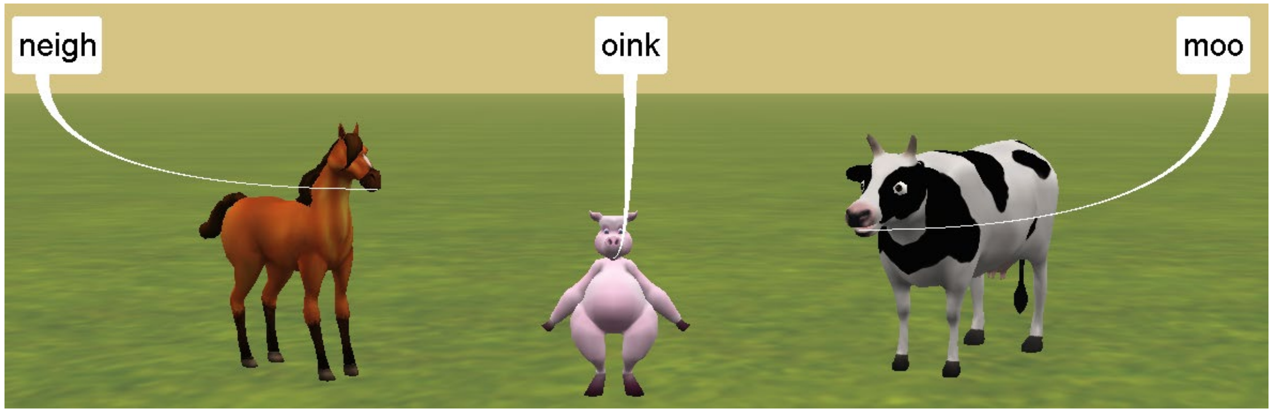
Compsci 101
Functions, Randomness, Selection

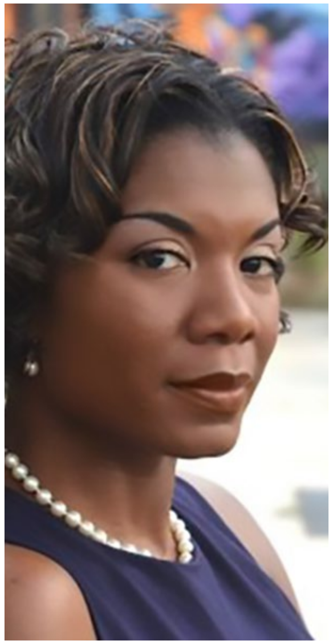
Susan Rodger

January 24, 2023



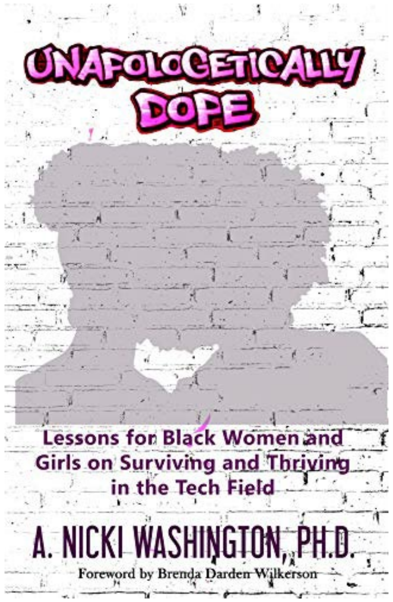
D is for …

- **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:


UNAPOLOGETICALLY DOPE
Lessons for Black Women and Girls on Surviving and Thriving in the Tech Field
A. NICKI WASHINGTON, PH.D.
Foreword by Brenda Darden Wilkerson

"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 can still turn in due to Drop/Add**
- **Assignment 1 out later today**
- **Prelab 2 out today!**
- **APT-1 due Jan. 26**
- **Drop/Add over Tomorrow! 1/25**
  - You cannot change lab section without a perm no.
- **QZ01-QZ05 submitted by Thursday, Jan 26, 10:15am**
- **QZ05 is DUE at 10:15am on Jan 26 will turn off!**
- **Trouble with Pycharm? Get help**

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

WOTO grading

- **WOTO's are the forms we do in lecture**

- **We expect you to come to class and do them.**
- **We understand occasionally you may miss class. The WOTOs must be completed by the next night!**
- **Tuesday WOTO by Wed night, Thur WOTO by Fri night!**

- **You should be submitting them late only a few times**
- **Lecture Video is put up later the day of lecture on today's date on our calendar webpage**
- **Video is NOT always guaranteed to work – many mess-ups!**

Join SAGE

- **STEM Advancement through Group Engagement**
- **Small groups of students working on additional problems related to CompSci 101**
- **Limited spots**
- **Sign up now on Academic Resource Center website**
- **See Ed Discussion Post (pinned at the top)**

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

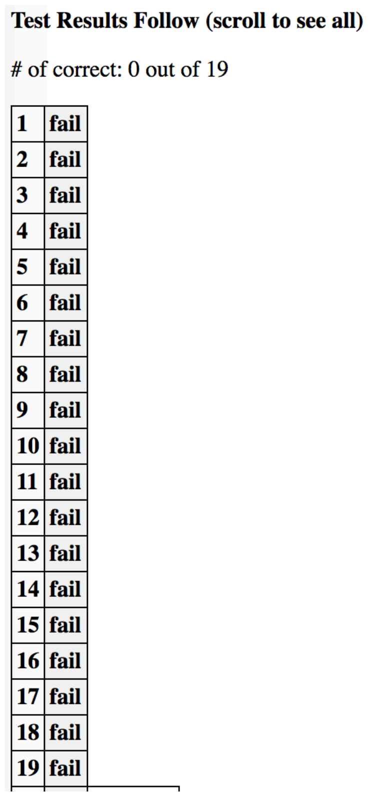
- **Review APT**
- **Print vs. Return**
- **Python Tutor**
- **Why use functions?**
- **Selection (if…elif…else)**
- **Random library**

Finish Slides From Last Time

- **Solving an APT**

Names and Return 0 Submission

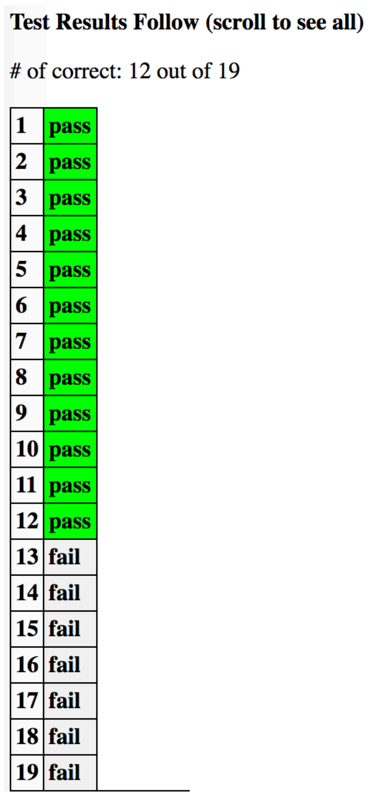
- **Take small steps to get all green!**



Test Results Follow (scroll to see all)

# of correct: 0 out of 19

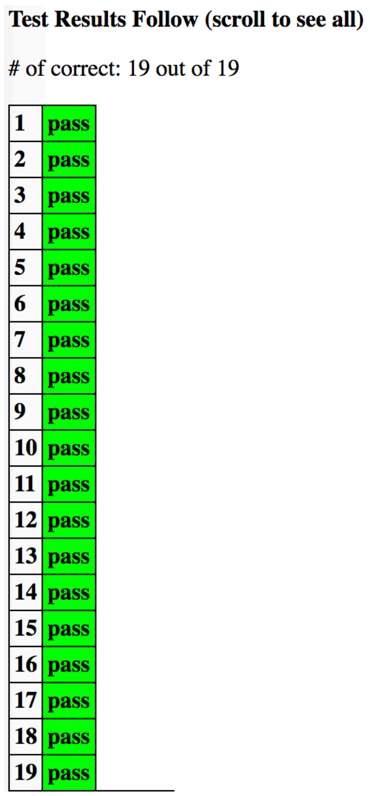
1	fail
2	fail
3	fail
4	fail
5	fail
6	fail
7	fail
8	fail
9	fail
10	fail
11	fail
12	fail
13	fail
14	fail
15	fail
16	fail
17	fail
18	fail
19	fail



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



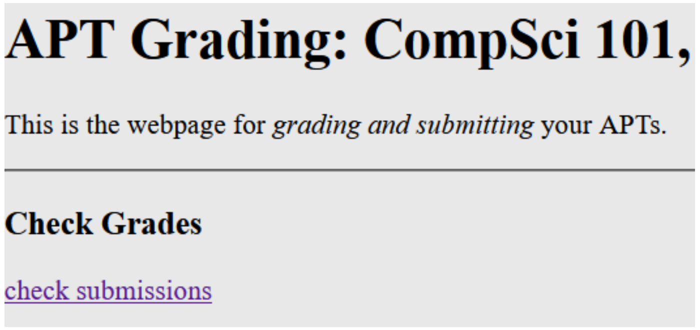
Test Results Follow (scroll to see all)

# 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)
```

Laundry dissected

Defining

Parameter

```
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)
print(time)
```

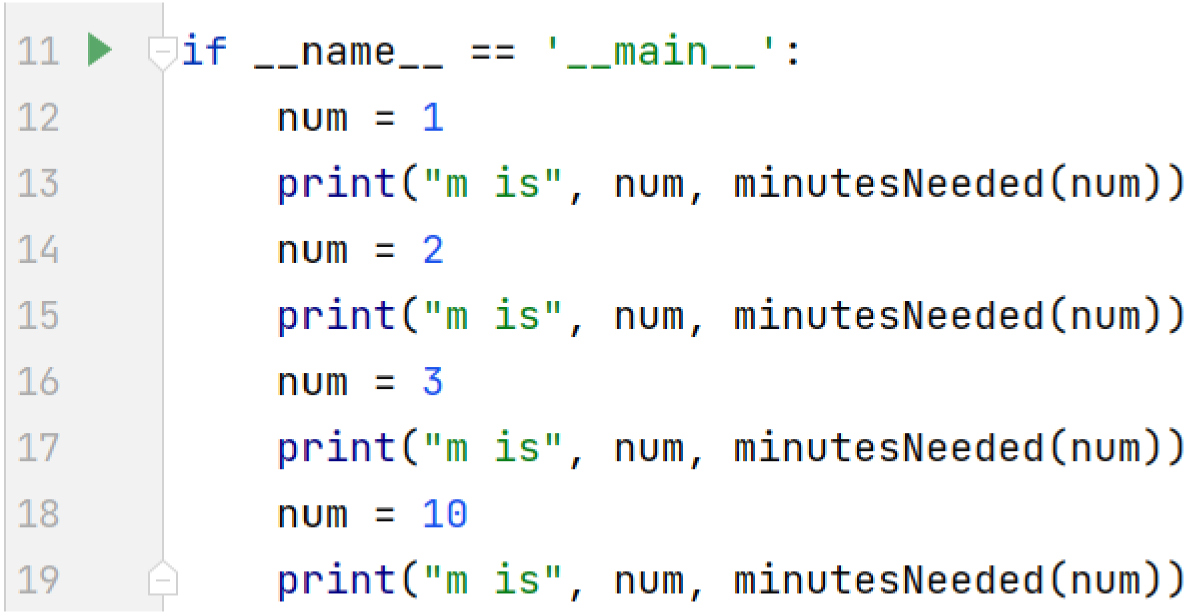
Argument

Calling

Output is 85

Testing Laundry – two ways

1) **Locally in Python Program Laundry**
   - Get it working before you use apt page



