

Compsci 101

Functions, Randomness, Selection

Susan Rodger
September 8, 2022



D is for ...

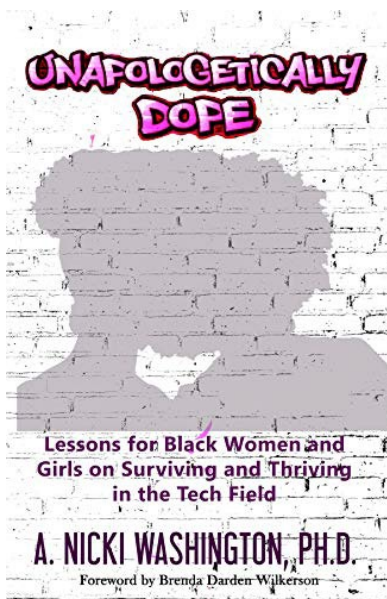


- **Debugging**
 - A key skill in making your programs run
- **Data (Science)**
 - Creating information from 0's and 1's
- **Dictionary**
 - Ultimate Python Data Structure



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 Tuesday**
- **APT-1 due Sept 15**
- **Drop/Add over Friday**
 - You cannot change lab section without a perm no.
- **QZ01-QZ05 submitted by Tuesday, Sept 13, 10:15am**
- **QZ05 is DUE at 10:15am on Sept 13 will turn off!**
- **Trouble with Pycharm? Get help**

- **Remember: Ed Discussion back channel during lecture**

Join Duke Mailing lists

compsci@duke.edu

- **Mailing list about**
 - Jobs, internships, research positions
 - Events related to computer science
- **How to join:**
 - Go to: lists.duke.edu
 - Be sure to authenticate
 - Add compsci@duke.edu
- **BE IN THE KNOW ABOUT COMPSCI!**

Plan for the Day

- **Review APT**
- **Print vs. Return**
- **Python Tutor**
- **Why use functions?**
- **Selection (if...elif...else)**
- **Random library**

Finish Slides From Last Time

- **Solving an APT**

Names and Return 0 Submission

- Take small steps to get all green!

Test Results Follow (scroll to see all)

of correct: 0 out of 19

1	fail
2	fail
3	fail
4	fail
5	fail
6	fail
7	fail
8	fail
9	fail
10	fail
11	fail
12	fail
13	fail
14	fail
15	fail
16	fail
17	fail
18	fail
19	fail

Test Results Follow (scroll to see all)

of correct: 12 out of 19

1	pass
2	pass
3	pass
4	pass
5	pass
6	pass
7	pass
8	pass
9	pass
10	pass
11	pass
12	pass
13	fail
14	fail
15	fail
16	fail
17	fail
18	fail
19	fail

Test Results Follow (scroll to see all)

of correct: 19 out of 19

1	pass
2	pass
3	pass
4	pass
5	pass
6	pass
7	pass
8	pass
9	pass
10	pass
11	pass
12	pass
13	pass
14	pass
15	pass
16	pass
17	pass
18	pass
19	pass

APT Testing and Submission

- **You wrote the code, how is it tested?**
 - Submit .py file with function to server
 - Server imports it
 - Server tests and checks by calling your function
- **The APT testing framework calls your code!**
 - Don't call us, we'll call you: *Hollywood principle*
- **Test/Submit + Check Grade**

APT Grading: CompSci 101,

This is the webpage for *grading and submitting* your APTs.

Check Grades

[check submissions](#)

Laundry dissected

```
def minutesNeeded(m):  
    return 60 + (m-1) * 25
```

- **Wrote formula using code to define a function**
- **How to use and re-use? By “calling” it**
 - Functions allow code to be re-used
 - `len()`, `float()`, `minutesNeeded()`

```
time = minutesNeeded(2)
```

Testing Laundry – two ways

1) Locally in Python Program Laundry

- Get it working before you use apt page

```
11 ▶ if __name__ == '__main__':  
12     num = 1  
13     print("m is", num, minutesNeeded(num))  
14     num = 2  
15     print("m is", num, minutesNeeded(num))  
16     num = 3  
17     print("m is", num, minutesNeeded(num))  
18     num = 10  
19     print("m is", num, minutesNeeded(num))
```

2) Run on the apt page

- Need internet connection, may take time

Where to put/use what in Python file

- **Top: docstring with date and username**
- **Function definitions right after docstring**
- **Test code inside `if __name__ == '__main__':`**

- **Variables inside vs outside a function**
 - ***Only*** use the variables inside that function
 - Therefore, ***do not*** use the variables outside the function (like in the main)
 - Your code will not work on the server

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

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



Python Tutor Tool: Understanding Execution

- **Using PythonTutor:** <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

Start on Line 1

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))
```

[Edit this code](#)

→ line that just executed

→ next line to execute

Click to step through code

<< First < Prev Next > Last >>

Step 1 of 24

Print output (drag lower right corner to resize)

Frames Objects

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

<http://bit.ly/101f22-0908-1>

- **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!

