Kingston</td>
<td>5</td>
<td>Linde Hall</td>
<td>11</td>
</tr>
<tr>
<td>Parsons</td>
<td>2</td>
<td>South and North Halls</td>
<td>5</td>
<td>Atwood Hall</td>
<td>12</td>
</tr>
<tr>
<td>Sprague</td>
<td>2</td>
<td>Hoch-Shanahan Dining Commons</td>
<td>5</td>
<td>Case Hall</td>
<td>12</td>
</tr>
<tr>
<td>Galileo</td>
<td>3</td>
<td>West Hall</td>
<td>7</td>
<td>Dorm 2015</td>
<td>10</td>
</tr>
<tr>
<td>Jacobs, Keck</td>
<td>3</td>
<td>Linde Activities Center</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beckman</td>
<td>3</td>
<td>Sontag Hall</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platt Campus Center</td>
<td>4</td>
<td>East Hall</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R. Michael Shanahan Center</td>
<td>4b</td>
<td>Garrett House</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for Teaching and Learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>