```
if __name__ == '__main__':
    num = 1
    print("m is", num, minutesNeeded(num))
    num = 2
    print("m is", num, minutesNeeded(num))
    num = 3
    print("m is", num, minutesNeeded(num))
    num = 10
    print("m is", num, minutesNeeded(num))
```

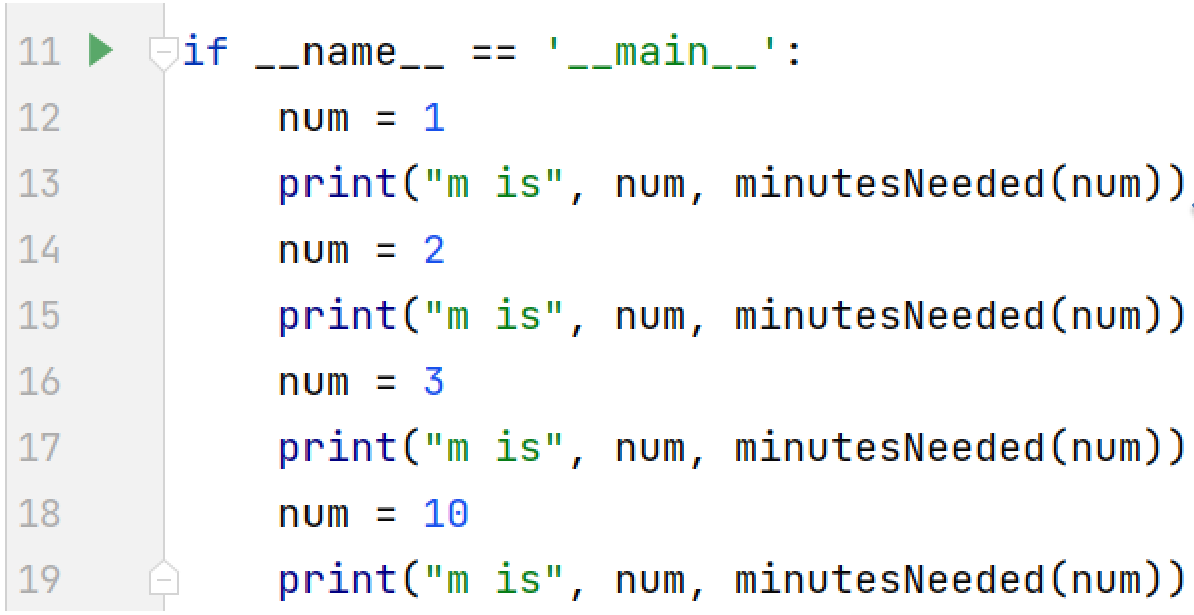
2) **Run on the apt page**
   - Need internet connection, may take time

Testing Laundry – two ways

1) **Locally in Python Program Laundry**

Testing it in Pycharm

- Get it working before you use apt page



```
if __name__ == '__main__':
    num = 1
    print("m is", num, minutesNeeded(num))
    num = 2
    print("m is", num, minutesNeeded(num))
    num = 3
    print("m is", num, minutesNeeded(num))
    num = 10
    print("m is", num, minutesNeeded(num))
```

Identify your output, make the print statements meaningful

2) **Run on the apt page**

Submitting it for a grade

- 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__’:`**

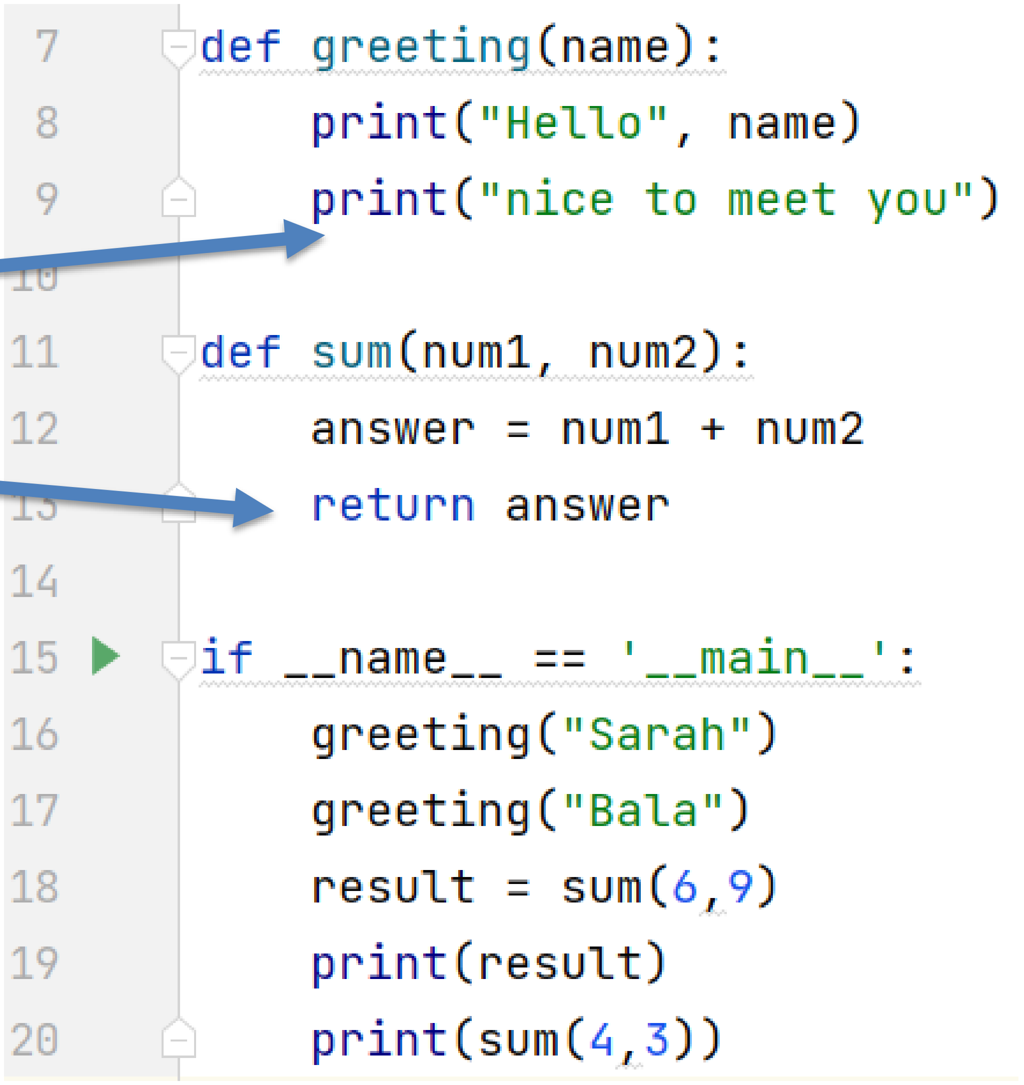
- **Variables inside vs outside a function**
  - ***Only*** use the variables inside that function
  - Therefore, ***do not*** use the variables outside the function (like in the main)
    - Your code will not work on the server

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



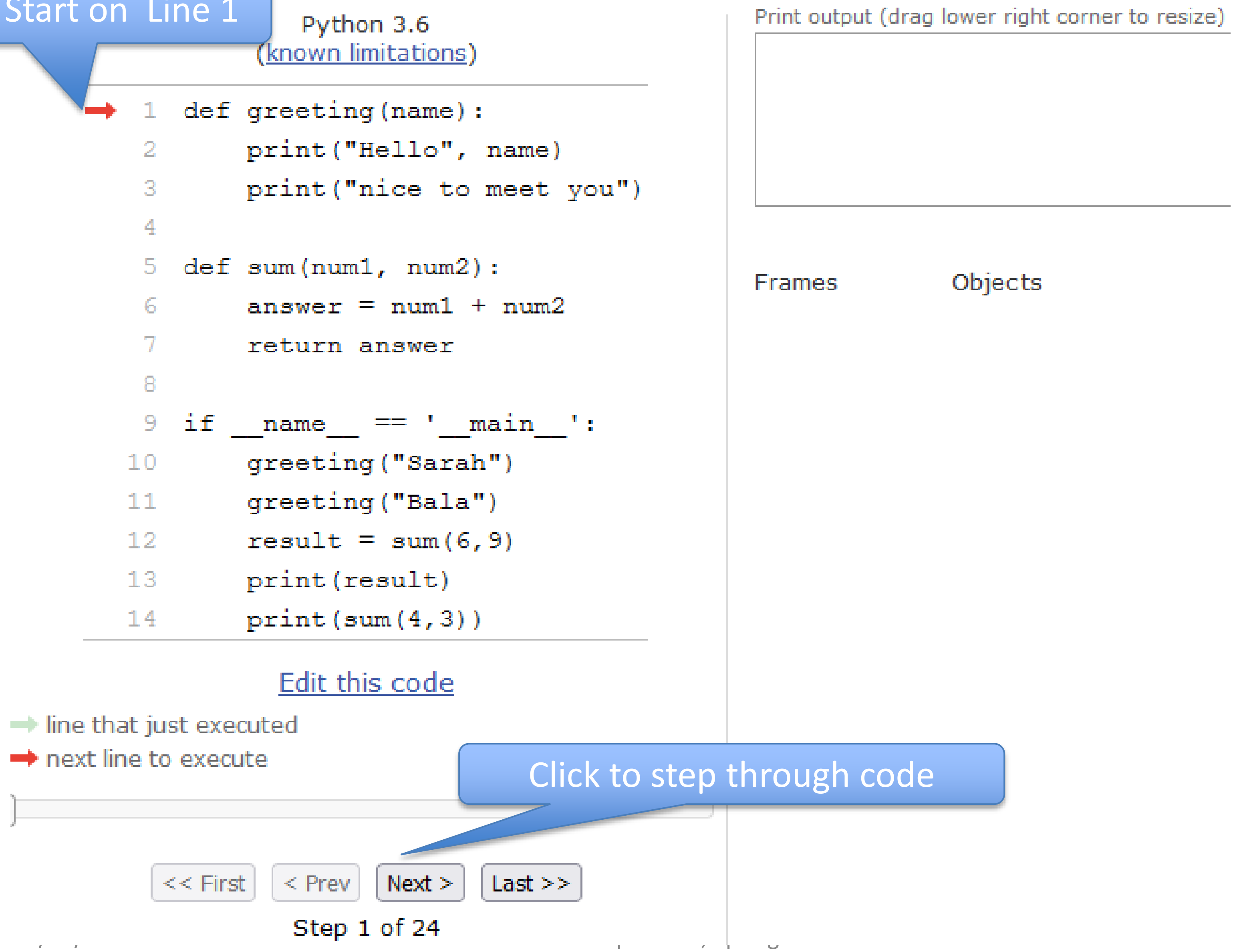
```
def greeting(name):
    print("Hello", name)
    print("nice to meet you")

