# Discussion 01

## Control, Environment Diagrams

Aditya Balasubramanian
aditbala [at] berkeley [dot] edu

Slides available at `teaching.aditbala.com`

# Announcements 📣

- Last chance to change sections on sections.cs61a.org is 11:59pm Wed 9/14.
- Request to change to CS 10 at 2pm on Wed 8/31 (on Ed).
- Lab 1 is due 11:59pm Wednesday 8/31.
- Homework 1 is due 11:59pm Thursday 9/1.
- Lots of office hours are available to help you.
- Zoom access (go.cs61a.org/lecturezoom) to live lecture will be offered all semester.

# All Slides can be found on

`teaching.aditbala.com`

# Control

# Booleans

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

# Boolean Operators

- `not <conditional expression>`
  - returns opposite of `<conditional expression>`
  - `not (1 == 2)` -> `True`
- `<conditional expression> or <conditional expression>`
  - returns the first **Truthy** value it finds, `False` if none
  - `0 or None or 1` -> `1`
- `<conditional expression> and <conditional expression>`
  - return first **Falsey** value, or last value if everything is true
  - `40 and 0 and True` -> 0
  - `40 and 1 and True` -> True

# Short Circuiting

- Sort of like making an assumption
  - If I'm broke, then I don't need to check the price of boba since I'll never be able to buy it **lol** 😢
- `and` will stop at the first **Falsey** value and return it
- `or` will stop at the first **Truthy** value and return it
- Why is this important?
  - May not need to evaluate all expressions. Even if there is an expression that errors, e.g. `1/0`, `and` / `or` expression might short circuit before it reaches error

# Boolean Examples

- `0 or 435 or False`
  - returns `435`
- `True and "Hello" and 0`
  - returns `0`
- Short Circuiting
- `3 and 1/0 and False`
  - returns `Error`
- `3 and False and 1/0`
  - returns `False`

# If Statements

- How to use `<conditional expressions>` to execute/skip lines of code?

```
if <conditional expression>:
    <suite of statements>
elif <conditional expression>:
    <suite of statements>
else:
    <suite of statements>
```

- Colons after `if`, `elif`, `else` statements
- `else` doesn't need `<conditional expression>`

# If Statements Example

```
wallet = 0

if wallet > 0:
    print('you are not broke')
else:
    print('you are broke')
if wallet == 0:
    print(0)
```

# If Statements Example

```
wallet = 0

if wallet > 0:
    print('you are not broke')
else:
    print('you are broke')
if wallet == 0:
    print(0)
```

```
you are broke
0
```

# General Tips for Approaching Problems

- Do not immediately start coding
  - Ensure you understand the problem
  - Have an idea of what you want to code
- Groupwork
  - Bounce ideas off of each other!
  - Share any ideas, questions, or misconceptions
- Reading the problem
  - Please read the entire problem
  - Hints are very useful
  - Doctests are SUPER useful

# Worksheet

# While Loops

- How to execute a statement multiple times in a program?

```
while <conditional clause>:
    <statements body>
```

- program executes until `<conditional clause>` is false
- In other words, only run when `<conditional clause>` evaluates to `true`

# While Loop Examples

```
x = 3
while x > 0:
    print(x)
    x -= 1
```

# While Loop Example

```
x = 3
while x > 0:
    print(x)
    x -= 1
    # x = x - 1
```

```
3
2
1
```

# While Loop Example

- What is wrong with this while loop

```
x = 3
while x > 0:
    print(x)
```

- This will result in an infinite loop
- Make sure you are modifying the condition in the while loop

# Enviroment Diagrams 

# Enviroment Diagrams

- What are they?
  - A way to model how our program runs line by line
  - Keep track of variables, function calls and what they return, etc.
- Why use them?
  - Can help us understand where there is a bug in program (debugging)
  - Useful for other questions (WWPD, coding)
  - Exam points!

# Important Concepts

- Expressions
  - Evaluate to values
  - `1 + 1 -> 2`
- Assignment Statements
  - Bind (left side) **names** to (right side) **values**
  - **Names**
    - variable names
  - **Values**
    - Evaluate right side before binding
  - `x = 2 * 2`
  - `x -> 4`
  - doesn't return anything

# Frames

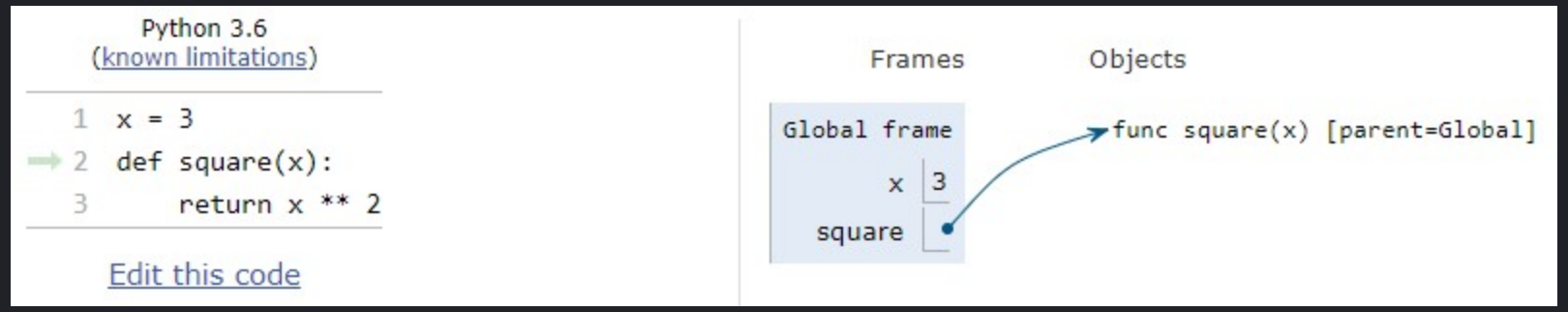
- `Global Frame` always exists
- Frames list the bindings of variables and their corresponding value
- Used to look up the value of a variable

# Question 7: Assignment Diagram

```
x = 11 % 4
y = x
x **= 2
```

# `def` statements

- `def` statements are used to bind **function objects** to a **variable**
- Only bind, **NO** execution until function is called
  - `def foo():` -> define function called `foo` with no parameters
  - `foo()` -> execute foo
- Binding name is function name
- Parent function is frame where function is defined
- Keep track of *name*, *parameters*, *parent frame*



# Thank you!

**Attendance ->** `teaching.aditbala.com`

**Anon Feedback ->** `https://tinyurl.com/adit-anon`