F is for …

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

PFTD

- Totem
- Debugging
- List concatenation and nesting
- Mutability
- Objects and what that means

Assignment 1: Totem Poles
Learning Goals: Totem Pole

- 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 lessom(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

Compsci 101
Lists, Mutation, Objects
Part 2 of 4

Susan Rodger
February 9, 2021
Functions can be arguments

```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)
```

Output of Pizza2.py

```bash
C:\Users\Susan\AppData\Local\Programs\Python\Python3\python3 Pizza2.py
Let's go eat!
Pizza!
The address is 2608 Erwin Rd # 140, Durham, NC 27705
Let's go eat!
Indian cuisine!
The address is 2812 Erwin Road, Durham, NC 27705
Process finished with exit code 0
```
Functions Need Docstrings

• Docstring – a triple quote string right after the "def ...( ...):" describing the function
  • Recommend: 80 characters wide
  • PyCharm has a command for this!
    • Highlight string
    • Edit -> Fill Paragraph

```python
def f():
    ""
    I am a docstring, which stands for document string, that describes what this function is doing
    ""
```

List Concatenation

• String concatenation:
  • "hi" + " there" == "hi there"

• List concatenation:
  • [1, 2] + [3, 4] == [1, 2, 3, 4]

Nested Lists

• Lists are heterogeneous, therefore!
  • lst = [1, ‘a’, [2, ‘b’]] is valid
  • len(lst) == 3
    • [2, ‘b’] is one element in list lst

• How to index?
  • [...] all the way down
  • lst[2][1] returns ‘b’
Mutating Lists

- `lt = ['Hello', 'world']`
- Change to: `['Hello', 'Ashley']`
- Concatenation: `lt = [lt[0]] + ['Ashley']`
- Index: `lt[1] = 'Ashley'`
- How change ‘b’ in `lt = [1, 'a', [2, 'b']]`?
  - `lt[2][1] = 'c'`

Mutating Lists code

```python
1  lst1 = ['Hello', 'world']
2  print(lst1)
3  lst2 = [lst1[0]] + ['Ashley']
4  print(lst2)
5  print(lst1)
6  lst1[1] = 'Ashley'
7  print(lst1)
8
9  lst3 = [1, 'a', [2, 'b']]  # This line seems to be a mistake.
10 print(lst3)
11 print(lst3[2][1]) = 'c'
12```

Sir Tim Berners-Lee

- Turing award 2016
- World Wide Web
- HTTP vs. TCP/IP
- Just protocols?

I want you to realize that, if you can imagine a computer doing something, you can program a computer to do that.

Unbounded opportunity... limited only by your imagination. And a couple of laws of physics.
**Compsci 101**  
Lists, Mutation, Objects  
Part 4 of 4  

Susan Rodger  
February 9, 2021

---

**Immutable built-in Types**

- In python string, int, float, boolean - Immutable
  - Once created cannot change
  - These are still objects in Python3!!

- PythonTutor gets this wrong
  - Everything should be in Objects area

- Objects don't change
  - Value associated with variable changes

---

```python
val = 0  
bee = val
val = val + 20
```

---

**Aside: The Object Concept**

- Sometimes it helps to know how things “work”
  - Sometimes it’s wonderful to be oblivious
  - Abstraction!

- Object – a “thing” in memory/object space
  - Arrow / Computer memory address

- Python variables are references
  - Label that refers to object
  - Label is small, object is big
bat or ant?

Python 3.6

```python
1  a = ["pig", "cow", "dog", "bat"]
2  b = a
3  print(a)
4  a[-1] = "ant"
5  print(a)
6  print(b)
```

print(b) -> has ‘bat’ or ‘ant’?