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**
  - 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

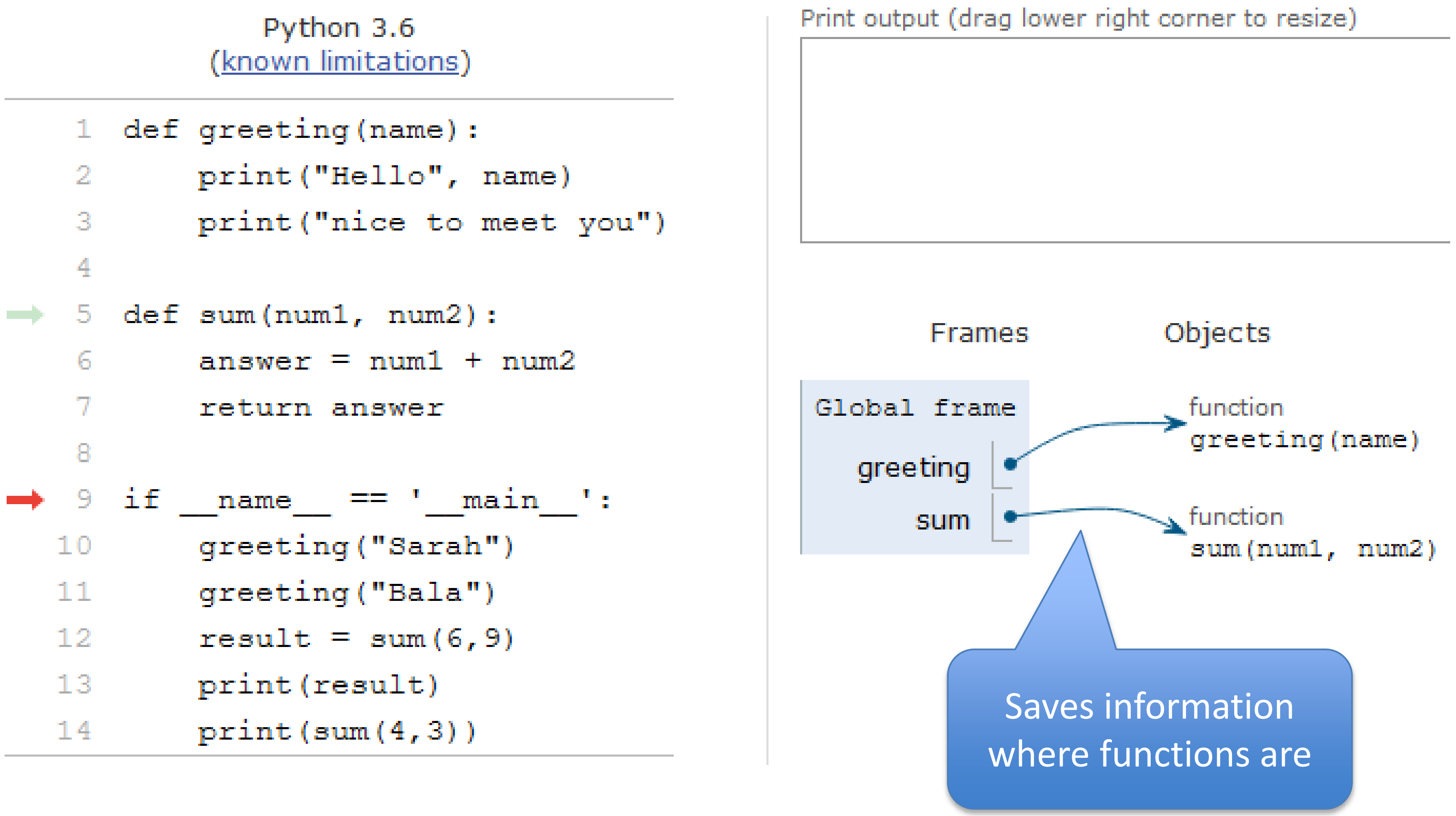


```
 1  def greeting(name):
 2      print("Hello", name)
 3      print("nice to meet you")
 4
 5  def sum(num1, num2):
 6      answer = num1 + num2
 7      return answer
 8
 9  if __name__ == '__main__':
10      greeting("Sarah")
11      greeting("Bala")
12      result = sum(6,9)
13      print(result)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

➡ line that just executed
🡆 next line to execute

Click to step through code

Python Tutor Trace: Step 3

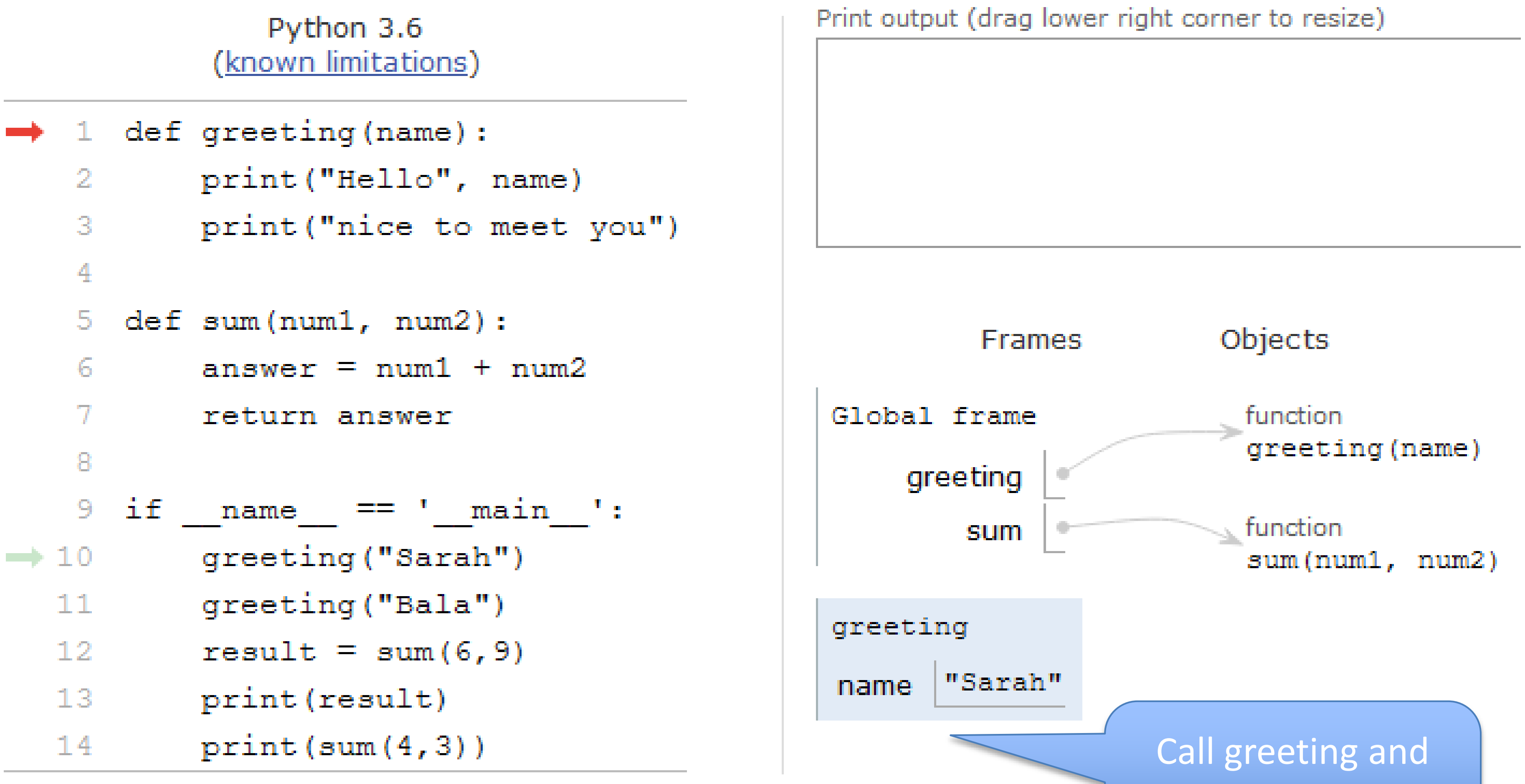


Python 3.6
(known limitations)

