Let's pretend we're a Python

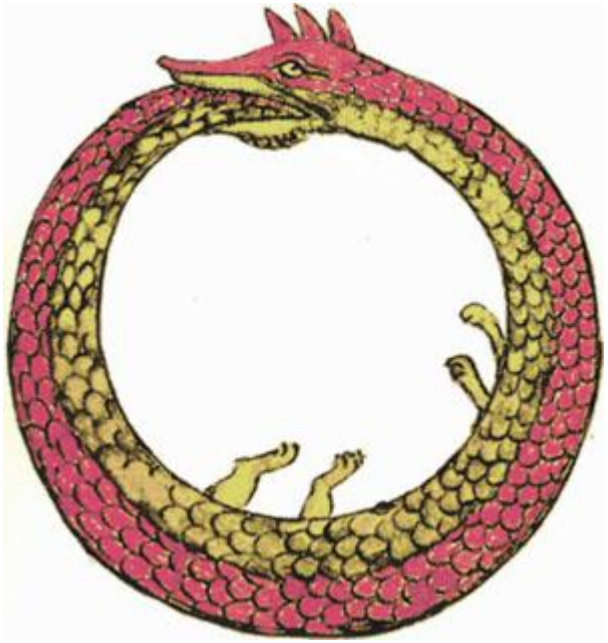
sssSSSsssSsssssssssssSSSSssssss

Order of operations

- (), making lists, making dictionaries
- lst[4], lst[2:5]
- function(x, y, z)
- -x
- x*y, x/y, x%y
- x+y, x-y
- x < y, x > y, x <= y, x >= y
- x == y, x != y
- x = 13
- x in [1, 2, 3], x not in [1, 2, 3]
- not a
- a and b
- a or b

How to evaluate things

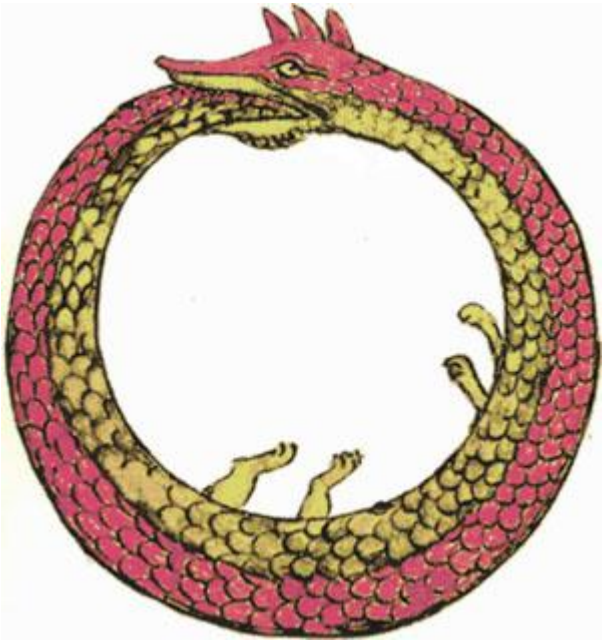
- Pick the thing that's lowest on the order of operations but don't do it yet
- Evaluate the arguments, and substitute in the results*
- Do the operation



* Some operators are special and don't evaluate all the sides -- go for the left side first

How to evaluate things

- Note that this procedure results in you actually **doing** the things higher in the order of operations first.



Example

5 < 10 and 3 + 4 * 2 >= 15

5 < 10

True

Example

5 < 10 and 3 + 4 * 2 >= 15

True

Example

True and 3 + 4 * 2 >= 15

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

3 + 4 * 2

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

3 + 4 * 2

4 * 2

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

3 + 4 * 2

8

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

3 + 8

Example

True and 3 + 4 * 2 >= 15

3 + 4 * 2 >= 15

11

Example

True and 3 + 4 * 2 >= 15

11 >= 15

Example

True and 3 + 4 * 2 >= 15

False

Example

True and False

Example

False

On looking up variables

- Every function call has its own table of variables called an **environment**
- Arguments to a function go in that function's environment
- Assignments within a function go in that function's environment
- If you can't find a variable in your local environment, you search the enclosing environment, until you get to variables that are defined in the file outside any function

