Compsci 101
7-steps, Functions, Order of Execution

Susan Rodger
September 6, 2022

Specification

```
filename: Laundry.py

def minutesNeeded(m):
    """
    This function returns the integer number of minutes
    """
    Return integer number of minutes
```

9/6/22  Compsci 101, Fall 2022
C is for ...

- **Computer Science and Computing**
  - It’s what we do
- **Collaboration**
  - Review the policy
- **Cookies**
  - Good for the web and for …
- **CSV**
  - Comma Separated Values: Data
Ayanna Howard

- Educator, Researcher and Innovator
- Professor & Chair of the School of Interactive Computing, Georgia Tech
- Now Dean of Engineering at The Ohio State University!
- Robotics – Robots and Bias, Robots changing lives of children with disabilities, Robots beyond part of the family
- Top 50 U.S. Women in Tech, Forbes, 2018

I believe that every engineer has a responsibility to make the world a better place. We are gifted with an amazing power to take people’s wishes and make them a reality.
Announcements

• Lab 01 Friday,
  • Complete Prelab before going to lab
• APT-1 out today, due Thursday, September 15
• Assignment 0 due Thursday, September 8
• Sakai quizzes on readings due 10:15am on date due
  • Get three tries, score highest score
  • First two weeks we allow you to submit late
  • First 5 quizzes turn off, 10:15am Sept 13

• Read Ed Discussion Every Day – You will learn things!
• Reminder: Ed Discussion back channel in lecture!
PFTD

- Functions
- Order of execution
- 7 steps of programming
- APTs
- Testing and Submitting APTs
What is a Function?

• Function has:
  • Name
  • Maybe inputs
  • Processes or calculates something
  • Has a result
Functions in the Real World: URL in webpage

• Function has:
  • Name: “Search”
  • Input: www.duke.edu
  • Calculates:
  • Returns back:
Functions in the Real World: URL in webpage

- Function has:
  - Name: “Search”
  - Input: www.duke.edu
  - Calculates: Figures out where web page is
  - Returns back: the actual web page
Functions in the Real World: calculator

• Function has:
  • Name: calculator
  • Input: number(s), operator
    • Example: 25, squareroot
  • Calculates:
  • Returns back:
Functions in the Real World: calculator

• Function has:
  • Name: calculator
  • Input: number(s), operator
    • Example: 25, squareroot
  • Calculates: value of expression
  • Returns back:
Functions in the Real World: calculator

• Function has:
  • Name: calculator
  • Input: number(s), operator
    • Example: 25, squareroot
  • Calculates: value of expression
  • Returns back: 5
Functions in the Real World: Counting words in Microsoft Word

- Function has:
  - Name:
  - Input:
  - Calculates:
  - Returns back:
Functions in the Real World: Counting words in Microsoft Word

- **Function has:**
  - **Name:** Word Count
  - **Input:** contents of the document (e.g. a story)
  - **Calculates:** counts number of words
  - **Returns back:** number of words (e.g. 352)
Built-in Python Function – `len()` already exists, you use it

- `len()` function
- Function has:
  - Name: `len`
  - Input: a string
  - Calculates: number of characters in string
  - Returns back: number

Examples:

```
x = len("duke")
# value of x:
y = len("computer")
```
Built-in Python Function – len() already exists, you use it

• **len() function**

• **Function has:**
  • Name: `len`
  • Input: a string
  • Calculates: number of characters in string
  • Returns back: number

**Examples:**

x = len("duke")
# value of x: 4

y = len("computer")
# value of y: 8
Built-in Python Function – `str()` already exists, you use it

- **str() function**
- **Function has:**
  - Name: `str`
  - Input: an expression
  - Calculates: string version of expression’s value
  - Returns back: string

**Examples:**

```python
x = str(623)
# value of x: "623"
```

```python
y = len(str(2**8))
# value of y: 3
```

```python
z = str(6 + 8.3)
```
Built-in Python Function – `str()` already exists, you use it

- **str() function**
- **Function has:**
  - Name: `str`
  - Input: an expression
  - Calculates: string version of expression’s value
  - Returns back: string

**Examples:**

```
x = str(623)
# value of x: “623”

y = len(str(2**8))
   = len(str(256))
   = len(“256”)  # value of y: 3

z = str(6 + 8.3)
# value of z: “14.3”
```
Other Python built-in functions

