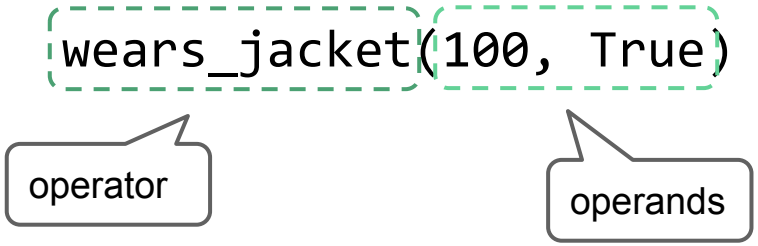
Functions Review: WWPD

```
print(print(“one”) and 2 and print(“tHrE3”))
```

Call Expressions

A call expression calls a function on its arguments.

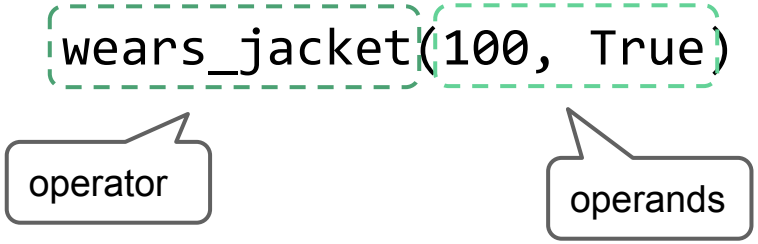
1.
2.
3.



Call Expressions

A call expression calls a function on its arguments.

1. Evaluate the operator to get a function.
2. Evaluate the operand(s) from left to right.
3. Apply the value of the operator on the value(s) of the operand(s).



Boolean Stuff in Python

False-y

-

Truth-y

-

Boolean Stuff in Python

False-y

- False
- 0
- None
- “ ”, [ ], { }

Truth-y

- True
- 1
- -1
- “Hello”
- Almost everything else

Boolean Logic

And

Or

Boolean Logic

And

- first false-y or last truth-y value (and stops evaluating there)
- “Are you free Saturday and Sunday?”

Or

- first truth-y or last false-y value (and stops evaluating there)
- “Are you free Saturday or Sunday?”

Discussion 1:

Caroline Lemieux (clemieux@berkeley.edu)
January 31, 2019

Slides adapted from Nancy Shaw's

Announcements

- **HW 1** due tonight
- **Lab 0 & Lab 1** due tomorrow
- **Hog project**
  - **Phase 1** due next Tuesday
  - Whole project due next Thursday (can work with partners)
- There are office hours tonight 6:30-8:00 if you need last minute help!

Agenda

- Concept check
- Announcements
- Anything *you* want to add to the agenda?
- Review of Functions & Control
- Environment Diagrams

Control

Control Review

```
n = 0
if n < 10:
    print(“1”)
elif n >= 0:
    print(2)
```

Control Review

```
n = 0
if n < 10:
    print(“1”)
elif n >= 0:
    print(2)
```

1

Control Review

```
n = 0
if n < 10:
    print(1)
if n >= 0:
    print(2)
```

Control Review

```
n = 0
if n < 10:
    print(1)
if n >= 0:
    print(2)
```

```
1
2
```

Control Review

```
n = 100
if n < 10:
    print(1)
print(2)
```

Control Review

```
n = 100
if n < 10:
    print(1)
print(2)
```

2

Control Review

```
n = 100
if n == 100:
    print(1)
print(2)
```

```
1
2
```

TRY 1.1

Iteration

```
while <cond>:
    <body>
```

- Keep evaluating body until <cond> is false-y
  - Should make sure it’s eventually false!

TRY 1.2 & 1.3

Summary

False Values:	True Values:	
False, 0, None, [ ], “ ”, { }	Everything else	
And: first false, last true value **Or:** first true, last false value		

```
if <cond>:
    ...
elif <cond>:
    ...
else:
    ...
```

Any number of these

Optional

```
while <cond>:
    ...
```

Keep evaluating body until False-y

Attendance

links.cs61a.org/caro-disc

Environment Diagrams

Assignment Statements

```
x = 10 % 4

y = x

x **= 2
```

Assignment Statements

1. Evaluate the right!
2. Var | Val

def Statements

```
def double(x):
    return x * 2

def triple(x):
    return x * 3

hmmm = double
double = triple
```

Try it!

def Statements

1. Create a function thingy (intrinsic name, parameters, and parent)
2. FUNC | ---------> to thingy

def Statements

1. Create a *function object* (intrinsic name, parameters, and parent)
2. FUNC | ---------> to object

- Note: function values are objects that are **POINTED POINTED POINTED** to
  - (only values are not pointed at. Objects which you will learn later and lists are pointed at as well)
- **DO NOT** evaluate the body of the function

def Statements

```
def double(x):
    return x * 2

def triple(x):
    return x * 3

hmmm = double
double = triple
```

Try it!

