Compsci 101
Functions, Randomness, Selection

Susan Rodger
January 18, 2022
Don’t sit in the last four rows

Come closer
D is for …

• **Debugging**
  • A key skill in making your programs run

• **Data (Science)**
  • Creating information from 0's and 1's

• **Dictionary**
  • Ultimate Python Data Structure
Prof. Nicki Washington
Duke University

• Research focuses on identity and cultural competence in computing
• Teaches: CompSci 240
• Book: *Unapologetically Dope: Lessons for Black Women and Girls on Surviving and Thriving in the Tech Field*
• On changing the environment, she says:

“The only way things will change is if those in the majority do the work. This also means that companies should place high expectations of cultural competence on prospective interns and new employees. This, in turn, places more expectations on college and university computing departments to focus on it as well. Only then will we start to see a real paradigm shift.”
Announcements

- Assignment 0 due tonight, 11:30pm
- Assignment 1 out today
- APT-1 due Thursday
- Drop/Add over Wednesday
  - You cannot change lab section without a perm no.
- QZ01-QZ04 submitted through Thursday 10:15am
- QZ05 is DUE at 10:15am on Thursday/will turn off!
- Trouble with Pycharm? Get help

- Remember: Ed Discussion back channel during lecture
Plan for the Day

• Assignment 1
• Print vs. Return
• Python Tutor
• Why use functions?
• Selection (if…elif…else)
• Random library
Assignment 1 and Pre-Lab 2

• Assignment 1 Faces due January 27

• Sakai Quiz on Assignment 1
  • Read through assignment 1
  • Take the quiz
  • Can take many times
  • Due January 25!

• Prelab 02 – before lab
  • Read Assignment 1 and take quiz once
Program execution

• Start at first line
• Ignore comments and blank lines
• Function – recognize, don’t execute
• Statements – executed one line at a time
  • After one statement, next statement
  • Calling a function transfers control to function
  • Function returns control back to where it was called by one of these:
    • Reach last line in the function, returns with None
    • Execute a return statement, return value
Print vs. Return

- Function ends one of two ways:
  - Reach end of function
  - Execute return statement
- Printing is not the same as returning
  - Print doesn’t leave the function

```python
# greeting.py

def greeting(name):
    print("Hello", name)
    print("nice to meet you")

def sum(num1, num2):
    answer = num1 + num2
    return answer

if __name__ == '__main__':
    greeting("Sarah")
    greeting("Bala")
    result = sum(6, 9)
    print(result)
    print(sum(4, 3))
```
Python Tutor Tool: Understanding Execution

- Using PythonTutor: [http://pythontutor.com](http://pythontutor.com)
  - Tool to trace through code
  - Copy and paste in your code
  - Think about these things as we trace code with Python Tutor
    - How are functions defined?
    - Where does execution begin?
    - What is the global frame?
    - What is a local/function frame?
Trace code with Python Tutor: Start

```python
def greeting(name):
    print("Hello", name)
    print("nice to meet you")

def sum(num1, num2):
    answer = num1 + num2
    return answer

if __name__ == '__main__':
    greeting("Sarah")
    greeting("Bala")
    result = sum(6, 9)
    print(result)
    print(sum(4, 3))
```

Click to step through code
Python Tutor Trace: Step 3

```python
Python 3.6
(known limitations)

def greeting(name):
    print("Hello", name)
    print("nice to meet you")

def sum(num1, num2):
    answer = num1 + num2
    return answer

if __name__ == '__main__':
    greeting("Sarah")
    greeting("Bala")
    result = sum(6, 9)
    print(result)
    print(sum(4, 3))
```

Saves information where functions are
Python Tutor Trace: Step 5

Python 3.6
(known limitations)

```python
1. def greeting(name):
2.     print("Hello", name)
3.     print("nice to meet you")
4.
5. def sum(num1, num2):
6.     answer = num1 + num2
7.     return answer
8.
9. if __name__ == '__main__':
10.    greeting("Sarah")
11.    greeting("Bala")
12.    result = sum(6, 9)
13.    print(result)
14.    print(sum(4, 3))
```

Call greeting and pass value “Sarah” to name
Python Tutor Trace: Step 8

Finish executing greeting function, no return value, so return None
Python Tutor Trace: Step 15

Python 3.6

Known limitations

1 def greeting(name):
2     print("Hello", name)
3     print("nice to meet you")

4

5 def sum(num1, num2):
6     answer = num1 + num2
7     return answer

8

9 if __name__ == '__main__':
10     greeting("Sarah")
11     greeting("Bala")

12     result = sum(6, 9)
13     print(result)
14     print(sum(4, 3))

Call function sum and pass values 6 and 9
Python Tutor Trace: Step 18

Python 3.6
(known limitations)