- **type(something)**
  - Returns type of variable something

- **int(7.8)**
  - Returns integer value of decimal number, e.g. 7

- **float(4)**
  - Returns float value of integer, e.g. 4.0
print() function

- General function has:
  - Name
  - Maybe inputs
  - Processes or calculates something
  - Has a result

- print("hi cat")
  - Name:
  - Input:
print() function

- General function has:
  - Name
  - Maybe inputs
  - Processes or calculates something
  - Has a result

- print(“hi cat”)
  - Name: print
  - Input: “hi cat”
  - “processes”, generates output
  - No return value, returns None

No return value!
Example with lines numbered:

```python
1  x = float(6)
2  print("x is", x)
3  y = print("x is", x)
4  print("y is", y)
```

Output:
Example with lines numbered:

1  x = float(6)
2  print("x is", x)
3  y = print("x is", x)
4  print("y is", y)

Output:

The variable x is assigned the value float(6) calculates
Example with lines numbered:

1  x = float(6)
2  print("x is", x)
3  y = print("x is", x)
4  print("y is", y)

Output:

x is 6.0

Print does not return a value, so there is no "=", since there is no value to catch
Example with lines numbered:

1  x = float(6)
2  print("x is", x)
3  y = print("x is", x)
4  print("y is", y)

Output:

x is 6.0
x is 6.0

What happens if we try to catch the return value in y?

No return value, so None is assigned to y

The RHS executes, and the print prints to output
Example with lines numbered:

1  x = float(6)
2  print("x is", x)
3  y = print("x is", x)
4  print("y is", y)

Output:

x is 6.0
x is 6.0
y is None

Correct way to use print
Wrong way to use print
The print function does NOT return a value. It just prints output.
Writing your own Python function

• Format:

```python
def nameOfFunction(parameters):
    <body, or lines of code>
    return value  # optional, but likely
```

• Example define function:

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
```

• Use or call function:

```python
answer = inchesToCentimeters(10.0)
print(answer)
```

Output:

```
25.4
```
Writing your own Python function

• Format:
  ```python
def nameOfFunction(parameters):
    <body, or lines of code>
    return value  # optional, but likely
  ```

• Example define function:
  ```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
  ```
Writing your own Python function

• Format:

    def nameOfFunction(parameters):
        <body, or lines of code>
        return value  # optional, but likely

• Example define function:

    def inchesToCentimeters(inches):
        centi = inches * 2.54
        return centi

• Use or call function:

    answer = inchesToCentimeters(10.0)
    print(answer)
Writing your own Python function

• Format:
  def nameOfFunction(parameters):
      <body, or lines of code>
      return value  # optional, but likely

• Example define function:
  def inchesToCentimeters(inches):
      centi = inches * 2.54
      return centi

• Use or call function:
  answer = inchesToCentimeters(10.0)
  print(answer)
Writing your own Python function

- **Parameter**
  - Variable, place holder for a value
  - In parenthesis in first line of definition of function

- **Argument**
  - Expression or value
  - In parenthesis when calling or using a function

- **Example:**
  
  ```python
  def inchesToCentimeters(inches):
      centi = inches * 2.54
      return centi
  ```

- **Use or call function:**
  
  ```python
  answer = inchesToCentimeters(10.0)
  print(answer)
  ```
Writing your own Python function

• Parameter
  • Variable, place holder for a value
  • In parenthesis in first line of definition of function

• Argument
  • Expression or value
  • In parenthesis when calling or using a function

• Example:
  ```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
  
  answer = inchesToCentimeters(10.0)
  print(answer)
```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output:

Note function inchesToCentimeter is on line 8
What happens when executes?

```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
```

```python
if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output:

Note lines below line 13 are indented 4 spaces each.

Ignore lines indented, so next line is line 13.
If `__name__ == '__main__'` is special and means:
Start executing program on next line
What happens when executes?

def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)

Evaluate the right hand side of the “=”
Call the function inchesToCentimeter
Pass the argument 10.0 for the parameter inches
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
```

Output:

```
if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Execution moves to line 8 where the definition of function inchesToCentimeters is. inches has the value 10.0
What happens when executes?

```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

The RHS `inches * 2.54` is calculated as 25.4. Then `centi` is assigned the value 25.4.
What happens when executes?

