DS2000

10/8 - Tues.

Admin

- HW3 due 10/11 9pm
- HW4 at 10/11, due 10/19 9pm

Agenda

1. Functions, code structure
2. Calling functions
3. Python

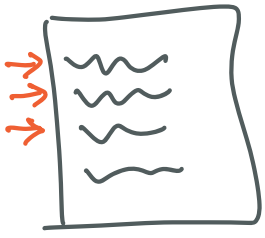
} Putting FUN into functions! :)

1. Functions, Code Structure

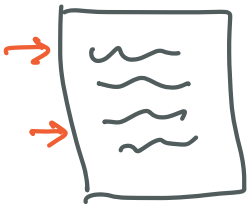
A function...

- is a mini program
- is standalone
- has one job
- returns one thing (or nothing)

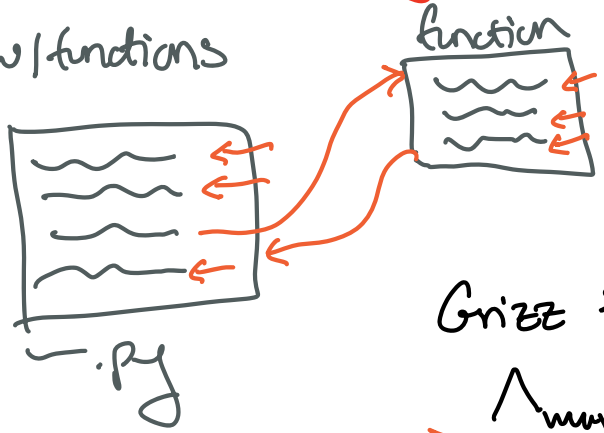
In our code...



w/ conditionals



w/ functions

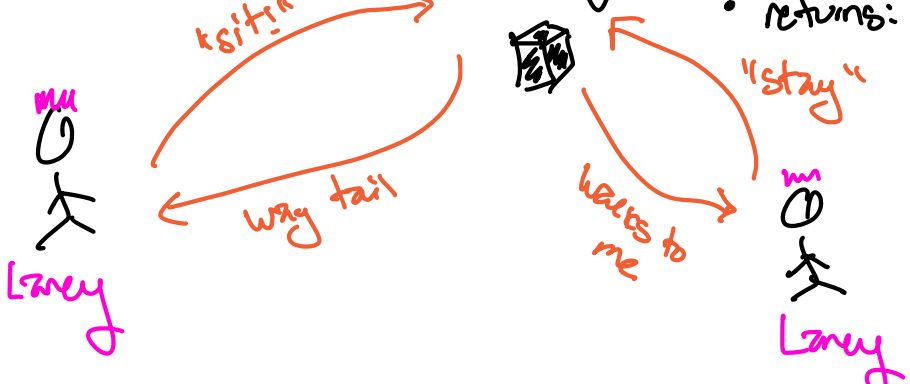


Calling a function:

- in main
- tells Python to execute other code
- function is black box

Grizz Function

- wants: command
- returns: reaction



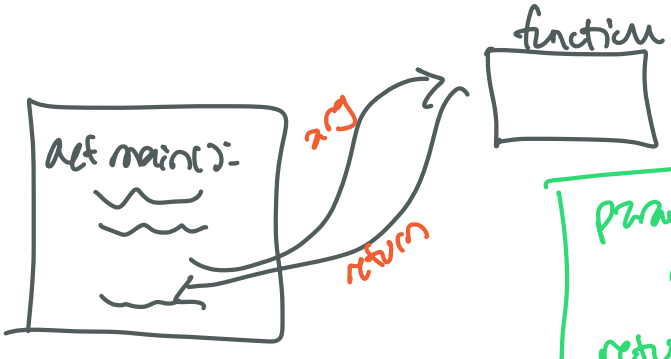
2. Calling Functions

Call → telling Python to execute the code

Function is a mini program
- already has been defined
- defined once
- we can call it over & over

- Function **argument** ~ value that it needs
  ex: "sit"
  we pass arguments to functions (when) we call the function
- Function **parameter** – data type it needs
  ex: command
  we specify params (when) we define the function
- **return** type – data type the function gives back
  ex: reaction
  info the function gives to function caller

return is 🤲
print is 🧍



params are how a func gets info
return is how a func gives info

ex
```
name = input("name?")
```

↳ calling a function

we can call any function:
- name
- params
- return type

funcname()
param: one string
returns: string

"name?"

What other functions have we called?

	open	random.randint
params	2 strings	2 ints
return	pointer to file	one int
example	`with open("file.txt", "r") as ...`	`var = random.randint(1,100)` (var: returns; 1, 100: args)

	max
params	some numbers
return	one int
example	`res = max(3,8,12)` (res: 12 (returns); 3, 8, 12: args)

3. Python

- Functions already exist!
- We need to call them
- Goal: call each function at least once

① generate_craps
- param: int
- returns: list of ints

② point_result
- params: int, 3 strings
- returns: one of the 3 strings

calling these functions?

```
lst = generate_craps(12)
```
↳ list of ints

```
result = point_result(7, "w", "l", "d")
```
↳ string