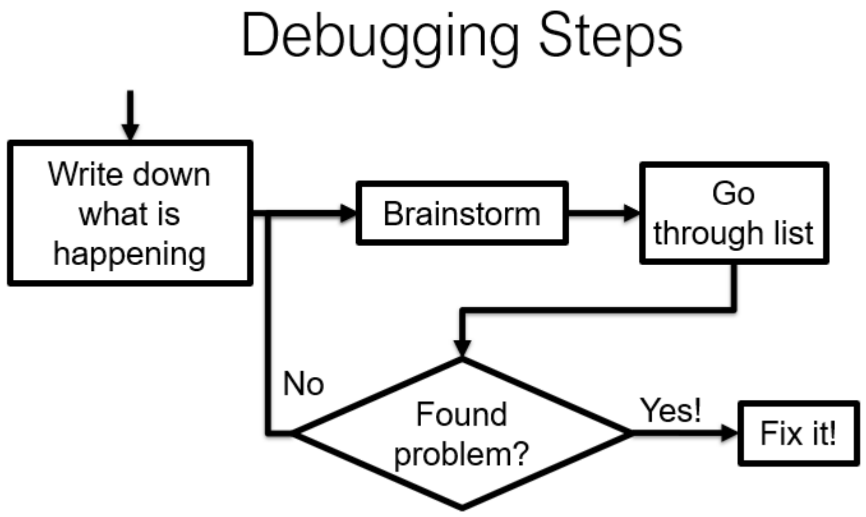
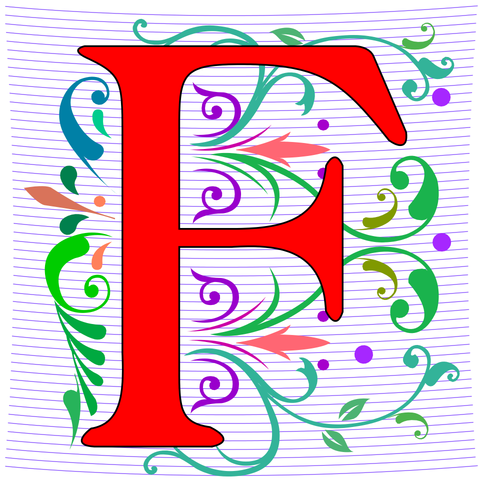
Compsci 101
Lists, Mutation, Objects

Susan Rodger

January 31, 2023



`F` is for …



- **Function**
  - Key to all programming
- **Floating Point**
  - Decimal numbers aka Python float
- **File**
  - Sequence of stored bits

Annie Easley

- American computer scientist, mathematician, and rocket scientist
- Worked at NACA and NASA
- BS in Math, Cleveland State
- Leader in developing the software for the Centaur rocket stage

On microaggressions: "If I can't work with you, I will work around you"