```
8  def inchesToCentimeters(inches):
9    centi = inches * 2.54
10   return centi
13  if __name__ == '__main__':
14    answer = inchesToCentimeters(3.0)
15    print(answer)
16    answer = inchesToCentimeters(3.0)
17    print(answer)
```

Output:

```
25.4
```

The value of the variable centi (25.4) is returned to the RHS of line 14 where the function was called.
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output:

answer is assigned the return value 25.4 and line 14 has completed executing
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output: 25.4

The value of variable answer is printed
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output: 25.4

Evaluate the right hand side of the “="
Pass the argument 3.0 for the parameter inches
What happens when executes?

```python
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output: 25.4

Execution moves to line 8 where the definition of function inchesToCentimeters is. inches has the value 3.0
What happens when executes?

```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output: 25.4

The RHS inches * 2.54 is calculated as 7.62. Then centi is assigned the value 7.62.
What happens when executes?

```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi
```

Output: 25.4

The value of the variable centi (7.62) is returned to the RHS of line 16 where the function was called.
What happens when executes?

```python
8    def inchesToCentimeters(inches):
9        centi = inches * 2.54
10       return centi
12
13  if __name__ == '__main__':
14      answer = inchesToCentimeters(10.0)
15      print(answer)
16      answer = inchesToCentimeters(3.0)
17      print(answer)
```

Output: 25.4

answer is assigned the return value 7.62 and line 16 has completed executing
What happens when executes?

```
def inchesToCentimeters(inches):
    centi = inches * 2.54
    return centi

if __name__ == '__main__':
    answer = inchesToCentimeters(10.0)
    print(answer)
    answer = inchesToCentimeters(3.0)
    print(answer)
```

Output:
25.4
7.62

The value of variable answer is printed.
Let’s go see this in Pycharm and add a function

```python
def pluralize(word):
    word = word + "es"
    return word
```

Add this function