```
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))
```

Print output (drag lower right corner to resize)
Frames
Objects
Global frame
greeting
sum
function
greeting(name)
function
sum(num1, num2)
Saves information where functions are

Python Tutor Trace: Step 5

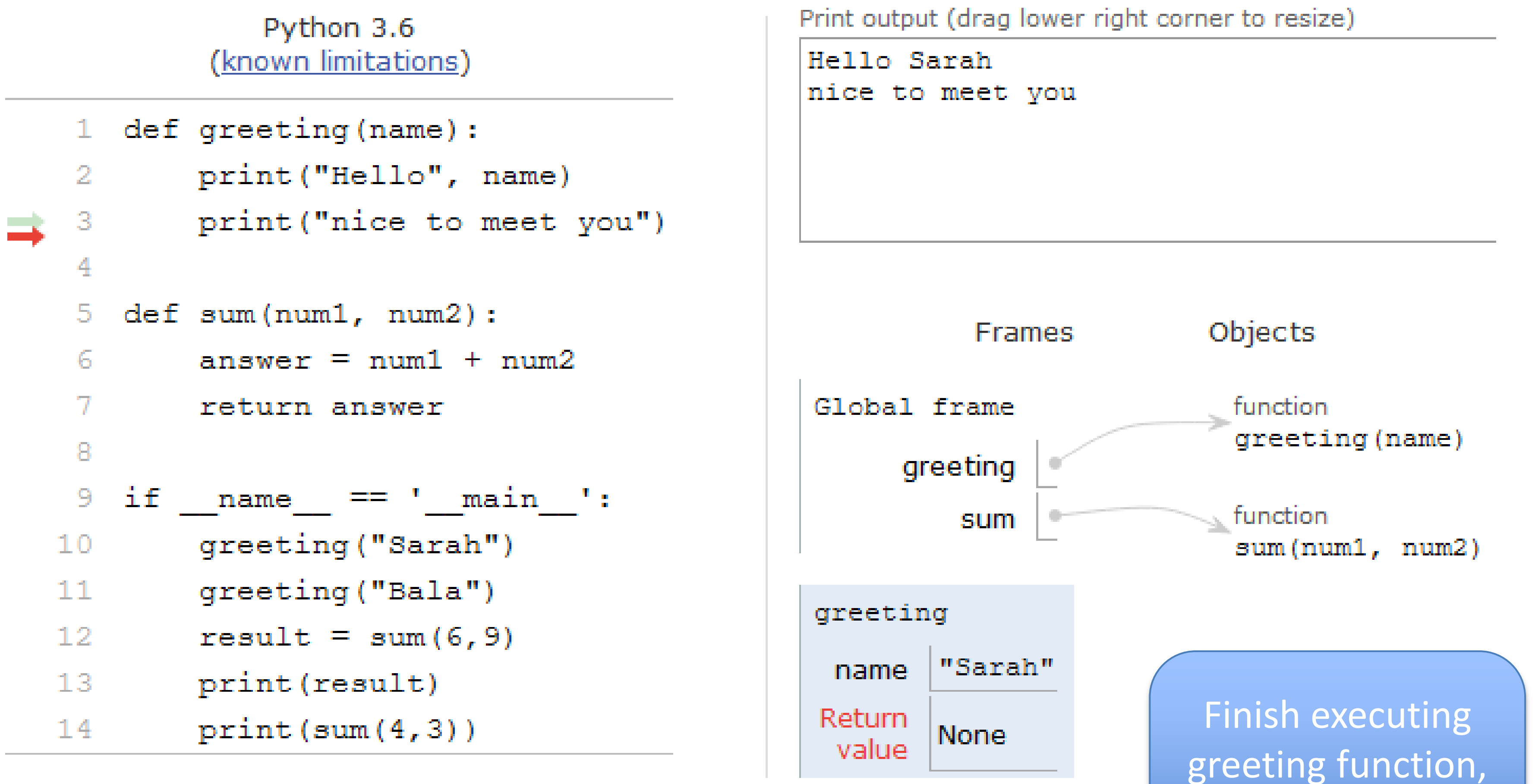


Python 3.6
(known limitations)

```
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))
```

Call greeting and pass value "Sarah" to name

Python Tutor Trace: Step 8



Python 3.6
(known limitations)

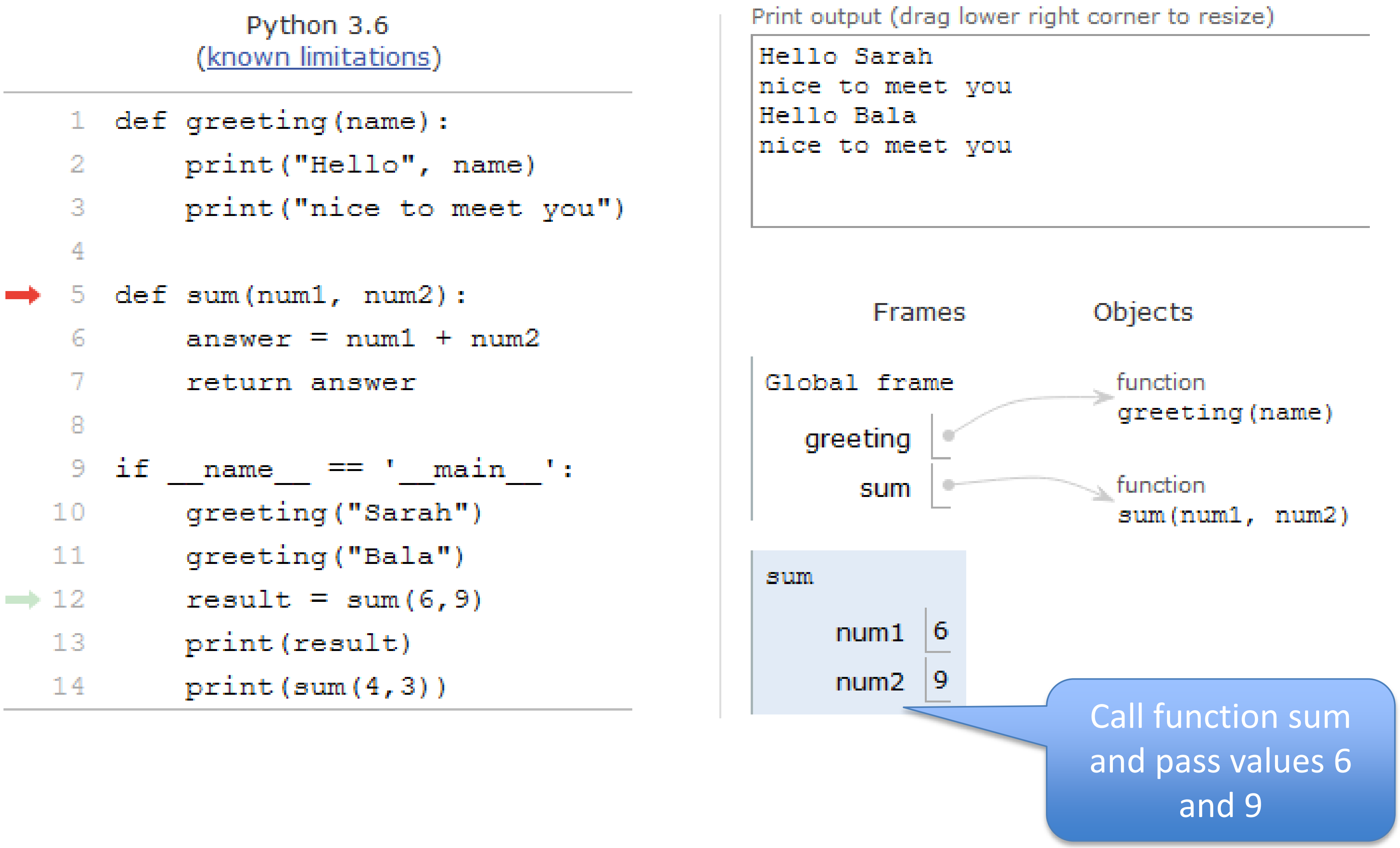
```
def greeting(name):
    print("Hello", name)
    print("nice to meet you")

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))
```

Print output (drag lower right corner to resize)

```
Hello Sarah
nice to meet you
```

Finish executing greeting function, no return value, so return None

Python Tutor Trace: Step 15

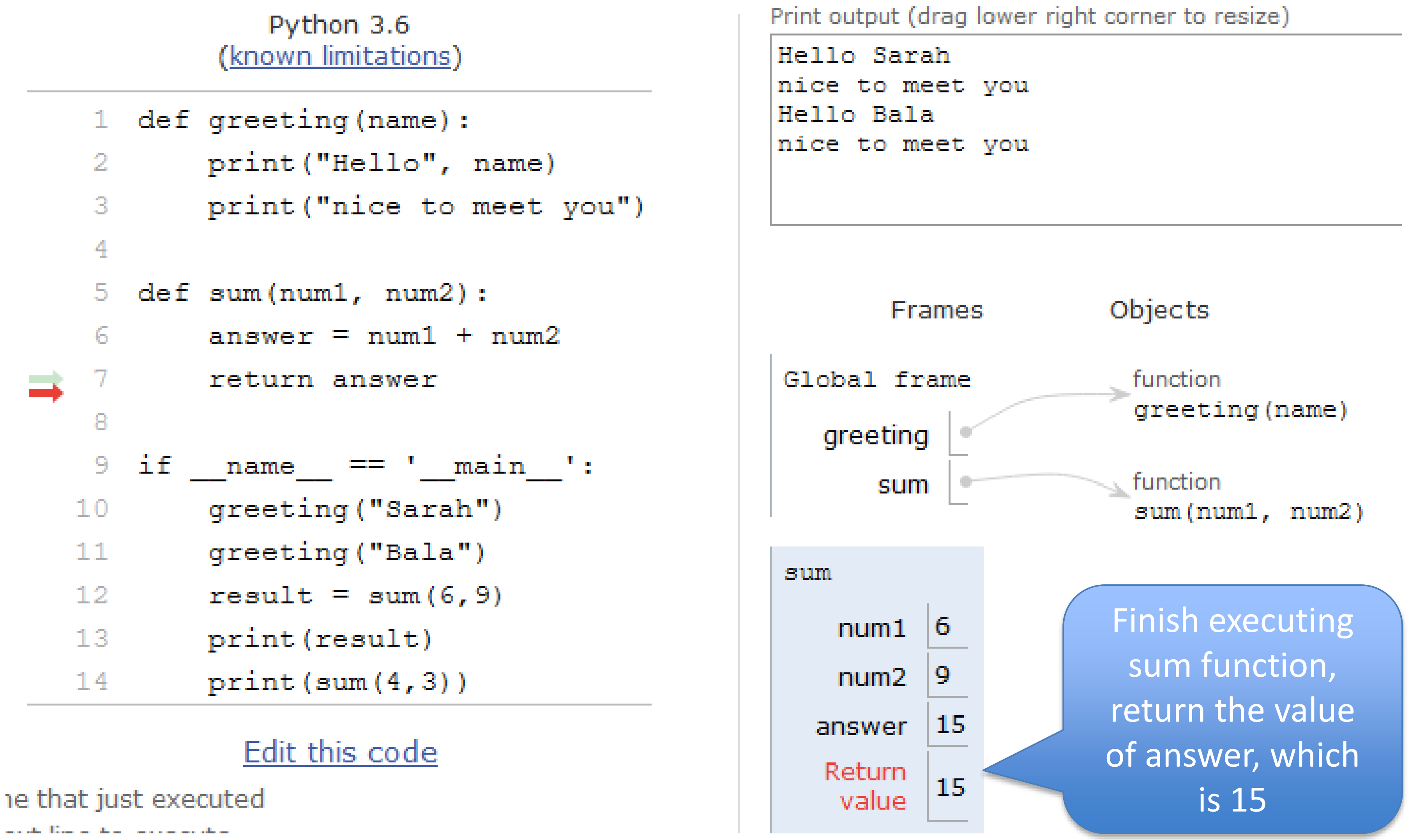