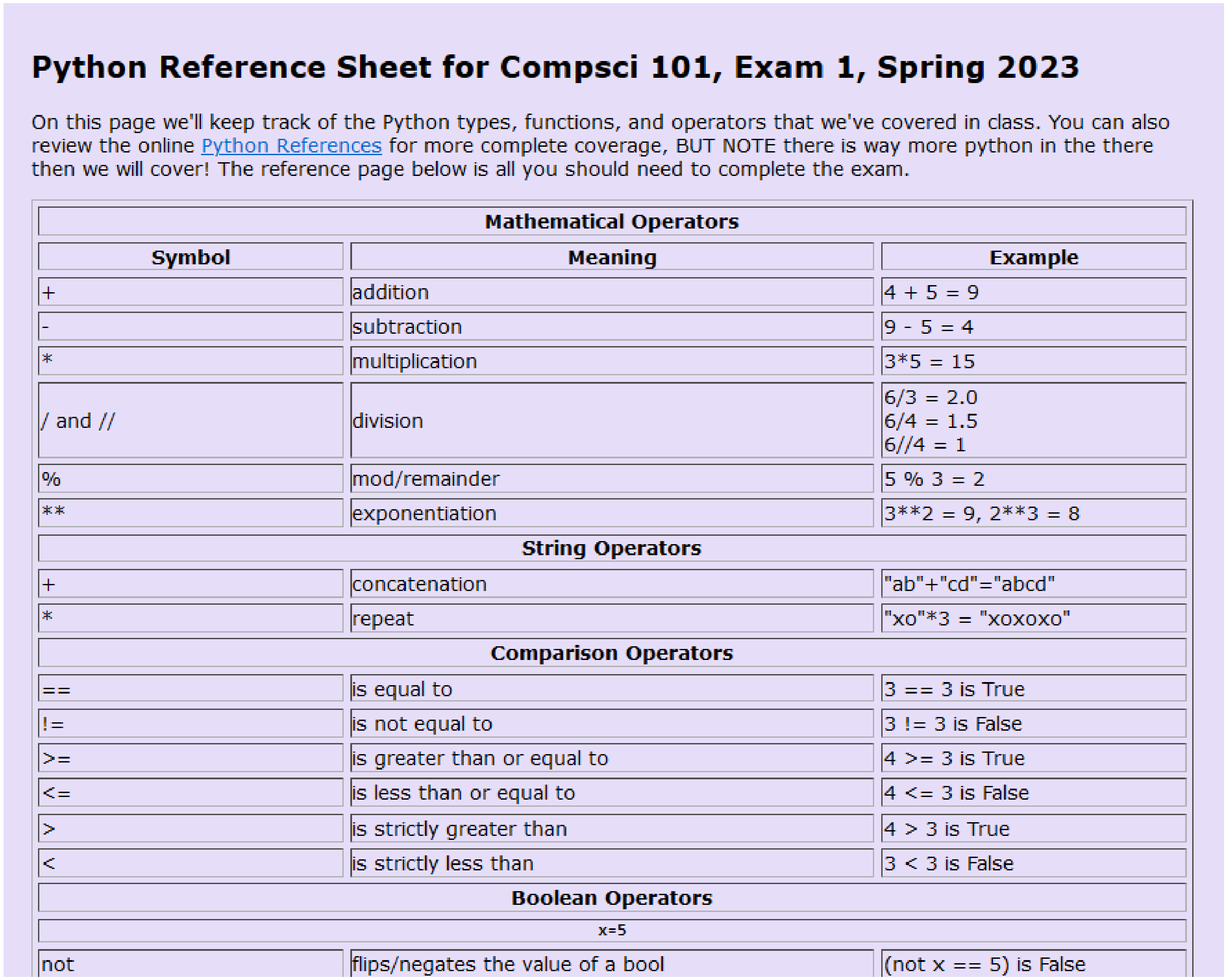
Announcements

- **Assign 1 Faces, Sakai QZ due TODAY (no grace day)**
  - Program is due Thursday (has one grace day)
- **Lab 3 Friday, Do Prelab 3 before lab**
- **Sakai QZ due by lecture time each day**

- **Exam 1 – Tuesday, February 7**
  - In person during class, covers topics through Feb 2
  - See old exams, python ref sheet on 2/7 date on calendar
  - Practice writing code on paper, more next time
- **Need SDAO letters for exams!**
  - Email them to Prof. Velasco

    yvelasco@cs.duke.edu

Python Reference Sheet, is attached to your exam (see link on calendar page, under 2/7)



Python Reference Sheet for Compsci 101, Exam 1, Spring 2023

On this page we'll keep track of the Python types, functions, and operators that we've covered in class. You can also review the online Python References for more complete coverage, BUT NOTE there is way more python in the there then we will cover! The reference page below is all you should need to complete the exam.

Mathematical Operators		
Symbol	**Meaning**	**Example**
+	addition	4 + 5 = 9
-	subtraction	9 - 5 = 4
*	multiplication	3*5 = 15
/ and //	division	6/3 = 2.0 6/4 = 1.5 6//4 = 1
%	mod/remainder	5 % 3 = 2
**	exponentiation	3**2 = 9, 2**3 = 8
String Operators		
+	concatenation	"ab"+"cd"="abcd"
*	repeat	"xo"*3 = "xoxoxo"
Comparison Operators		
==	is equal to	3 == 3 is True
!=	is not equal to	3 != 3 is False
>=	is greater than or equal to	4 >= 3 is True
<=	is less than or equal to	4 <= 3 is False
>	is strictly greater than	4 > 3 is True
<	is strictly less than	3 < 3 is False
Boolean Operators		
x=5		
not	flips/negates the value of a bool	(not x == 5) is False

PFTD

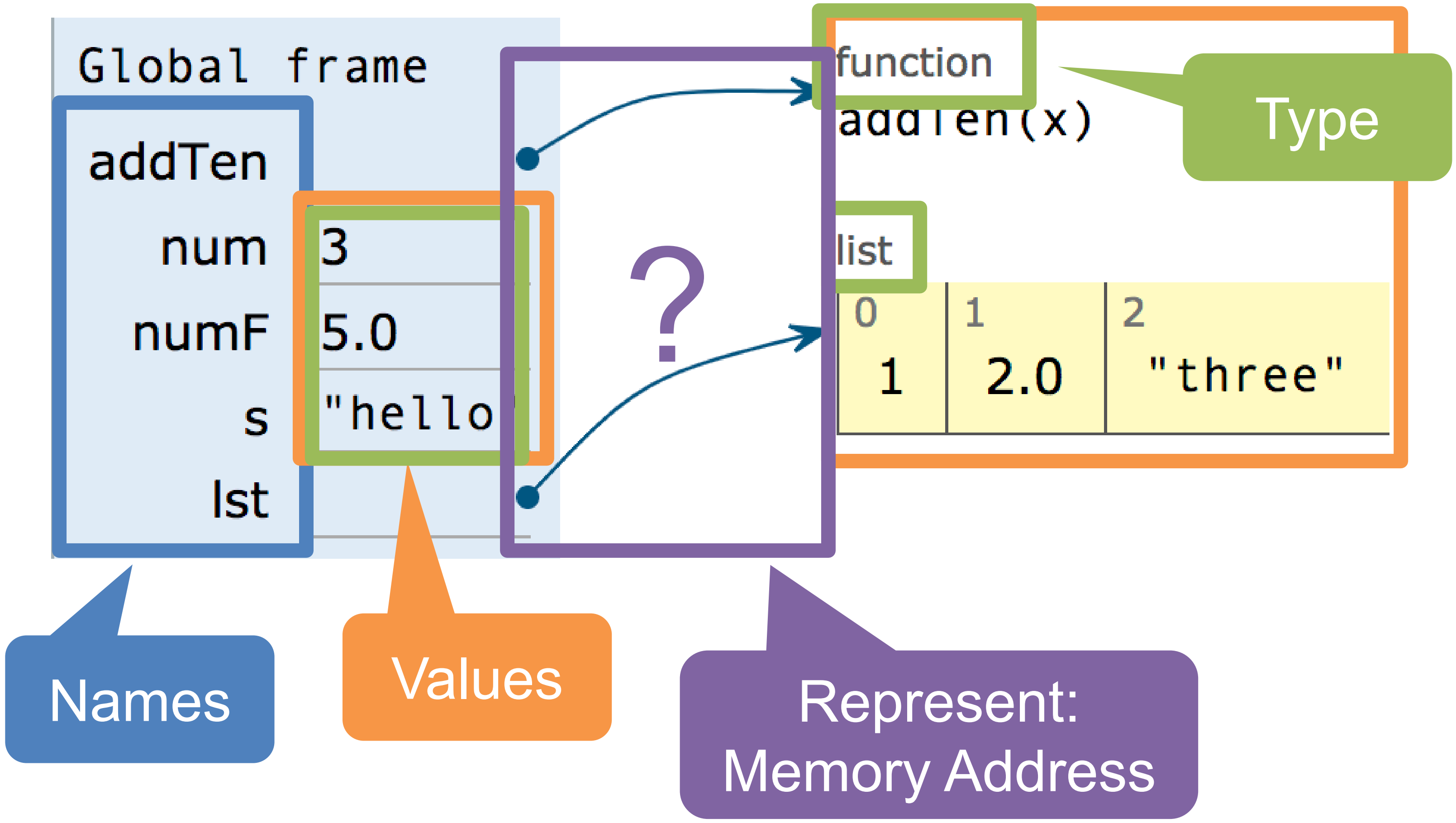
- **Functions as Parameters**
- **Debugging**
- **List concatenation and nesting**
- **Mutability**

