Discussion 1

Control and Environment Diagrams

Antonio Kam

`anto [at] berkeley [dot] edu`

Announcements

- HW2 releases on Monday
- Hog releases on Monday too
- Lab Checkoffs
- Come to lab in person and stay to the end and fill out attendance. If you don’t finish the required lab questions, you are required to finish them on their own (if you need help you should go to OH)
- Come to lab in person, finish all questions required AND optional, get attendance, and you can leave early.
- Finish all questions required and optional before lab starts, email, and get attendance.
- Pre-lab Checkoffs
  - Fill in `links.rouxl.es/finished`

Questions/comments from last section

- Please fill out the attendance form with stuff you want to show up here!
- (Put some fun stuff - I like answering fun questions)
  - Have had questions about games/music/etc. if that's the stuff you want to fill
- What are the logistics for discussions lab remotely?
  - You'll have to get an excused absence, and watch mega section

Temperature Check 🌡️

- Call Expressions (call expression order, what they look like)
- Values/Types (Integers, Strings, Floats, Booleans, etc.)
- Environment Diagrams
- Functions
- Control
  - `if`, `while`

Slides can all be found on `teaching.rouxl.es`

Control 🐈

Booleans

Falsey	Truthy
`False`	`True`
`None`	Everything else
`0`	
`[]`, `""`, `()`, `{}`	

This is Python specific! This table above doesn't necessarily apply to other languages (and later into the semester you'll see an example where the table above doesn't match the way the language works)

Boolean Operators

- `not`
    - Returns the *opposite* truthy value of the expression.
        - For example, if you type `not 0`, Python will return `True`
- `and`
    - Short circuits if it reaches a *falsey* value, and returns that value
        - *This is not necessarily `False`*
    - If all the values are truthy, the *last* value is returned
- `or`
    - Short circuits if it reaches a *truthy* value, and returns that value
        - *This is not necessarily `True`*
    - If all the values are falsey, the *last* value is returned.

Short Circuiting

- In call expressions, *everything* is evaluated from left to right, but this is not the case for when things short circuit.
- In `and` and `or` statements, all statements are not necessarily evaluated.
- `and` will keep on evaluating from left to right until it finds the first *falsey* value. If it finds a *falsey* value, it simply returns that value
- Think of this in an English sentence

Short Circuiting

- The last value will always be returned as is if you reach it:
  - For `or`, it will short circuit when it reaches the first truthy value, so if it sees only falsey values until the end, or will simply just return the last value as is.
  - Let's take a look at a smaller example:
    - False, **True** ⇒ **True**
    - False, **False** ⇒ **False**
    - Notice how the last elements are the same
    - This is why the last element is returned as is (it's very similar for and, except everything before is True)

Short Circuiting

Examples

- `1 and True and 1/0`
  - This will error 😭
- `1 and True and 0 and 1/0`
  - Returns 0 ✅
- `1 or True or 1/0`
  - Returns 1 ✅
- `0 or 1 or True or 1/0`
  - Returns 1 ✅

If Statements

```
if <condition>:
    <block of statements>
[elif <condition>:] # optional; short for 'else if'
    <block of statements>
[else:] # optional
    <block of statements>
```

- Don't forget the colons!
- `else` does not need a conditional
- You can chain together as many `elif` blocks as you want
- Evaluate all `if`s unless there's a `return` statement that ends the function
  - If you have a whole block of if/elif/else, you only evaluate at maximum 1 of the blocks.

Example (If Statements)

```python
n = 0
if n == 0:
    print("hi")
else:
    print("bye")
if n == n:
    print("0")
```

In this case, the console will output

```
hi
0
```

`return` in Functions

```
def box(x):
    return x + 2
    print(x + 3) # you will never reach this line

x = box(3)
x # 5
```

- `return` will prematurely exit a function, and will make Python evaluate that function call to whatever gets `returned`

Worksheet!