```python
newWord = pluralize("fish")
print(newWord)
word1 = "dress"
word2 = pluralize(word1)
print(word2)
word1 = "book"
print(pluralize(word1))
```

Add these lines of code that call the function
WOTO – Working Together (breakout groups)

• Given a bitly link
  • Type it in OR click on it on the calendar page

• What you should do:
  • Introduce yourselves
  • Each person fills out google form
  • Put in your name, email and netid
  • Discuss each question and fill out
  • Be mindful of time
Details: print(addTen(addTen(x))))

print(addTen(addTen(x))))
print(addTen(addTen(5))))
print(addTen(15)))
print(25)

Output:
25
APTs in 101 and 201

- Algorithm Problem-solving and Testing
  - Algorithm that’s Automatically Tested
  - In use at Duke since 2003, million+ APTs solved

- Given a problem statement
  - Read, think, plan on paper …
  - Write a function to solve the problem
  - Submit the code for testing, debug if necessary

- Where do you start with problem solving?
First part: devise the algorithm
- The meta-problem solving piece
- Big/complex enough to be 4 steps (more shortly)
The Seven Steps
Programming Process: High-level

- After devising the algorithm, translate to code
  - Plan first, then code
  - Bridge analogy: blue prints, then construction
  - Essay analogy: outline, then prose
Next test our program
   • Testing important, often under-taught skill
The Seven Steps
Programming Process: High-level

• Ideally would be correct first time; may need to debug
  • Identify problem (with science!)
  • Return to appropriate prior step to fix the problem
The Seven Steps
Programming Process: High-level

- Work through cycle until program works
Steps 1—4: Devise Algorithm

- Steps 1—4: devise the algorithm
  - Learn to do this well, be an excellent programmer
  - Language: does not matter
Steps 1—4: Example:
Calculate the average of two numbers
Steps 1—4: Example: Calculate the average of two numbers

- Step 1: \(2 + 5 = 7, \frac{7}{2} = 3.5\)
- Step 2:
  - Add \(2 + 5\) and get 7
  - Divide 7 by 2 and the result is 3.5
Steps 1—4: Example:
Calculate the average of two numbers

(1) Work an instance yourself
(2) Write down exactly what you just did
(3) Generalize your steps from (2)
(4) Test your steps
(5) Translate to Code

• Step 3:
  • Two variables num1 and num2
  • Add the two numbers together:
    result is num1 + num2
  • Divide the result by 2 and you have the answer
    answer is result / 2
Steps 1—4: Example:
Calculate the average of two numbers

- Step 4: Try a different example
  - Use 8 and 6, num1 is 8, num2 is 6
  - Add the two numbers together:
    - result is num1 + num2, is 14
  - Divide the result by 2 and you have the answer
    - Answer is result/2, which is 7

- IT WORKS!
Step 5: let’s convert it to code!

- Go to Pycharm
- We will also:
  - Test it – Step 6
  - Debug it – Step 7
Seven Steps

Steps 1-4

1. Work an instance yourself
2. Write down exactly what you just did
3. Generalize your steps from (2)
4. Test your steps
5. Translate to Code

Steps 1-7

Steps 1-4: Devise Algorithm
Step 5: Translate to Code
Step 6: Test Program
Step 7: Debug Program

Algorithmic Problem
Implementation Problem
Program is Incorrect
Program appears correct
Success!
Solving Laundry APT

- Navigate to APTs in class website and ...

CompSci 101, Fall 2022

**APT Quiz**

Start the APT quiz on Sakai under quizzes, but not until you are ready to take the quiz.

**APT*s**

See below for hints on what to do if your APT doesn’t run.

For each problem in an APT set, complete these steps by the due date:

- first click on the APT set below to go to the APT page.
- write the code, upload the file and click the Submit link
- check your grade on the grade code page by clicking on check submissions

In solving APTs, your program should work for all cases, not just the test cases we provide.

<table>
<thead>
<tr>
<th>APT</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT-1</td>
<td>Sept. 15</td>
</tr>
</tbody>
</table>
Solving Laundry APT

APT Grading: CompSci 101, Fall 2022

This is the webpage for grading and submitting your APTs.

Check Grades
check submissions

<table>
<thead>
<tr>
<th>Problem Set 1</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT-1, Due on September 15, Complete all six of them</td>
<td></td>
</tr>
<tr>
<td>- IntroAPT</td>
<td>Do first, explains apts</td>
</tr>
<tr>
<td>- Bogsquare</td>
<td></td>
</tr>
<tr>
<td>- Cone</td>
<td></td>
</tr>
<tr>
<td>- Grayscale</td>
<td>in Lecture on 9/6</td>
</tr>
<tr>
<td>- Laundry</td>
<td>in Lab 1 on 9/9</td>
</tr>
<tr>
<td>- Gravity</td>
<td></td>
</tr>
</tbody>
</table>

Test file: Browse... No file selected.

9/6/22
Solving Laundry APT

• Navigate to APTs in class website and ...

Problem Statement

Consider the problem of trying to do a number of loads of laundry, given only one washer and one dryer. Washing a load takes 25 minutes, drying a load takes 25 minutes, and folding the clothes in a load takes 10 minutes, for a total of 1 hour per load (assuming that the time to transfer a load is built into the timings given). 10 loads of laundry can be done in 10 hours, 600 minutes, using the method of completing one load before starting the next one. Though it can be done faster, see examples.

Specification

```python
filename: Laundry.py

def minutesNeeded(m):
    """
    Return integer number of minutes to launder m (integer) loads
    """