Learning Goals: Faces

- **Understand differences and similarities:**
  - Function definitions vs function calls
  - Functions with return statements vs those without
  - Functions with parameters vs those without
  - Functions can be arguments

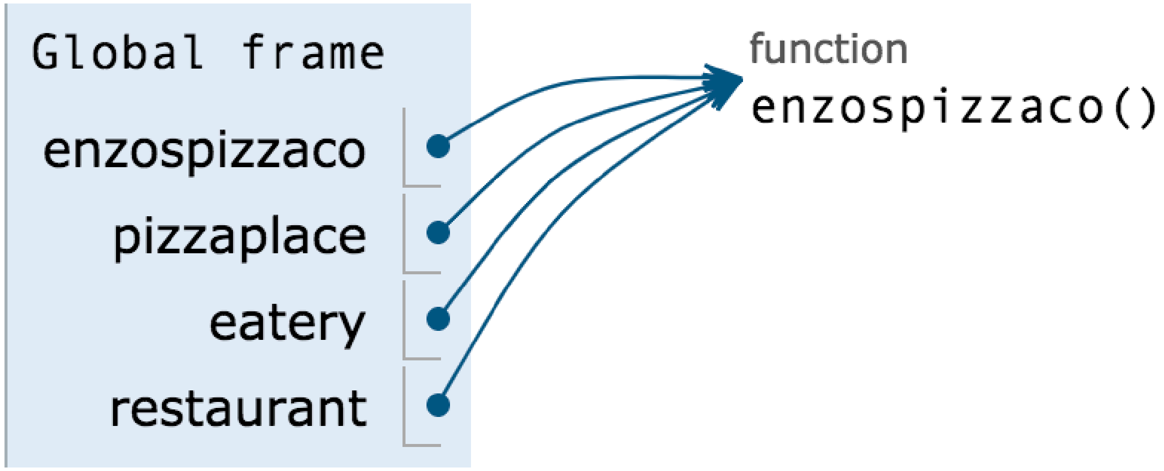
- **Be creative and learn lesson(s) about software design and engineering**
  - Create a small, working program, make incremental improvements.
  - Read the directions and understand specifications!