1 def greeting(name):
2     print("Hello", name)
3     print("nice to meet you")
4
5 def sum(num1, num2):
6     answer = num1 + num2
7     return answer
8
9 if __name__ == '__main__':
10    greeting("Sarah")
11    greeting("Bala")
12    result = sum(6,9)
13    print(result)
14    print(sum(4,3))

Finish executing sum function, return the value of answer, which is 15
Python Tutor Trace: Step 24

```python
Python 3.6
(known limitations)

1 def greeting(name):
2     print("Hello", name)
3     print("nice to meet you")
4
5 def sum(num1, num2):
6     answer = num1 + num2
7     return answer
8
9 if __name__ == '__main__':
10    greeting("Sarah")
11    greeting("Bala")
12    result = sum(6, 9)
13    print(result)
14    print(sum(4, 3))
```

here is the output

```
Hello Sarah
Hello Bala
nice to meet you
15
7
```

Frames

Global frame
- greeting
  - sum
  - result 15

Objects

- function greeting(name)
- function sum(num1, num2)

Done executing,
What PythonTutor Demonstrates

• What happens when program is first “executed”?
  • Execution starts at top of the file
    • Good practice: “Starting” code is in main program block
  • Functions created and referenced in global frame

• What happens when function called?
  • Arguments passed as parameters to function
    • Passed in same order inside parenthesis
    • See green and red arrows when executing
  • Control passes to function which executes
  • Return value replaces function call
WOTO-1 Simple Functions

• In your groups:
  • Come to a consensus
Why Use Functions?

• Re-use code/abstractions in multiple contexts
  • Sqrt, wordcount, URL-Webpage examples
• Test code/abstractions separately from their use
  • Develop independently, use with confidence
• Easier to change, re-use in different contexts
  • Relevant to Assignment 1: Faces
• Reduce risk of copy + paste mistakes
Old MacDonald Song!

```python
if __name__ == '__main__':
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!")
    print("And on his farm he had a pig, Ee-igh, Ee-igh, oh!")
    print("With a oink oink here")
    print("And a oink oink there")
    print("Here a oink there a oink everywhere a oink oink")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")

    print()
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!")
    print("And on his farm he had a horse, Ee-igh, Ee-igh, oh!")
    print("With a neigh neigh here")
    print("And a neigh neigh there")
    print("Here a neigh there a neigh everywhere a neigh neigh")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")
```
How to make code better?

```python
if __name__ == '__main__':
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!"")
    print("And on his farm he had a pig, Ee-igh, Ee-igh, oh!"")
    print("With a oink oink here")
    print("And a oink oink there")
    print("Here a oink there a oink everywhere a oink oink")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")

    print()
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!"")
    print("And on his farm he had a horse, Ee-igh, Ee-igh, oh!"")
    print("With a neigh neigh here")
    print("And a neigh neigh there")
    print("Here a neigh there a neigh everywhere a neigh neigh")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")
```
How to make code better?

```python
if __name__ == '__main__':
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!"")
    print("And on his farm he had a pig, Ee-igh, Ee-igh, oh!")
    print("With a oink oink here")
    print("And a oink oink there")
    print("Here a oink there a oink everywhere a oink oink")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")

    print()
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh!")
    print("And on his farm he had a horse, Ee-igh, Ee-igh, oh!")
    print("With a neigh neigh here")
    print("And a neigh neigh there")
    print("Here a neigh there a neigh everywhere a neigh neigh")
    print("Old MacDonald had a farm, Ee-igh, Ee-igh, oh")
```
def refrain():
    return "E-I-E-I-O\n"

def hadFarm():
    return "Old MacDonald had a farm, "

def verse(animal, sound):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + "," + refrain()
    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + ", " + sound + "\n"
    s += hadFarm() + refrain()
    return s

if __name__ == '__main__':
    print(verse("pig", "oink"))
    print(verse("horse", "neigh"))
def refrain():
    return "E-I-E-I-O\n"

def hadFarm():
    return "Old MacDonald had a farm, "

def verse(animal, sound):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + "," + refrain()
    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + ", " + sound + "\n"
    s += hadFarm() + refrain()
    return s

if __name__ == '__main__':
    print(verse("pig", "oink"))
    print(verse("horse", "neigh"))
WOTO-2 Old MacDonald