```
def greeting(name):
    print("Hello", name)
    print("nice to meet you")

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))
```

Print output (drag lower right corner to resize)

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

Python Tutor Trace: Step 18



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))
```

Edit this code

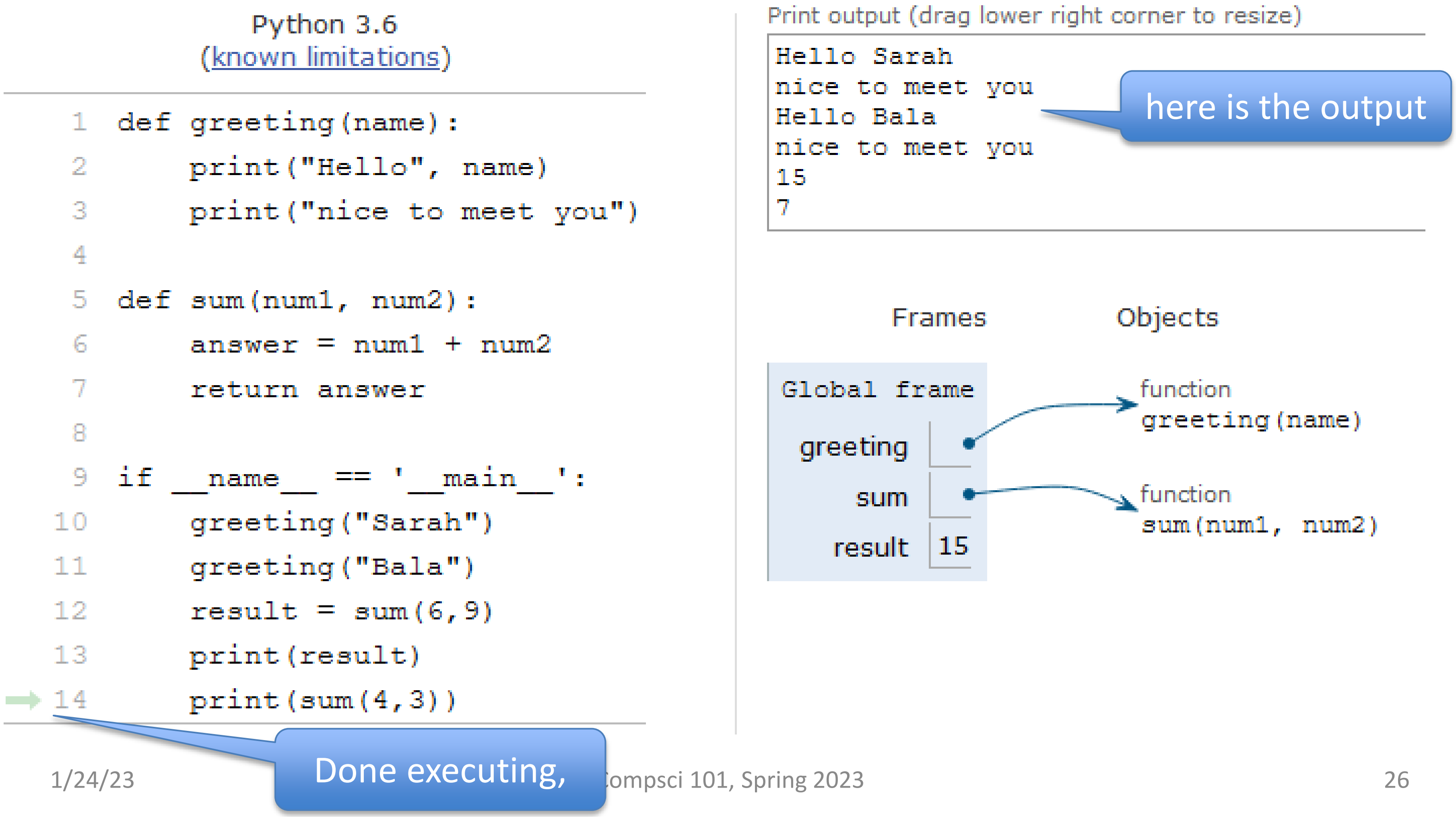
ne that just executed

Print output (drag lower right corner to resize)

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

Finish executing sum function, return the value of answer, which is 15

Python Tutor Trace: Step 24



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))
```

Print output (drag lower right corner to resize)

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

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/101s23-0124-1

- **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")
```

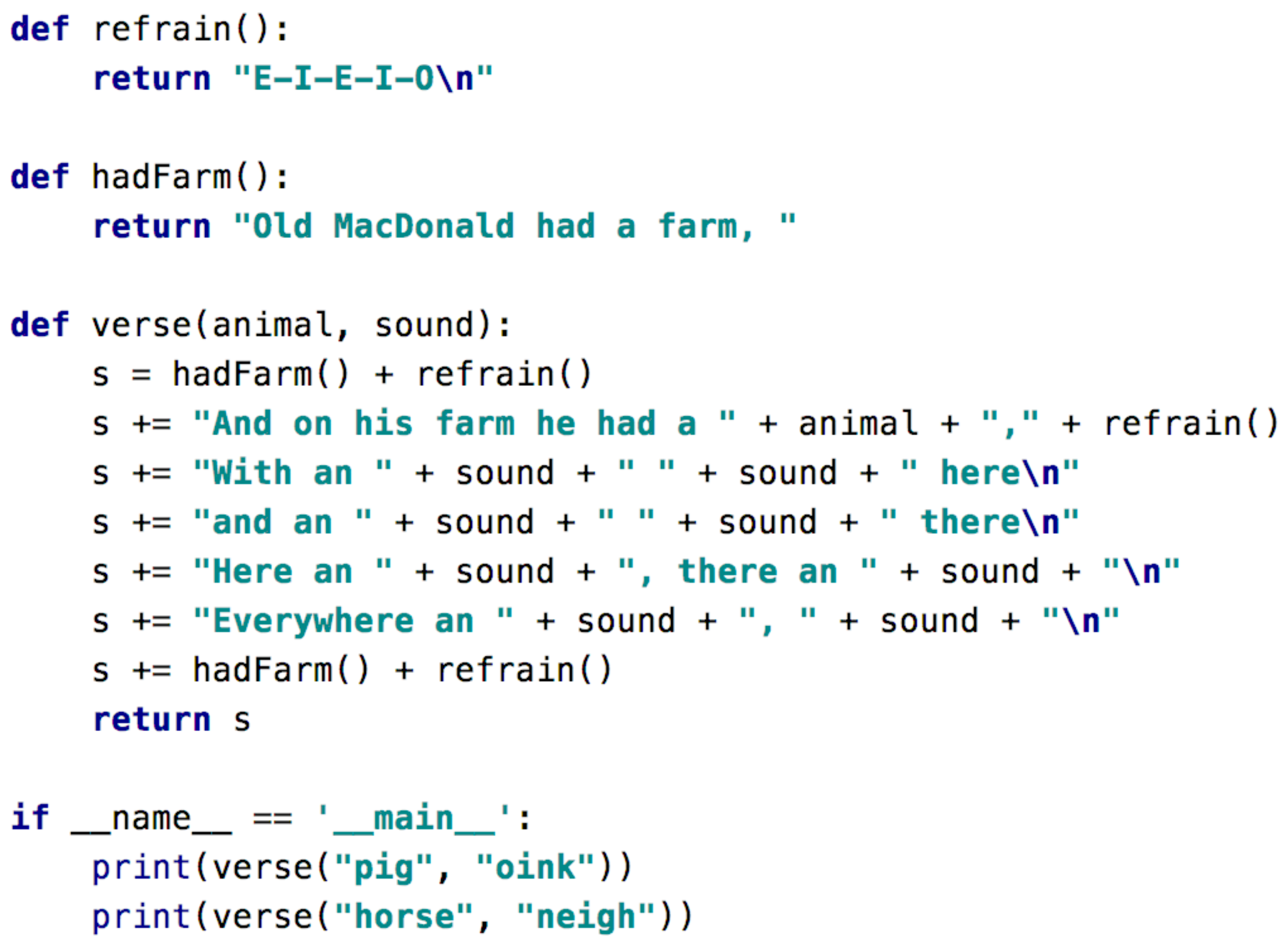
How to make code better?

```
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")
```

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")
```

BetterOldMcDonald.py



```python
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"))
```

BetterOldMcDonald.py