```
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?

```
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")
```

WOTO-2 Old MacDonald

<http://bit.ly/101f22-0908-2>

- **Discuss what is new in the code**



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

```
if BOOLEAN_CONDITION:  
    CODE_BLOCK_A
```

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

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

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

Example: If

Output:

```
6  def larger(num1, num2):
7      if num1 > num2:
8          return num1
9      return num2
10
11  if __name__ == '__main__':
12      print(larger(9, 17))
13      print(larger(17, 9))
14      print(larger(25, 6))
```

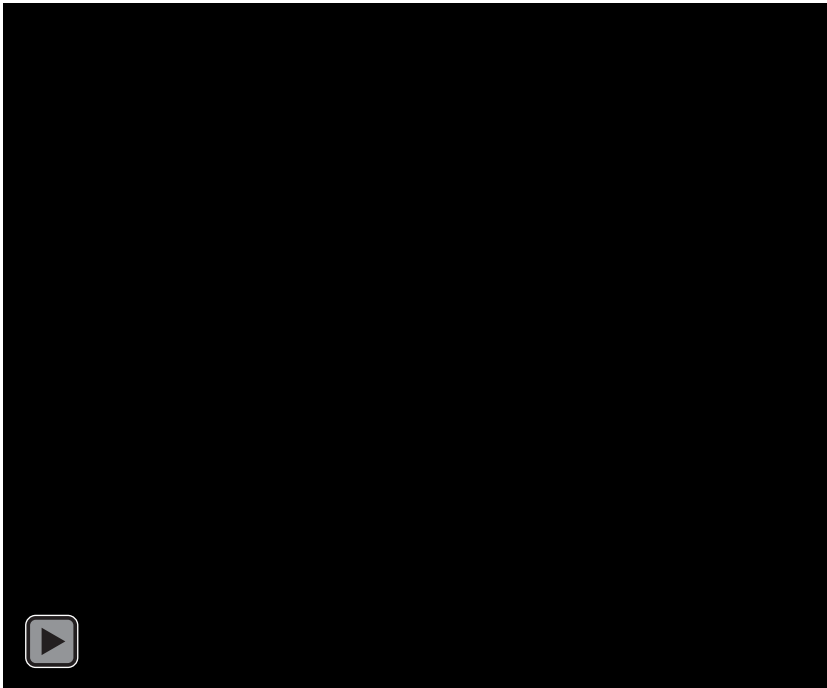
Example2: If-Elif-Else

```
6  def pluralize(word):
7      if word == "fish":
8          return word + "es"
9      elif word == "brush":
10         return word + "es"
11     else:
12         return word + "s"
13
14  if __name__ == '__main__':
15     print(pluralize("brush"))
16     print(pluralize("card"))
17     print(pluralize("fish"))
18     print(pluralize("frog"))
19     print(pluralize("fox"))
```

Output:

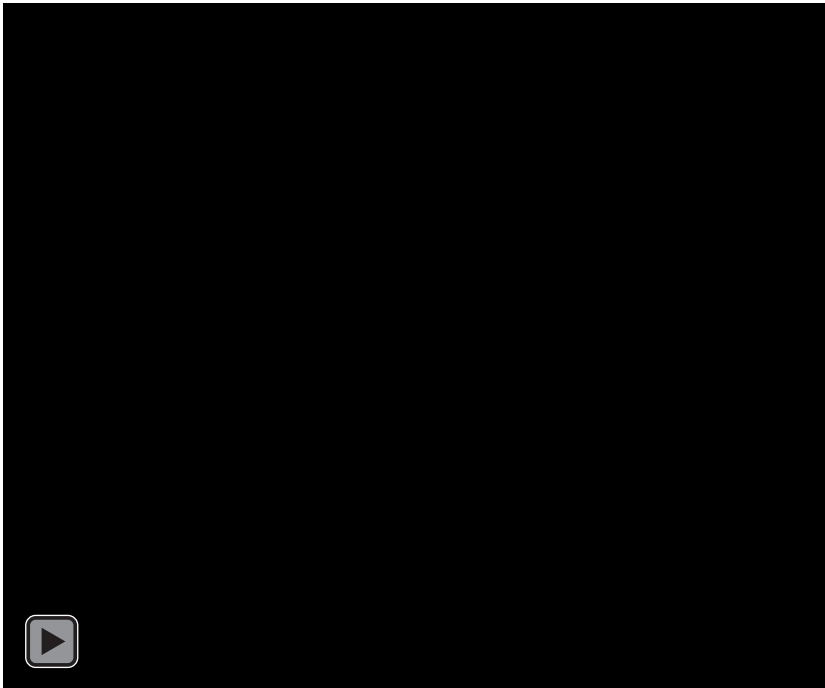
Cat Jumping Not Random

Cat always jumps to its right



Cat Jumping Random Direction

Cat jumps right or left, randomly



Random Module

- <https://docs.python.org/3/library/random.html>
- **Must import random at top of file to use the library**
 - import random
- **Now can use any of random's functions**
- **To call a function from a module**
 - `<MODULE_NAME>.<FUNCTION_NAME>(args)`
- **Example:**
 - `random.randint(a, b)`
 - Return a random integer N such that $a \leq N \leq b$.

Example: Random

Output:

```
6 import random
7
8 def larger(num1, num2):
9     if num1 > num2:
10         return num1
11     return num2
12
13 if __name__ == '__main__':
14     x = random.randint(1,20)
15     y = random.randint(1,20)
16     print(x, y, larger(x,y))
17     x = random.randint(1,200)
18     y = random.randint(1,200)
19     print(x, y, larger(x,y))
```


WOTO-3

<http://bit.ly/101f22-0908-3>