Name vs Value vs Type

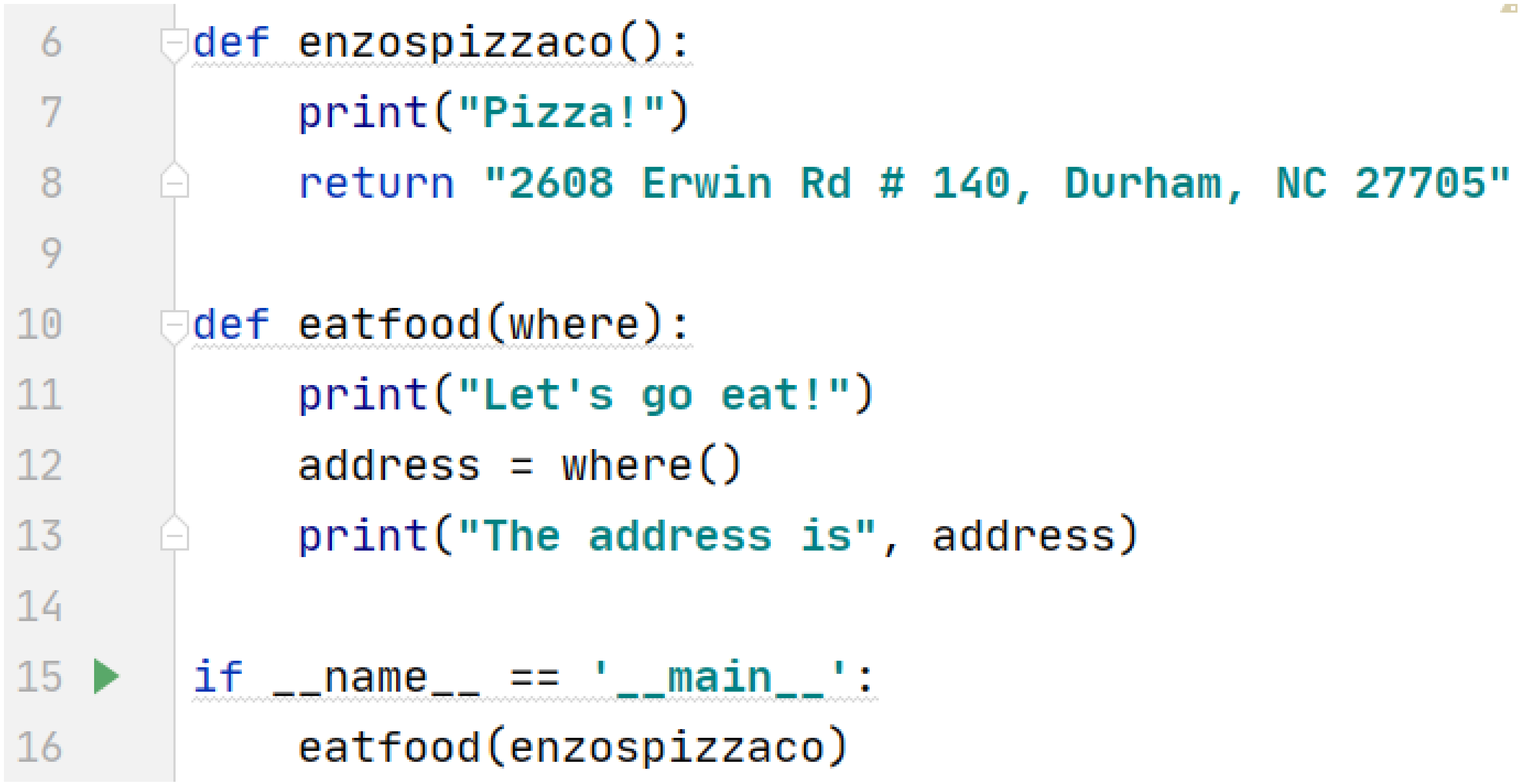


What are the arrows?

- **Name: Enzo's Pizza Co.**
- **Address (arrow): 2608 Erwin Rd # 140, Durham, NC 27705**
- **Value: Physical Store**

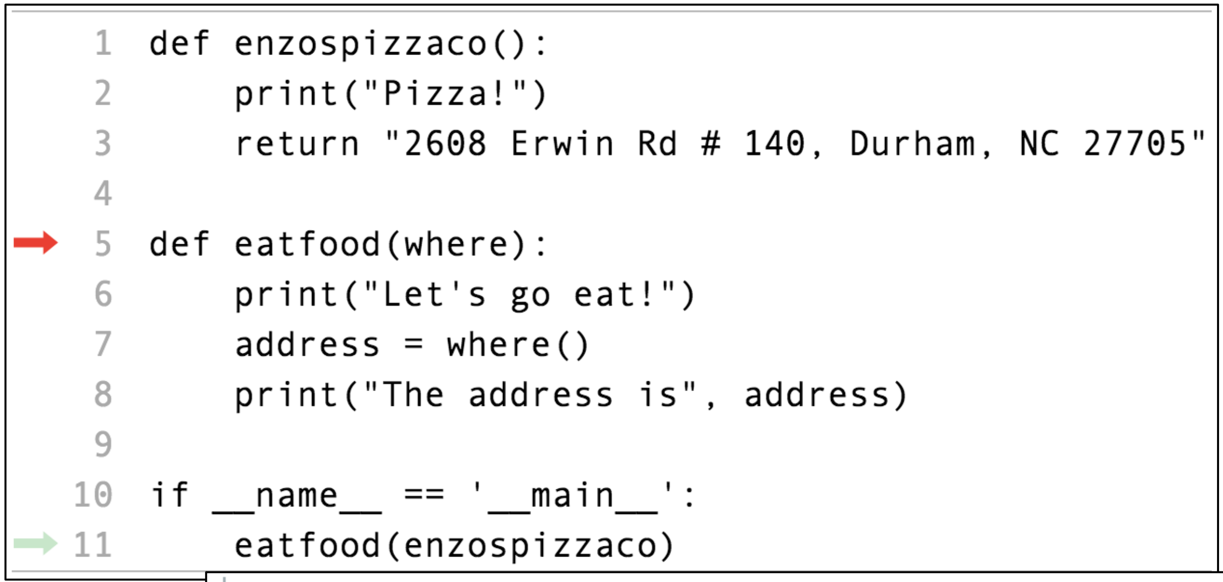


Pizza.py

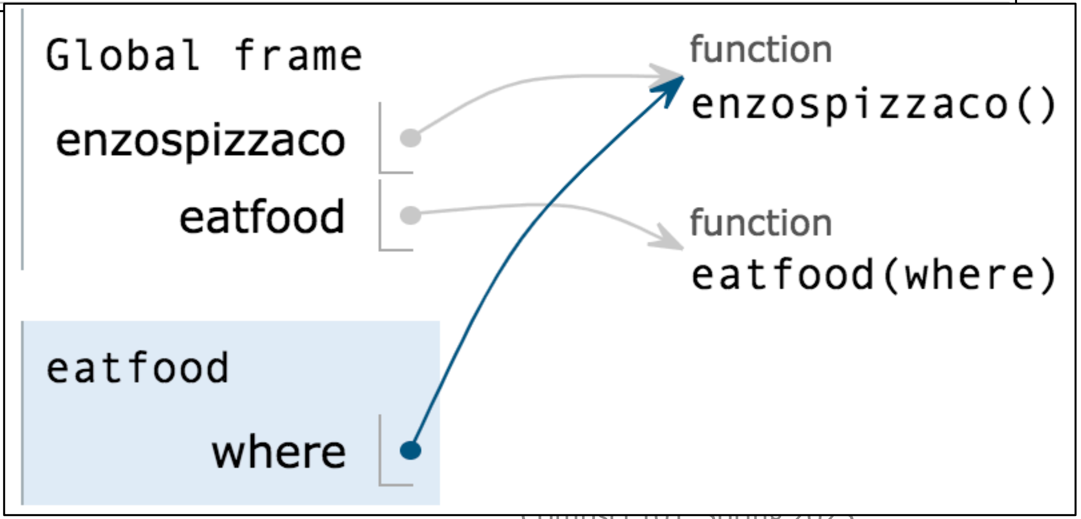


```python
def enzospizzaco():
    print("Pizza!")
    return "2608 Erwin Rd # 140, Durham, NC 27705"

def eatfood(where):
    print("Let's go eat!")
    address = where()
    print("The address is", address)

if __name__ == '__main__':
    eatfood(enzospizzaco)
```

