Hmmm  ≡  Prolog  ≡  Racket

Turing Machine  ≡  (proofs in) Predicate Calculus  ≡  λ Calculus

0 := qzn r0 0  p := p.

(λ(x)(x x))(λ(x)(x x))
The Essence of Python

Everything is an object.*

Every object has

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

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

*but some things (e.g., numeric and boolean literals) are “special”.*
<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>
Slicing a Sequence

$$seq[start : end : step]$$

think: $[start, end)$

```
>>> range(10)[3:]
[3, 4, 5, 6, 7, 8, 9]
>>> range(10)[3:5]
[3, 4]
>>> range(10)[::2]
[0, 2, 4, 6, 8]
```
Some Things to Try Out

Investigate operations for built-in types.

What kinds of dictionaries are not allowed?

Reverse a list using slicing notation.

Copy a list using slicing notation.

*How do you know you’ve made a copy?*

Implement factorial and fibonacci.

*recursively, tail-recursively, iteratively (i.e., with loops)*

Implement some things from HW1 & HW2.

Guess, check, explain the following...
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)

(I)mutability

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

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

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

t1 = (1,2)
t2 = t1
t1 += (3,)
print t1
print t2
```
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)
What tha what?! (guess, check, explain)

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)
```