• Discuss what is new in the code
def refrain():
    return "E-I-E-I-O\n"

def hadFarm():
    return "Old MacDonald had a farm, "

def verse(animal, sound):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + ",," + refrain()
    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + ", " + sound + "\n"
    s += hadFarm() + refrain()
    return s

if __name__ == '__main__':
    print(verse("pig", "oink"))
    print(verse("horse", "neigh"))
def refrain():
    return "E-I-E-I-O\n"

def hadFarm():
    return "Old MacDonald had a farm, "

def verse(animal, sound):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + "," + refrain()
    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + " " + sound + "\n"
    s += hadFarm() + refrain()
    return s

if __name__ == '__main__':
    print(verse("pig", "oink"))
    print(verse("horse", "neigh"))
def refrain():
    return "E-I-E-I-O\n"

def hadFarm():
    return "Old MacDonald had a farm, "

def verse(animal, sound):
    s = hadFarm() + refrain()
    s += "And on his farm he had a " + animal + ",", + refrain()
    s += "With an " + sound + " " + sound + " here\n"
    s += "and an " + sound + " " + sound + " there\n"
    s += "Here an " + sound + ", there an " + sound + "\n"
    s += "Everywhere an " + sound + ", " + sound + "\n"
    s += hadFarm() + refrain()
    return s

if __name__ == '__main__':
    print(verse("pig", "oink"))
    print(verse("horse", "neigh"))
Try out code? Add a Verse?

- I will make the code from lecture available after class as a .zip file
- **Steps:**
  1. Create new project
     1. Project Interpreter is what created before
  2. Download zip file
  3. Unzip and copy files into new project
Functions Summarized

• Function call and Function definition related
  • Call must provide correct arguments
  • Names don’t matter, types are important
    • `print(verse("robot", 42))`?

• Functions help design, implement, organize
  • Without functions no APIs, no big programs
Making Decisions:

- Execute different code depending on something
  - Ask a question
  - Make decision based on answer

- If condition is true then do something
  - Condition: true or false
  - Something: any Python code
Selection Syntax

```python
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B
elif BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B
else:
    CODE_BLOCK_C
```

- What is similar and different?
  - What other variations could work?
  - Could only `elif...else` work?
Selection Syntax

```python
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
else:
    CODE_BLOCK_B
```

- What is similar and different?
- What other variations could work?
- Could only `elif...else` work?
- `if` – required
- `elif` – optional, as many as needed
- `else` – optional, no condition
Example: If

```python
def larger(num1, num2):
    if num1 > num2:
        return num1
    return num2

if __name__ == '__main__':
    print(larger(9, 17))
    print(larger(17, 9))
    print(larger(25, 6))
```
Example: If

```python
def larger(num1, num2):
    if num1 > num2:
        return num1
    return num2

if __name__ == '__main__':
    print(larger(9, 17))
    print(larger(17, 9))
    print(larger(25, 6))
```

Output:
17
17
25
Example 2: If-Elif-Else

```python
def pluralize(word):
    if word == 'fish':
        return word + 'es'
    elif word == 'brush':
        return word + 'es'
    else:
        return word + 's'

if __name__ == '__main__':
    print(pluralize('brush'))
    print(pluralize('card'))
    print(pluralize('fish'))
    print(pluralize('frog'))
    print(pluralize('fox'))
```

Output:
Example2: If-Elif-Else

```python
def pluralize(word):
    if word == "fish":
        return word + "es"
    elif word == "brush":
        return word + "es"
    else:
        return word + "s"

if __name__ == '__main__':
    print(pluralize("brush"))
    print(pluralize("card"))
    print(pluralize("fish"))
    print(pluralize("frog"))
    print(pluralize("fox"))
```

Output:
- brushes
- cards
- fishes
- frogs
- foxs
Random Module

- [https://docs.python.org/3/library/random.html](https://docs.python.org/3/library/random.html)

- `random.randint(a, b)`
  - Return a random integer $N$ such that $a \leq N \leq b$.  
  - Must import random at top of file to use the library
Example: Random

```python
import random

def larger(num1, num2):
    if num1 > num2:
        return num1
    return num2

if __name__ == '__main__':
    x = random.randint(1,20)
    y = random.randint(1,20)
    print(x, y, larger(x,y))
    x = random.randint(1,200)
    y = random.randint(1,200)
    print(x, y, larger(x,y))
```

Output:
Example: Random

```python
import random

def larger(num1, num2):
    if num1 > num2:
        return num1
    return num2

if __name__ == '__main__':
    x = random.randint(1, 20)
    y = random.randint(1, 20)
    print(x, y, larger(x, y))
    x = random.randint(1, 200)
    y = random.randint(1, 200)
    print(x, y, larger(x, y))
```

Output:
20 5 20
78 22 78

Run again…
Output:
17 6 17
5 123 123

Different values every time you run program
Does it say Meow? Does it Neigh?
import random

s += "What does a " + animal + " say?\n"
which = random.randint(0,1)

if which == 1:
    s += otherSound1 + "? No. "
    s += otherSound2 + "? No. "
else:
    s += otherSound2 + "? No. "
    s += otherSound1 + "? No. "

s += sound + "? Yes!\n"
import random

s += "What does a " + animal + " say?\n"
which = random.randint(0,1)

if which == 1:
    s += otherSound1 + "? No. "
    s += otherSound2 + "? No. "
else:
    s += otherSound2 + "? No. "
    s += otherSound1 + "? No. "

s += sound + "? Yes!\n"