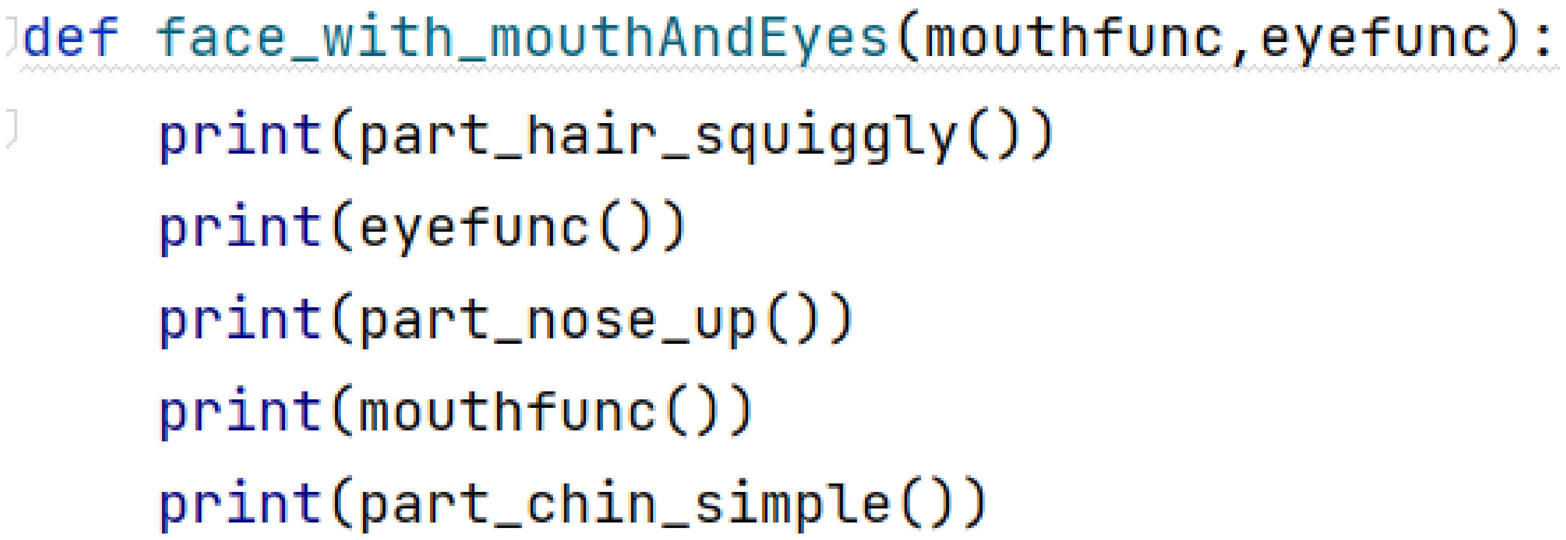
Functions can be arguments



```
def enzospizzaco():
    print("Pizza!")
    return "2608 Erwin Rd # 140, Durham, NC 27705"

def eatfood(where):
    print("Let's go eat!")
    address = where()
    print("The address is", address)

if __name__ == '__main__':
    eatfood(enzospizzaco)
```

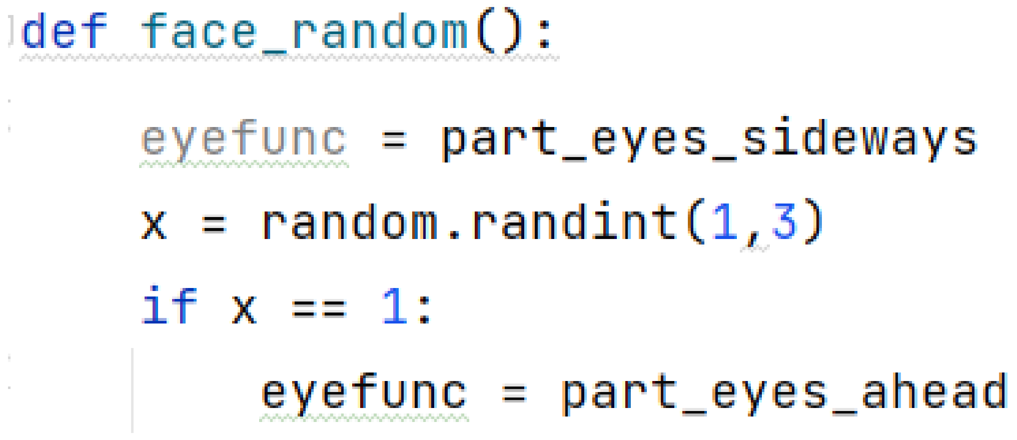


In Assignment 1 Faces



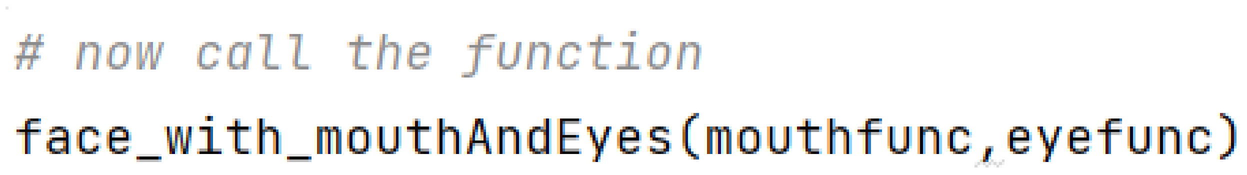
```python
def face_with_mouthAndEyes(mouthfunc,eyefunc):
    print(part_hair_squiggly())
    print(eyefunc())
    print(part_nose_up())
    print(mouthfunc())
    print(part_chin_simple())
```