    # you write code here
```

Write the method, minutesNeeded, that returns the shortest time needed to do m loads of laundry. In other words, given an integer value representing the number of loads to complete, m, determine the smallest number of minutes needed to complete all loads of laundry.
Not ready for coding yet!!!!!
Solving Laundry APT – Step 1

• What is important info?

Problem Statement

Consider the problem of trying to do a number of loads of laundry, given only one washer and one dryer. Washing a load takes 25 minutes, drying a load takes 25 minutes, and folding the clothes in a load takes 10 minutes, for a total of 1 hour per load (assuming that the time to transfer a load is built into the timings given). 10 loads of laundry can be done in 10 hours, 600 minutes, using the method of completing one load before starting the next one. Though it can be done faster, see examples.

Write the method, minutesNeeded, that returns the shortest time needed to do m loads of laundry, representing the number of loads to complete, m, determine the smallest number of minutes needed.
Solving Laundry APT

- $m = 1$

- Return: $25 + 25 + 10 = 60$ minutes
Solving Laundry APT

- \( m = 2 \)

- Return: 25 + 25 + 25 + 10

- = 85 minutes
Write down what we just did for m=2

• Washed first load (25 minutes)
• Dried first load and washed second load (25 min)
• Folded first load dried second load (25 min)
• Folded second load (10 min)
• Total time was 25 + 25 + 25 +10 = 85 minutes
Step 1: Work an instance yourself

Step 2: Write down exactly what you just did

Step 3: Generalize your steps

Step 4: Test your steps (with new input)

What should be a variable?

Examples

1. \( m = 1 \)
   
   \[
   \text{returns: 60}
   \]
   
   You must wash minutes.

2. \( m = 2 \)
   
   \[
   \text{returns: 85}
   \]
Solving Laundry APT – Steps 3 and 4


- What is important info?

Problem Statement

Consider the problem of trying to do a number of loads of laundry, given only one washer and one dryer. Washing a load takes 25 minutes, drying a load takes 25 minutes, and folding the clothes in a load takes 10 minutes, for a total of 1 hour per load (assuming that the time to transfer a load is built into the timings given). 10 loads of laundry can be done in 10 hours, 600 minutes, using the method of completing one load before starting the next one. Though it can be done faster, see examples.

Write the method, minutesNeeded, that returns the shortest time needed to do m loads.
Solving an APT

• Create new project
  • File > New Project
  • Existing interpreter (first project you made from installation)

• Create new Python File
  • Right click on project > New > Python File

• Create function within module
  • Name it properly!
Names and Return 0 Submission

- Take small steps to get all green!

<table>
<thead>
<tr>
<th>Test Results Follow (scroll to see all)</th>
<th>Test Results Follow (scroll to see all)</th>
<th>Test Results Follow (scroll to see all)</th>
</tr>
</thead>
<tbody>
<tr>
<td># of correct: 0 out of 19</td>
<td># of correct: 12 out of 19</td>
<td># of correct: 19 out of 19</td>
</tr>
<tr>
<td>1 fail</td>
<td>1 pass</td>
<td>1 pass</td>
</tr>
<tr>
<td>2 fail</td>
<td>2 pass</td>
<td>2 pass</td>
</tr>
<tr>
<td>3 fail</td>
<td>3 pass</td>
<td>3 pass</td>
</tr>
<tr>
<td>4 fail</td>
<td>4 pass</td>
<td>4 pass</td>
</tr>
<tr>
<td>5 fail</td>
<td>5 pass</td>
<td>5 pass</td>
</tr>
<tr>
<td>6 fail</td>
<td>6 pass</td>
<td>6 pass</td>
</tr>
<tr>
<td>7 fail</td>
<td>7 pass</td>
<td>7 pass</td>
</tr>
<tr>
<td>8 fail</td>
<td>8 pass</td>
<td>8 pass</td>
</tr>
<tr>
<td>9 fail</td>
<td>9 pass</td>
<td>9 pass</td>
</tr>
<tr>
<td>10 fail</td>
<td>10 pass</td>
<td>10 pass</td>
</tr>
<tr>
<td>11 fail</td>
<td>11 pass</td>
<td>11 pass</td>
</tr>
<tr>
<td>12 fail</td>
<td>12 pass</td>
<td>12 pass</td>
</tr>
<tr>
<td>13 fail</td>
<td>13 pass</td>
<td>13 pass</td>
</tr>
<tr>
<td>14 fail</td>
<td>14 pass</td>
<td>14 pass</td>
</tr>
<tr>
<td>15 fail</td>
<td>15 pass</td>
<td>15 pass</td>
</tr>
<tr>
<td>16 fail</td>
<td>16 pass</td>
<td>16 pass</td>
</tr>
<tr>
<td>17 fail</td>
<td>17 pass</td>
<td>17 pass</td>
</tr>
<tr>
<td>18 fail</td>
<td>18 pass</td>
<td>18 pass</td>
</tr>
<tr>
<td>19 fail</td>
<td>19 pass</td>
<td>19 pass</td>
</tr>
</tbody>
</table>
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
Testing Laundry – two ways

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

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

```python
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))
```
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