Compsci 101
7-steps, Functions, Order of Execution

Specification

```python
filename: Laundry.py

def minutesNeeded(m):
    """
    Return integer number of minutes
    """
```
Reminder

• Don't sit in the last 5 rows ever
• Also don't sit in that tiny 2 person row ever.

• Come closer and meet someone
C is for ...

• Computer Science and Computing
  • It’s what we do

• Cookies
  • Good for the web and for ...

• CSV
  • Comma Separated Values: Data

• ChatGPT
  • Trained AI model to answer questions
Ayanna Howard

- Educator, Researcher and Innovator
- BS Brown, MS/PhD USC, MBA Claremont
- Was Professor, Georgia Tech
- Now Dean of Engineering at Ohio State
- 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, January 26
• Assignment 0 due Today!
  • Due to Drop/Add -> ok to turn in by Jan 26
• 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 Jan 26

• Read Ed Discussion Every Day – You will learn things!
• Reminder: Ed Discussion back channel in lecture!
We are now in 2cd week of Drop/Add

• **What does that mean?**
  • You cannot add any course without a permission number!

• **If you decide to change your lab section and drop the course and re-add the new lab section**
  • you will NOT be able to re-Add it without a permission number
  • Get that permission number first!
  • Email Prof. Velasco with Subject: CompSci 101
Go over answers from last WOTO

\[ x = 8 \]
\[ y = 3 \]
\[ z = 2.0 \]

\[ \frac{x}{y} * z \]
\[ x + y * y \]
\[ (x+y) * z ** 3 \]

What is \((x + y) * z ** 3\)

193 responses

1/19/23 Compsci 101, Spring 2023
Go over answers from last WOTO

\[ x = 8 \]
\[ y = 3 \]
\[ z = 2.0 \]

\[ \frac{x}{y} \times z \]
\[ x + y \times y \]
\[ (x+y) \times z^{**3} \]
\[ = (x+y) \times (z^{**3}) \]

\[ 5.333333 \]
\[ 17 \]
\[ ** \text{ higher precedence than } * \]
\[ 11 \times 8.0 = 88.0 \]

\[ a = "Duke" \]

\[ a + a \]
\[ a + 3 \]
\[ a \times 3 \]
\[ a + b \]

\[ "DukeDuke" \]
\[ ERROR \]
\[ "DukeDukeDuke" \]
\[ "DukeCoolColors" \]
Go over answers from last WOTO

\[ x = 8 \]
\[ y = 3 \]
\[ z = 2.0 \]

\[ \frac{x}{y} \times z \]
\[ x + y \times y \]
\[ (x+y) \times z^{3} \]

\[ a = "Duke" \]
\[ b = "CoolColors" \]

\[ a + a \]
\[ a + 3 \]
\[ a \times 3 \]
\[ a + b \]
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, square root
  - 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:**

```python
x = len("duke")
# value of x:

y = len("computer")
```
Built-in Python Function – `len()`

already exists, you use it it

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

**Examples:**

```python
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"

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

z = str(6 + 8.3)
# value of z: "14.3"
```
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

**OUTPUT:**
hi cat

• print("hi cat")
  • Name: print
  • Input: "hi cat"
  • processes, generates output
  • Outputs value, 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. \( x = \text{float}(6) \)
2. \( \text{print}("x is", x) \)
3. \( y = \text{print}("x is", x) \)
4. \( \text{print}("y is", y) \)

Output:

\( x \text{ is } 6.0 \)
Example with lines numbered:

1 \( x = \text{float}(6) \)
2 \( \text{print("x is", x)} \)
3 \( y = \text{print("x is", x)} \)
4 \( \text{print("y is", y)} \)

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
```

• **Use or call function:**

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

```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)
```
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

• **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
  
  # Use or call function:
  answer = inchesToCentimeters(10.0)
  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:

Start on line 1 of the file and move line by line.
The first 7 lines are blank or are a comment, ignore.
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

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

Output:

Ignore lines 9 and 10 for now, so next line is line 13. If `__name__ == '__main__'` is special and means:
Start executing program on next line.
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:

Evaluate the right hand side of the "=".
Call the function inchesToCentimeters.
Pass the argument 10.0 for the parameter inches.
Execution moves to line 8 where the definition of function `inchesToCentimeters` is. `inches` has the value 10.0.

Output:

```
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:
- inches: 10.0
- centi: 25.4

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
11
12   if __name__ == '__main__':
13       answer = inchesToCentimeters(3.0)
14       print(answer)
15       answer = inchesToCentimeters(3.0)
16       print(answer)
```

Output:
```
inches: 10.0
centi: 25.4

answer: 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?

```
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: 25.4

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
```

```python
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?

Execution moves to line 8 where the definition of function inchesToCentimeters is. inches has the value 3.0
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)
```

The RHS \( \text{inches} \times 2.54 \) is calculated as 7.62. Then \( \text{centi} \) is assigned the value 7.62.
What happens when executes?