In Assignment 1 Faces



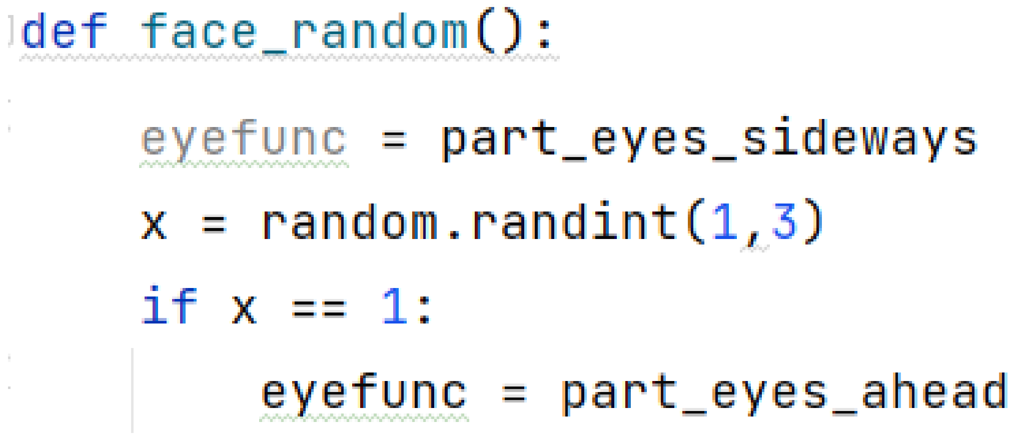
```python
def face_random():

    eyefunc = part_eyes_sideways
    x = random.randint(1,3)
    if x == 1:
        eyefunc = part_eyes_ahead
```

<Code Not Shown>



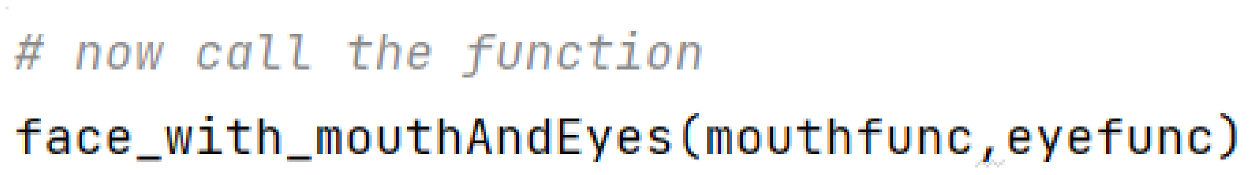
```python
    # now call the function
    face_with_mouthAndEyes(mouthfunc,eyefunc)
```

In Assignment 1 Faces



```
def face_random():

    eyefunc = part_eyes_sideways
    x = random.randint(1,3)
    if x == 1:
        eyefunc = part_eyes_ahead
```

<Code Not Shown>

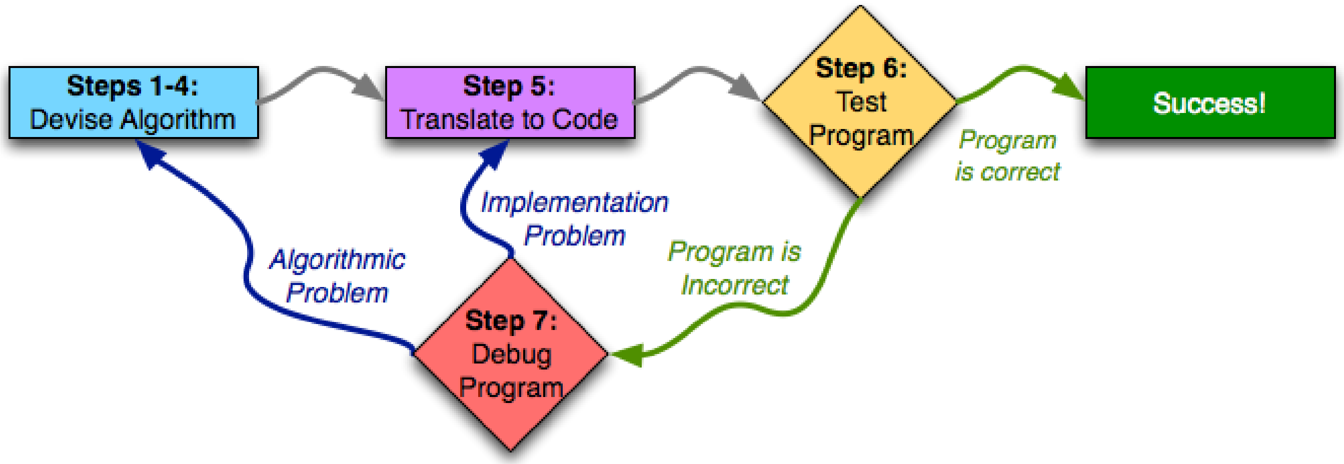


```
# now call the function
face_with_mouthAndEyes(mouthfunc,eyefunc)
```