Details of a function call

```
def fun(x, y):
    return x + y


a = 3
b = 4

print fun(a, b)
```

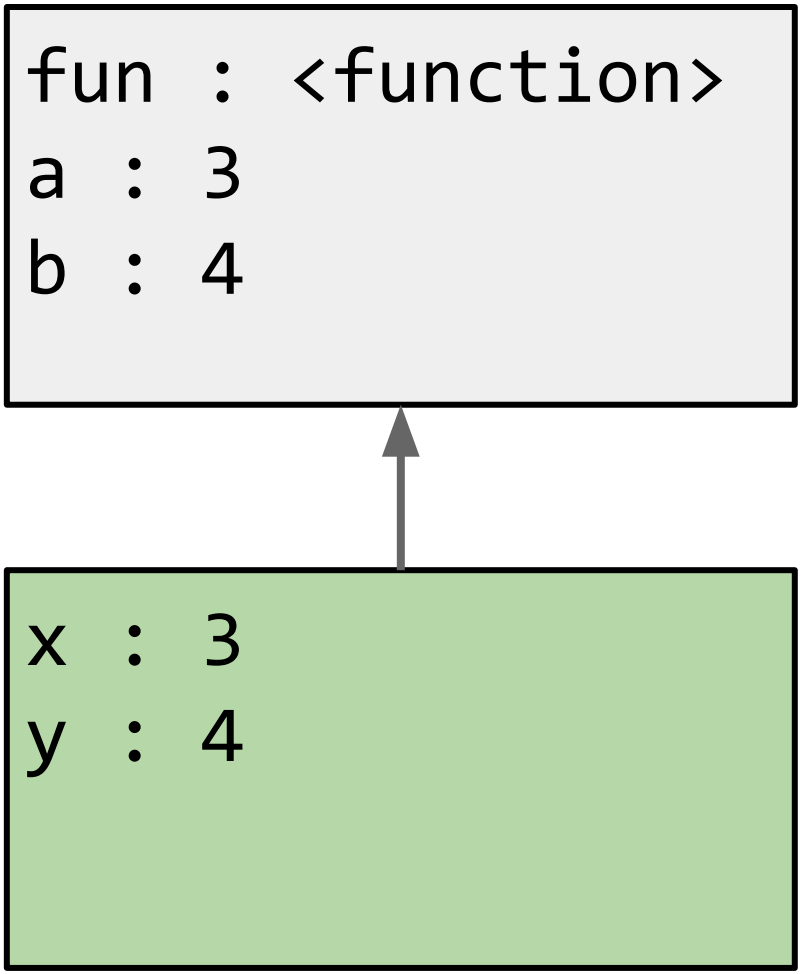
```
fun : <function>
a : 3
b : 4
```

Details of a function call

```
def fun(x, y):
    return x + y


a = 3
b = 4


print fun(a, b)
```



Details of a function call

```
def fun(x, y):
    return x + y


a = 3
b = 4

print fun(a, b)
```

```
fun : <function>
a : 3
b : 4
```

7

```
x = 3
def print1():
    print x

def print2(x):
    print x

def print3():
    x = 5
    print x
```

New operator: in

- 3 in [1, 2, 3, 4]
- 5 in [1, 2, 3, 4]
- "a" in "pants"
- "a" in "syzygy"
- d={"torso":"shirt","legs":"pants","foot":"shoe"}
- "legs" in d
- "pants" in d
- "a" in d

New operator: not in

- 3 not in [1, 2, 3, 4]
- 5 not in [1, 2, 3, 4]
- "a" not in "pants"
- "a" not in "syzygy"
- d={"torso":"shirt","legs":"pants","foot":"shoe"}
- "legs" not in d
- "pants" not in d
- "a" not in d

Some things you can do with strings

- **s.isalpha()** Return true if all characters in s are letters, and there is at least one character. Return false otherwise.
- **s.isalnum()** Return true if all characters in s are letters or numbers, and there is at least one character. Return false otherwise.

Some things you can do with strings

- **s.lower()** Return a copy of s with all letters lowercase
- **s.upper()** Return a copy of s with all letters uppercase
- **s.replace(old, new)** Return a copy of s with every occurrence of the substring old replaced with the substring new

More Practice

Hangman