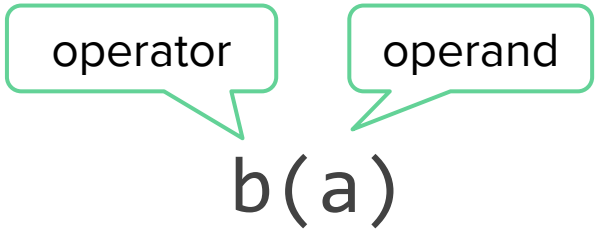
Call expressions

```
def double(x):
    return x * 2

hmmm = double
wow = double(3)
hmmm(wow)
```

Call Expressions

- Follow the golden rules of evaluation:
  - Evaluate operator
  - Evaluate operands
  - Apply operator to operands
- Call expressions create **new frames!**



Creating Frames

- Label frame # (f1, f2, f3)
- Label frame with function’s intrinsic name (the thing being pointed at)
- Label with the parent (defined earlier)

Call Expressions

- Bind parameters to arguments (what you pass in aka the stuff in the parentheses)
- Evaluate body using the golden rules
- At end, be sure to put the return value (default is None)

Call expressions

```
def double(x):
    return x * 2

hmmm = double
wow = double(3)
hmmm(wow)
```

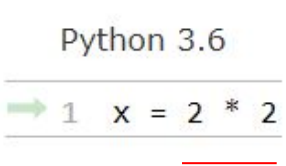
Try it!

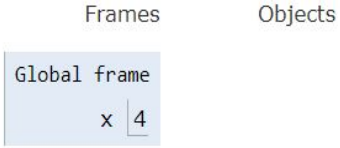
Lookups

- When trying to find the value of a variable:
  - If it's in your current frame, great!
  - If not, look in the parent of your frame, then in your parent's parent, and so on
  - If there are no more parents (you're in the global frame), it doesn't exist!

LET’S PUT IT ALL TOGETHER!

Assignment Statements:

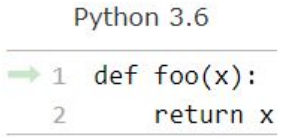


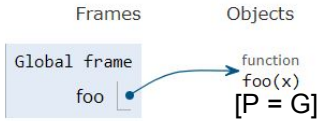


Evaluate RHS

Make binding in current frame

Def Statements:



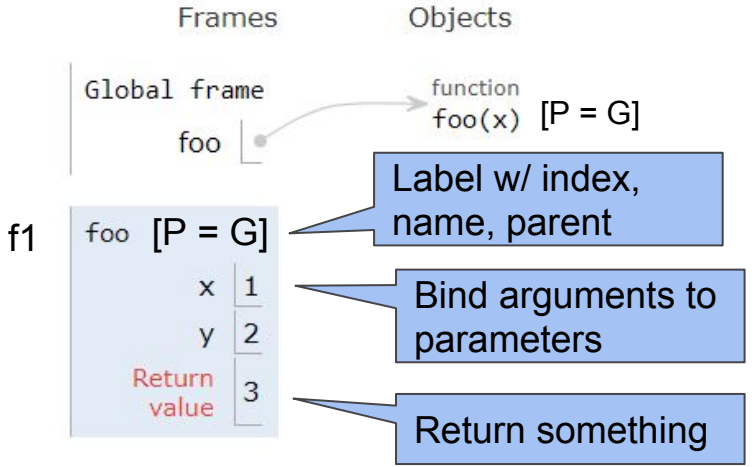


Don't go into body yet

Make binding in current frame

Call Expressions:

```
1 def foo(x):
2     y = x + 1
3     return y + x
4
5 foo(1)
```



2.4

```
from operator import add
def sub(a, b):
    sub = add
    return a - b

add = sub
sub = min
print(add(2, sub(2, 3)))
```

Common Misconceptions

When do you draw the pointer (vs not)?

Common Misconceptions

return f vs return f()

How do you know when you should call a function and need to open a new frame?

Common Misconceptions

f(a(2))

Which frame do I open first?

Function f or function a?

Common Misconceptions

```
x = 4
y = x
a = func
b = a
```

Why is 4 copied but why is func not?

(ie. why are there not two copies of func)