WOTO-1: Functions as Parameters? http://bit.ly/101s23-0131-1

Debugging

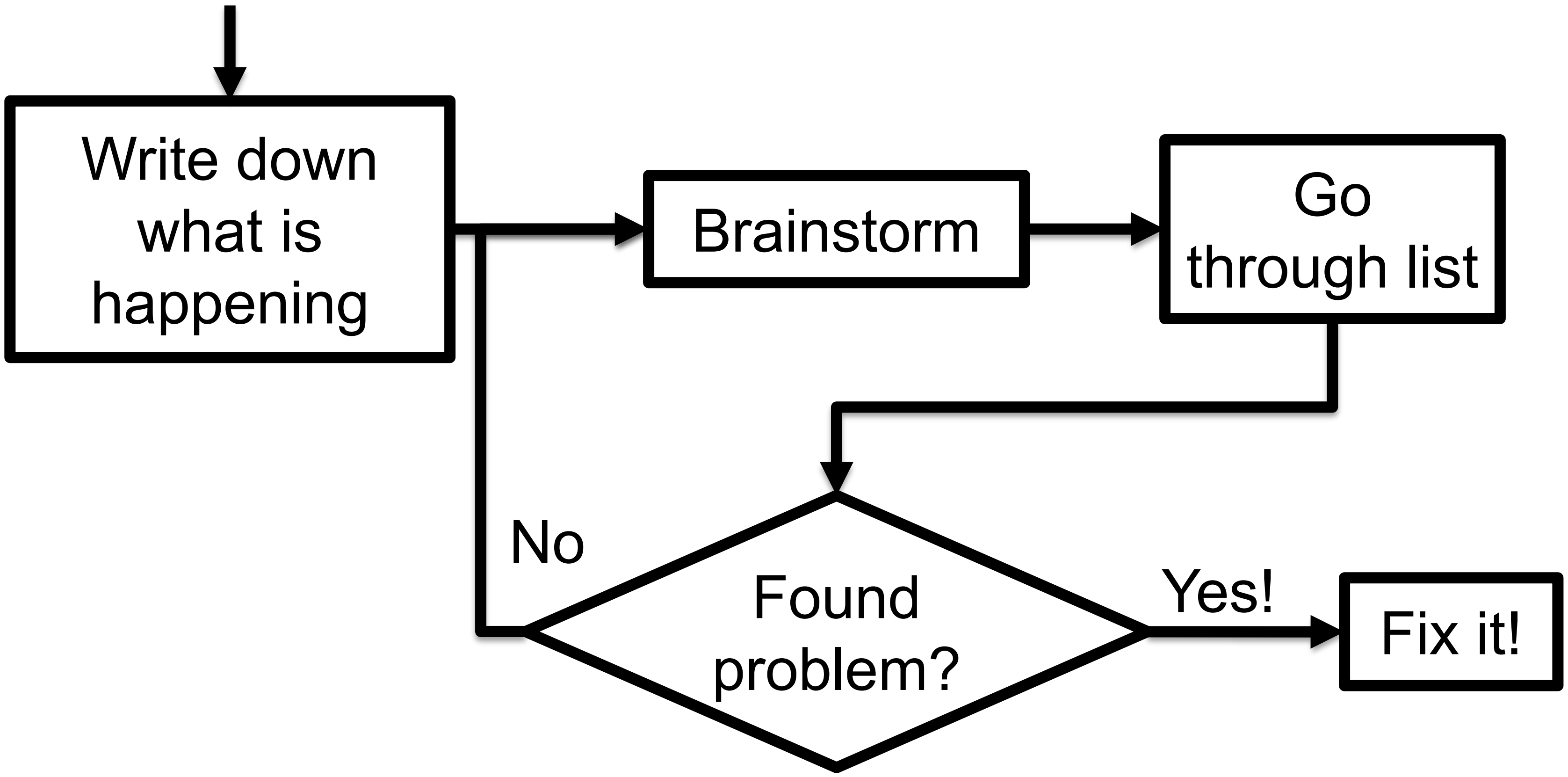
- **Finding what is wrong + fixing it**
  - Finding is its own skill set, and many find difficult
  - Fixing: revisit Step 1—5



Debugging Steps

1. **Write down exactly what is happening**
    1. input, output, what should be output
    2. ____ happened, but ____ should happen
2. **Brainstorm possible reasons this is happening**
    1. Write down list of ideas
3. **Go through list**
4. **Found it?**
    1. Yes, fix it using the 7-steps
    2. No, go back to step 2

Debugging Steps



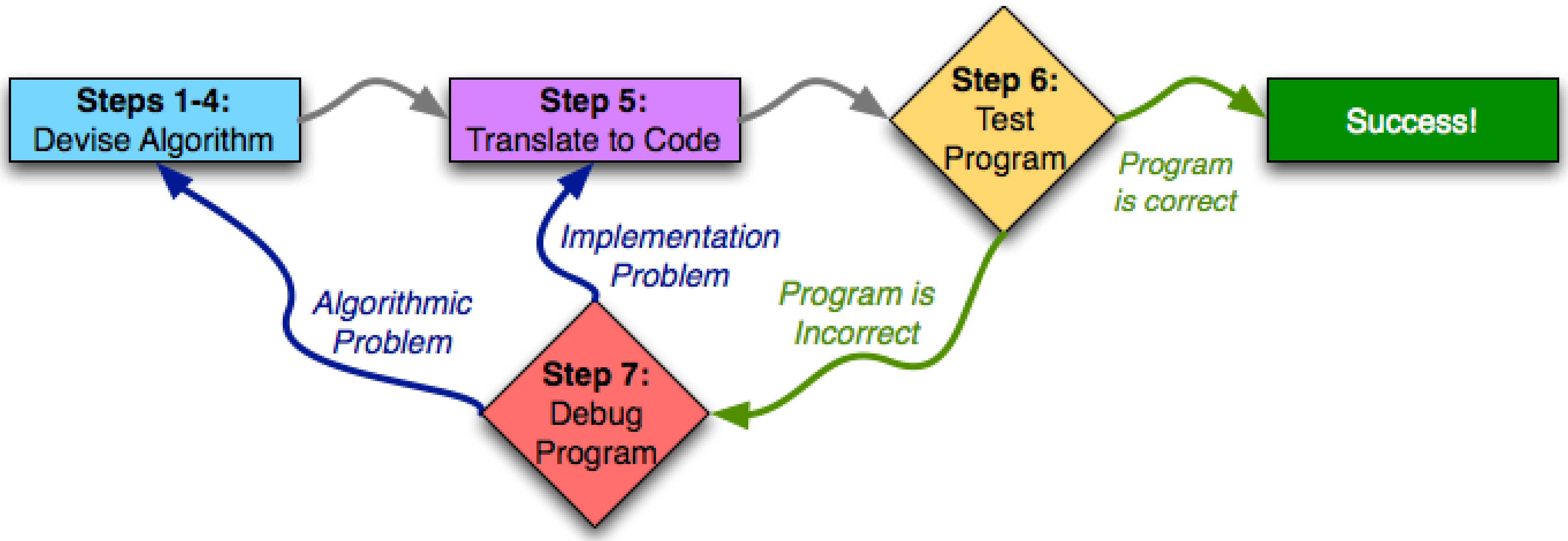
Relate W's to Debugging

- **Who was involved?**
  -
- **What happened?**
  -
- **Where did it take place?**
  -
- **When did it take place?**
  -
- **Why/How did it happen?**
  -

This Photo by Unknown Author is licensed under CC BY-NC-ND

Translate these questions to debugging

Step 7 -> Steps 1-4 or 5



Which year is a leap year?