While Loops

```
while <condition>:
    <block of statements>
```

- A while loop allows for a repeated execution of a certain block of code, allowing you to write just one thing that will end up being executed multiple times.
- The condition is checked before the execution of each *iteration*.
- To avoid an infinite loop, you must make sure your while loop changes the variable in the condition

While Loops Examples

Example 1

```
n = 0
while n < 5:
    print(n)
    n = n + 1 # Without this line you will have an infinite loop!
print(n)
```

Output:

```
0
1
2
3
4
5
```

While Loops Examples

Example 2

```
n = 5
while n < 5:
  # Doesn't pass the condition on the initial loop
  # as a result, doesn't run any of the blocks
    print(n)
print(n)
```

Output:

```
5
```

Attendance

`links.rouxl.es/disc`

Worksheet!

Environments 🌎

Environment Diagrams

- Environment diagrams are a great way to learn how coding languages work under the hood
- Keeps track of all the variables that have been defined, and the values that they hold
  - Done with the use of *frames*
- Expressions evaluate to values:
  - `1 + 1` → `2`
- Statements do not evaluate to values:
  - `def` statements, assignments, etc.
- Statements change our environment

Frames

- The `Global Frame` exists by default
- Frames list bindings between variables and their values
- Frames also tell us how to look up values

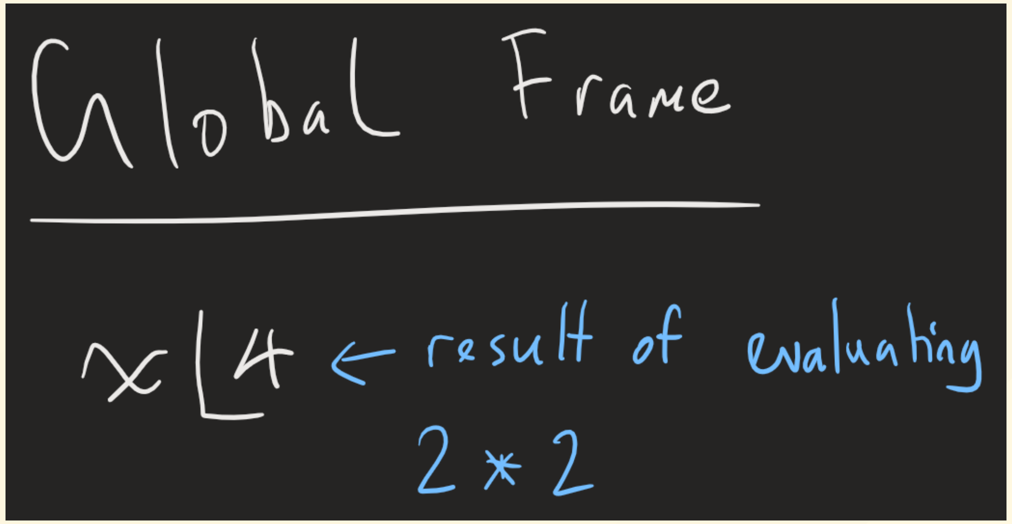
Assignment

- Assignment statements bind a value to a name
  - The right side is evaluated before being bounded to the name on the left
  - `=` is not the same in Python and mathematics
- These are then put in the *correct frame* in the environment diagram

```
x = 2 * 2 # 2 * 2 is evaluated before bound to the name x
```

Assignment

```
x = 2 * 2 # 2 * 2 is evaluated before bound to the name x
```



`def` statements

- Creates function (objects), and binds them to a variable name
- The function is **not** executed until called!
- Name of the variable is the name of the function
- Parent of the function is the frame where the function is *defined*
- Keep track of:
  - Name
  - Parameters
  - Parent

Worksheet!

Mental Health Resources

- CAPS:
  - If you need to talk to a professional, please call CAPS at 510-642-9494.
- After Hours Assistance
  - For any assistance after hours, details on what to do can be found at this link

Anonymous Feedback Form

`links.rouxl.es/feedback`

Thanks for coming! 🥳

Please give me feedback on what to improve!