```
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"))
```

Move repetitive strings to own function

Make verse specific strings into parameters

Build the string and then return

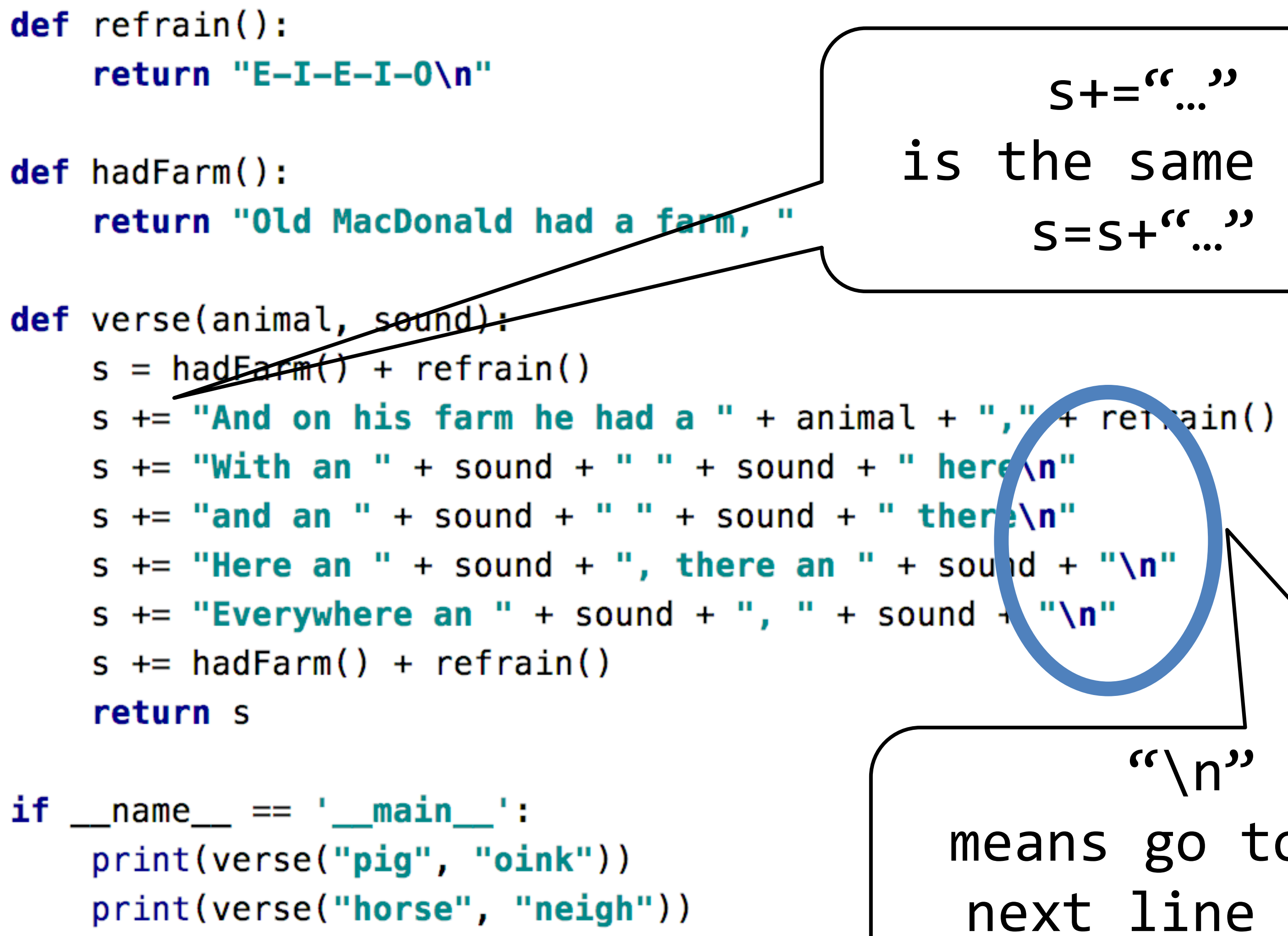
What's new?

WOTO-2 Old MacDonald http://bit.ly/101s23-0124-2

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

What's new?

BetterOldMcDonald.py



```
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"))
```

s+="…"
is the same as:
s=s+"…"

"\n"
means go to the next line when string is printed

Putting together concepts we have seen

BetterOldMcDonald.py

```python
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"))
```

Function call inside another function call

Putting together concepts we have seen

BetterOldMcDonald.py

```
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"))
```

Two functions both return a string, put the two strings together

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
```

Selection Syntax

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
```

IF condition is true, execute code in Block A

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

IF condition is true, execute code in Block A,
Otherwise
Execute code in Block B

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

IF condition is true, execute code in Block A,
Else if second condition true, execute code in Block B
Otherwise
Execute code in Block C

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?

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?
- **if – required**
- **elif – optional, as many as needed**
- **else – optional, no condition**

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
```

Selection Syntax

```
if BOOLEAN_CONDITION:
    CODE_BLOCK_A
```

ONE if with One code block

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

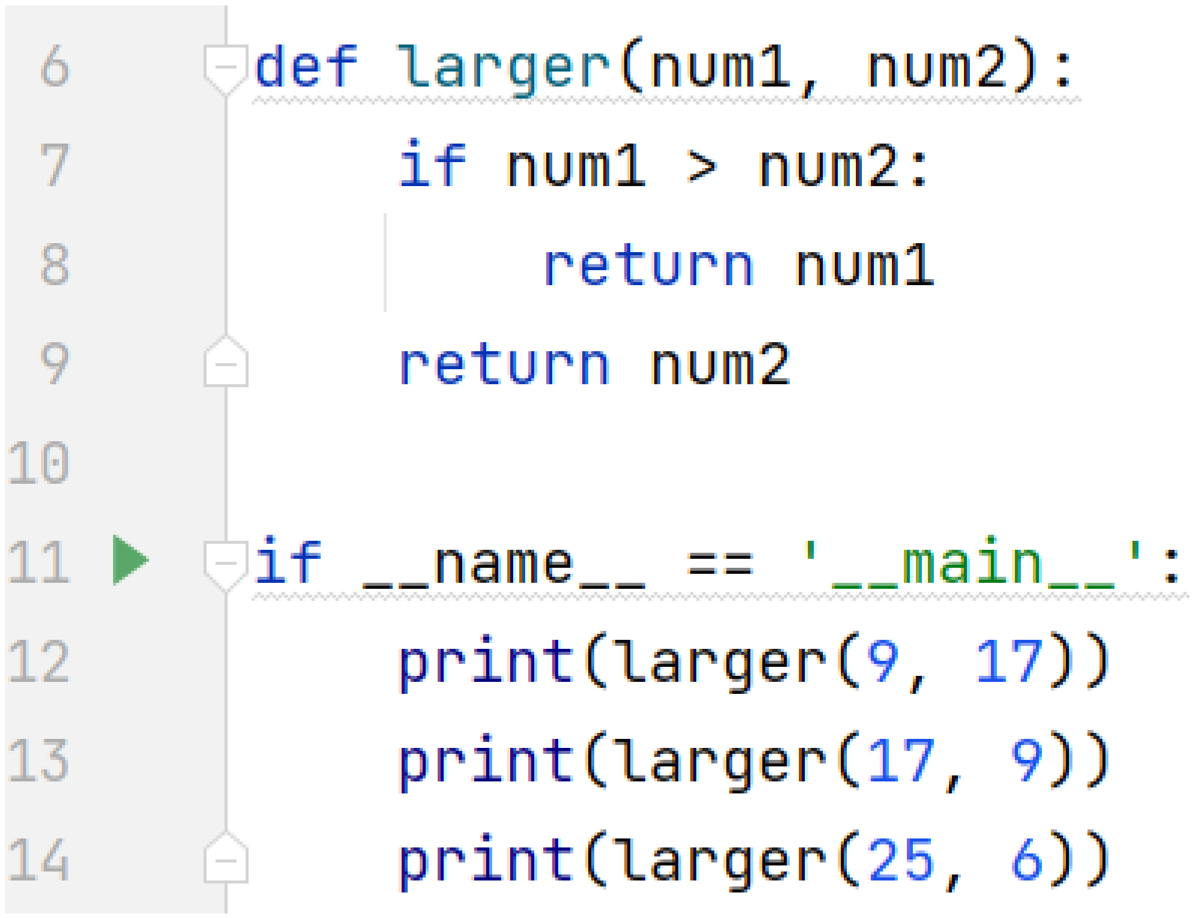
ONE if with Two parts, two code blocks

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

ONE if with Three parts, three code blocks

Each of these is just one IF statement,
So only one CODE Block is executed

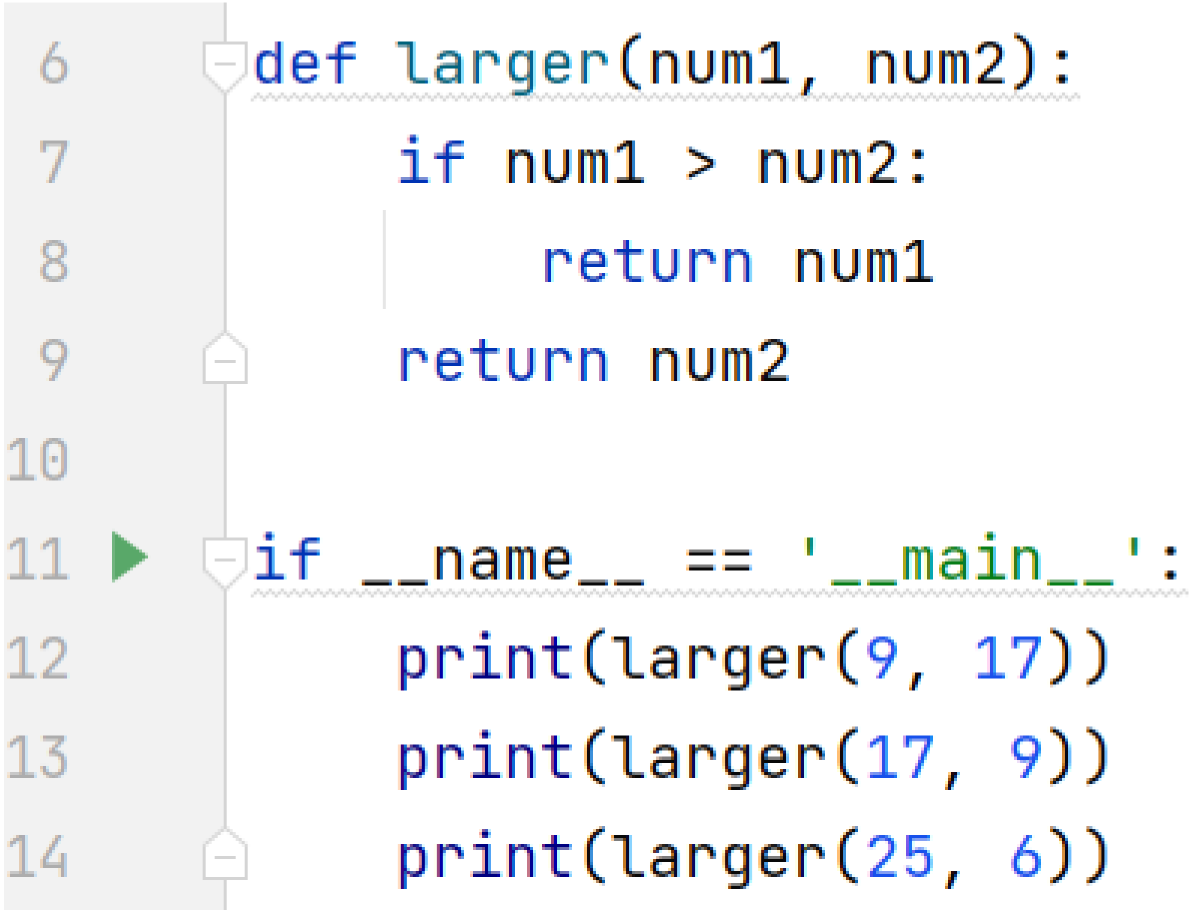
Example: If



```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:

Example: If

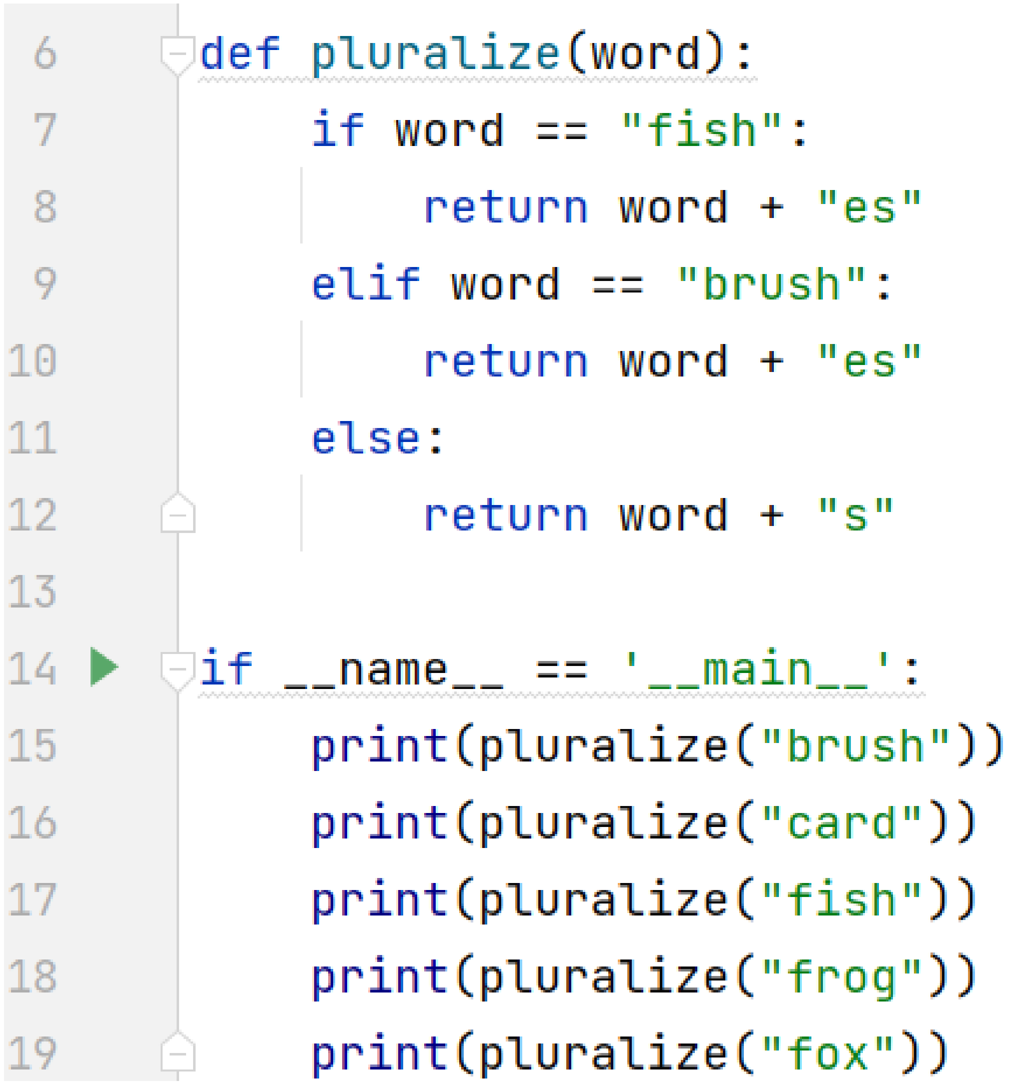