- **A Leap Year must be divisible by four.**
- **But Leap Years don't happen every four years … there is an exception.**
  - If the **year** is also divisible by 100, it is not a **Leap Year** unless it is also divisible by 400.

WOTO-2: Buggy Leap Year http://bit.ly/101s23-0131-2

List Concatenation

- **String concatenation:**
  - `“hi” + “ there” == “hi there”`

- **List concatenation:**
  - `[1, 2] + [3, 4] == [1, 2, 3, 4]`

List examples

[1, 2] + [3, 4]

lst1 = [‘a’, ‘b’]

lst2 = [5, 6]

lst1 + lst2

lst1 + “c”

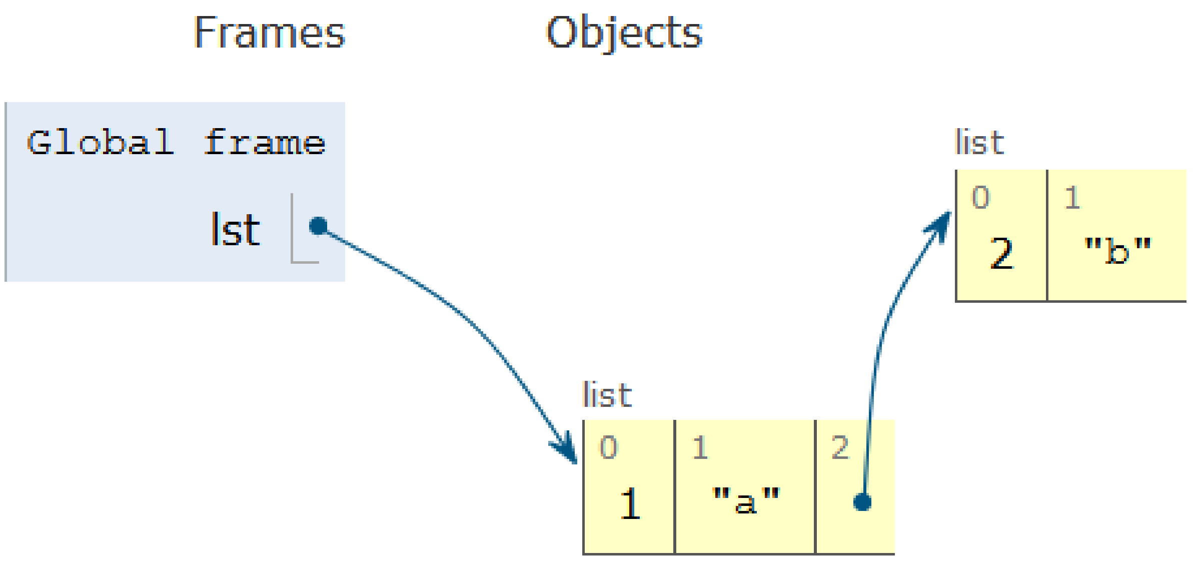
lst1 + [“c”]

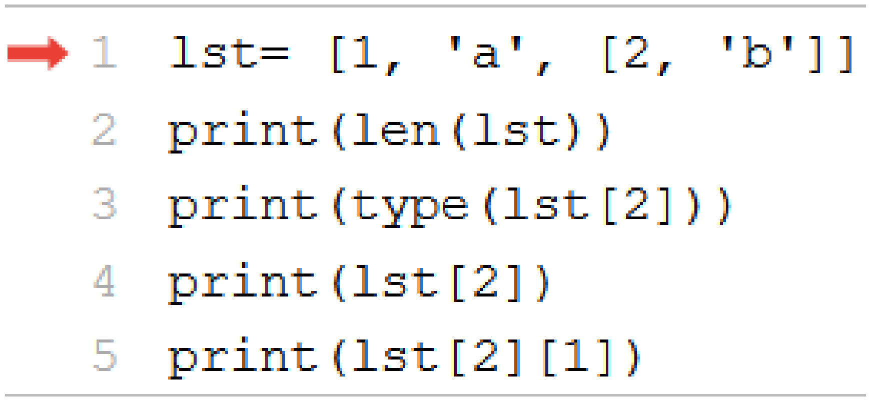
Nested Lists

- **Lists are heterogenous, therefore!**
  - `lst = [1, ‘a’, [2, ‘b’]]` is valid
  - `len(lst) ==`

- **How to index?**
  - [...] all the way down

Nested Lists with Python Tutor





```
lst= [1, 'a', [2, 'b']]
print(len(lst))
print(type(lst[2]))
print(lst[2])
print(lst[2][1])
```

Mutating Lists

- **`lt = [‘Hello’, ‘world’]`**
  - How to change **`lt`** to: `[‘Hello’, ‘Ashley’]`

- **Two ways: 1. Build new list or 2. modify list**
  1. Concatenation: `lt = [lt[0]] + [‘Ashley’]`
  2. Index: `lt[1] = ‘Ashley’`

- **How to change `‘b’` in `lt = [1, ‘a’, [2, ‘b’]]`?**
  - `lt[2][1] = ‘c’`

Mutating Lists code

```
lst1 = ['Hello', 'world']
print(lst1)
lst2 = [lst1[0]] + ['Ashley']
print(lst2)
print(lst1)
lst1[1] = 'Ashley'
print(lst1)

lst3 = [1, 'a', [2,'b']]
print(lst3)
lst3[2][1] = 'c'
print(lst3)
```

WOTO-3 List Mutation http://bit.ly/101s23-0131-3