```python
def larger(num1, num2):
    if num1 > num2:
        return num1
    return num2

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

Output:

```
17
17
25
```

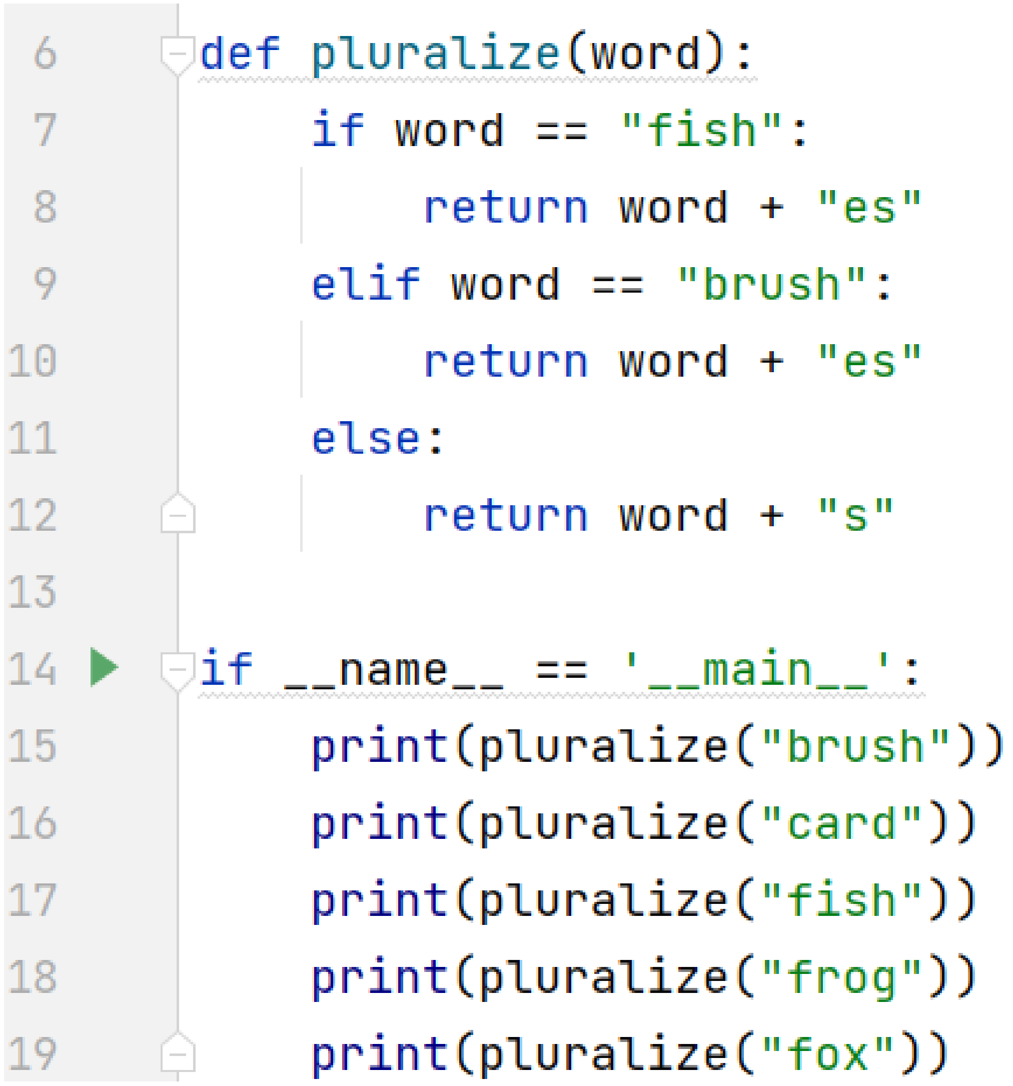
Example2: If-Elif-Else



```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:

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:

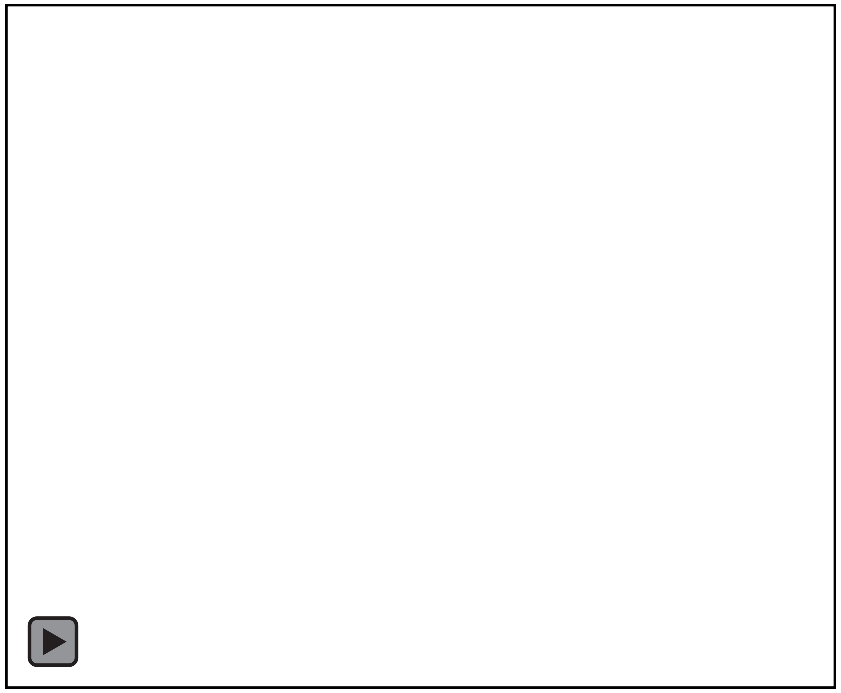
```
brushes
cards
fishes
frogs
foxs
```

Randomness

- **Want things to happen randomly**
- **Games are not interesting if the same things happen every time you play them!**

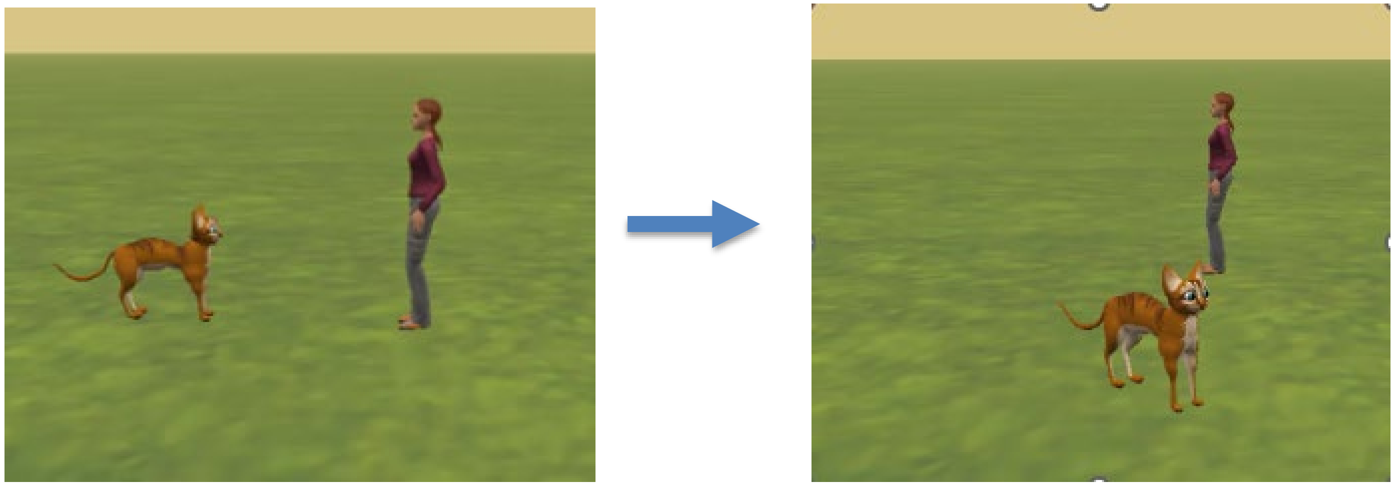
Cat Jumping Not Random

Cat always jumps to its right



Cat Jumping Not Random

Cat always jumps to its right

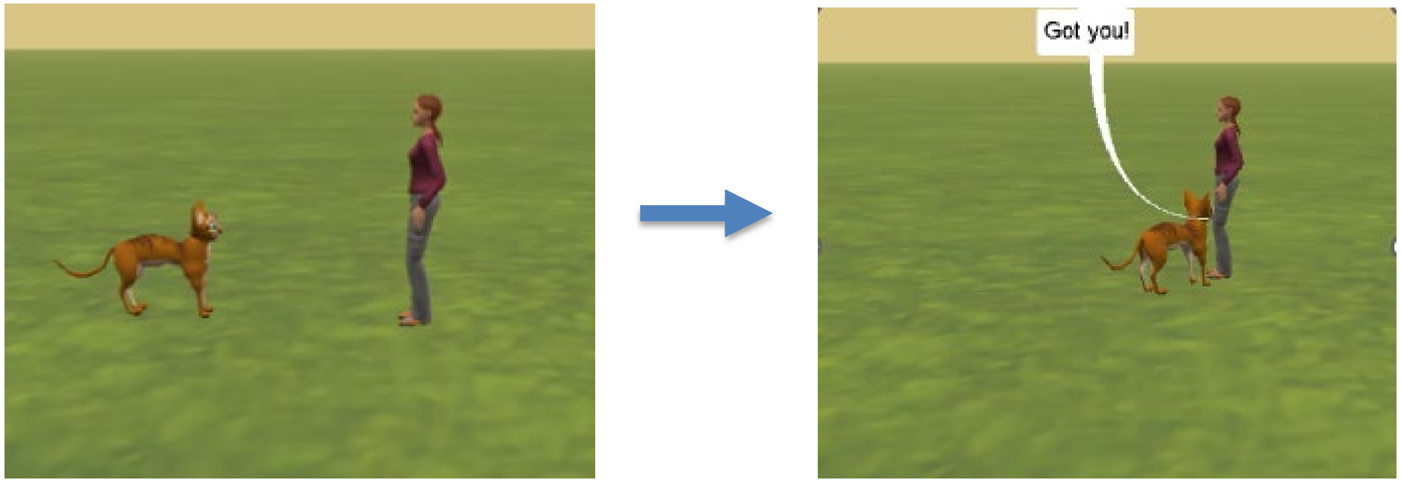


Cat Jumping Random Direction

Cat jumps right or left, randomly

Cat Jumping Random Direction

Cat jumps right or left, randomly



Randomness in Python? 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:** module name arguments
  - random.randint(*a*, *b*)
  - Return a random integer *N* such that a <= N <= b.

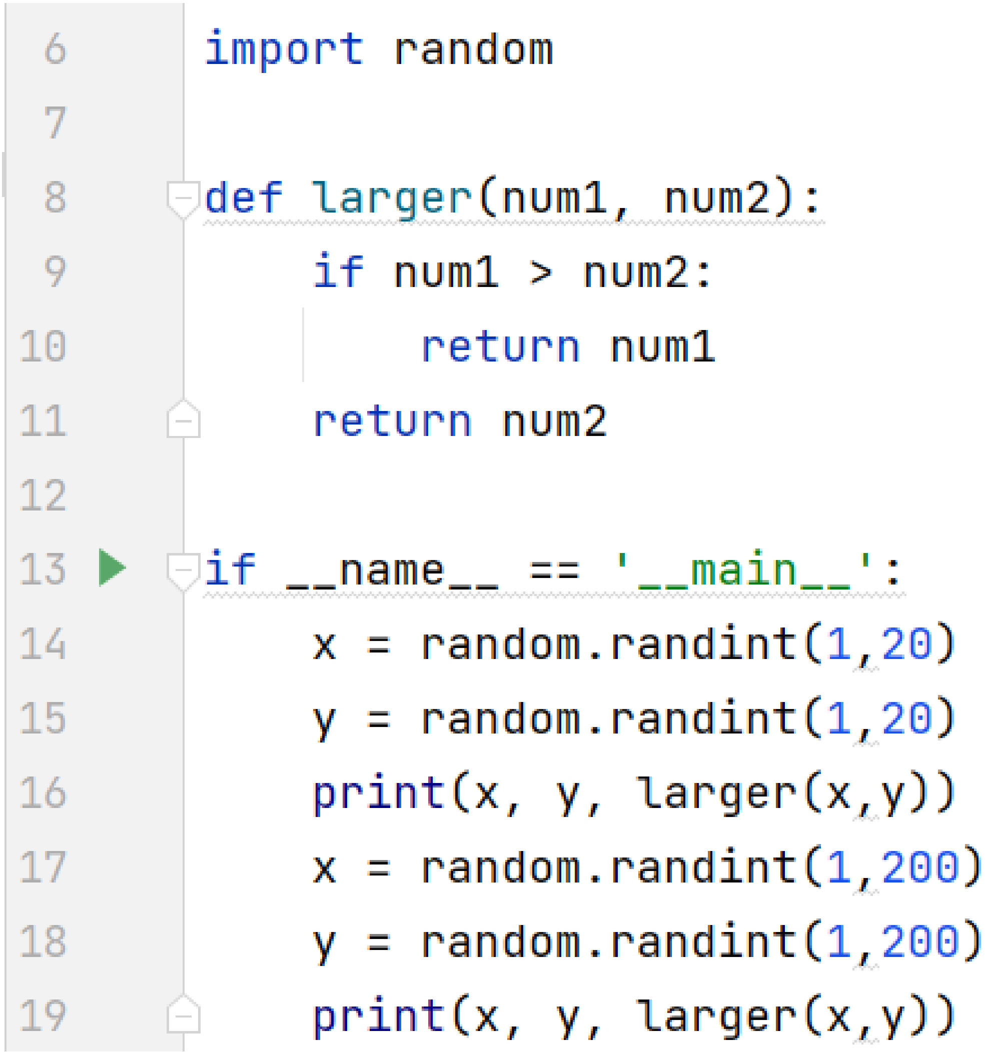
dot function name

Randomness in Python? 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 <= N <= b.

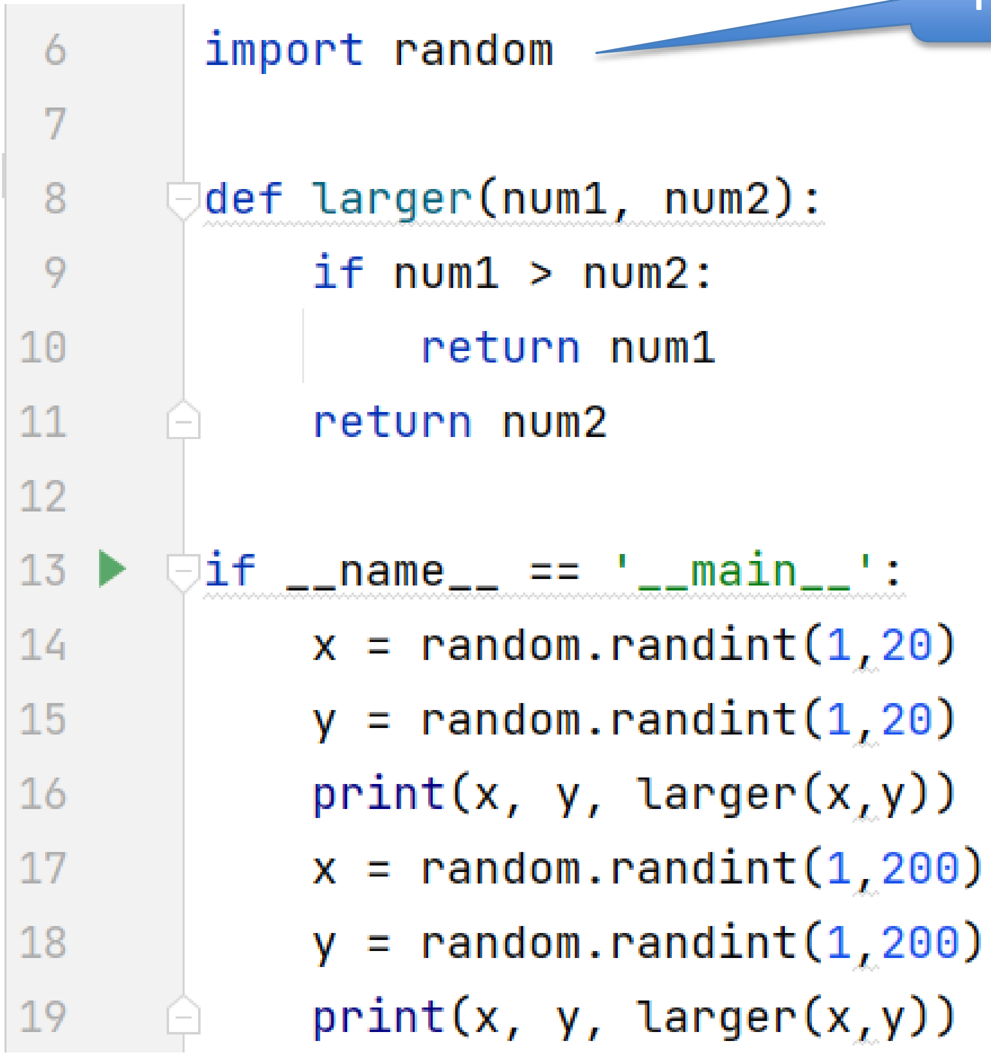
Example: Random



```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))
```

Must import random to use

Output:

20 5 20

78 22 78

Run again…

Output:

17 6 17

5 123 123

Different values every time you run program

WOTO-3

http://bit.ly/101s23-0124-3