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

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

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?

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

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

answer is assigned the return value 7.62 and line 16 has completed executing

Output:
25.4

answer: 7.62
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
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
```

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

Add this function

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
WOTO: Calling Functions
Details: print(addTen(addTen(x)))

print(addTen(addTen(x)))
print(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?
The Seven Steps
Programming Process: High-level

- 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

• 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
Success!

Algorithmic Problem
Implementation Problem
Program is Incorrect
Program appears correct
Step 7: Debug Program
Solving Laundry APT

- Navigate to APTs in class website and ...

**CompSci 101, Spring 2023 APTs**

**APT Quiz**

There will be two APT Quizzes that are just like APTs but are your own work and are timed 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, select the problem, 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. additional data.

<table>
<thead>
<tr>
<th>APT</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT-1</td>
<td>January 26</td>
</tr>
</tbody>
</table>
Solving Laundry APT

APT Grading: CompSci 101, Spring 2023

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 January 26, 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></td>
</tr>
<tr>
<td>Laundry</td>
<td>in Lecture on 1/19</td>
</tr>
<tr>
<td>Gravity</td>
<td>in Lab 1 on 1/20</td>
</tr>
</tbody>
</table>

Test file: Browse... No file selected. test/run
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.

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
</table>

```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!!!!!
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. 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.
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
Reading an APT

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

Examples

1. \( m = 1 \)
   
   returns: 60
   
   You must wash minutes.

2. \( m = 2 \)
   
   returns: 85

What should be a variable?
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. 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.
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!

Test Results Follow (scroll to see all)

<table>
<thead>
<tr>
<th># of correct: 0 out of 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 fail</td>
</tr>
<tr>
<td>2 fail</td>
</tr>
<tr>
<td>3 fail</td>
</tr>
<tr>
<td>4 fail</td>
</tr>
<tr>
<td>5 fail</td>
</tr>
<tr>
<td>6 fail</td>
</tr>
<tr>
<td>7 fail</td>
</tr>
<tr>
<td>8 fail</td>
</tr>
<tr>
<td>9 fail</td>
</tr>
<tr>
<td>10 fail</td>
</tr>
<tr>
<td>11 fail</td>
</tr>
<tr>
<td>12 fail</td>
</tr>
<tr>
<td>13 fail</td>
</tr>
<tr>
<td>14 fail</td>
</tr>
<tr>
<td>15 fail</td>
</tr>
<tr>
<td>16 fail</td>
</tr>
<tr>
<td>17 fail</td>
</tr>
<tr>
<td>18 fail</td>
</tr>
<tr>
<td>19 fail</td>
</tr>
</tbody>
</table>

Test Results Follow (scroll to see all)

<table>
<thead>
<tr>
<th># of correct: 12 out of 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pass</td>
</tr>
<tr>
<td>2 pass</td>
</tr>
<tr>
<td>3 pass</td>
</tr>
<tr>
<td>4 pass</td>
</tr>
<tr>
<td>5 pass</td>
</tr>
<tr>
<td>6 pass</td>
</tr>
<tr>
<td>7 pass</td>
</tr>
<tr>
<td>8 pass</td>
</tr>
<tr>
<td>9 pass</td>
</tr>
<tr>
<td>10 pass</td>
</tr>
<tr>
<td>11 pass</td>
</tr>
<tr>
<td>12 pass</td>
</tr>
<tr>
<td>13 fail</td>
</tr>
<tr>
<td>14 fail</td>
</tr>
<tr>
<td>15 fail</td>
</tr>
<tr>
<td>16 fail</td>
</tr>
<tr>
<td>17 fail</td>
</tr>
<tr>
<td>18 fail</td>
</tr>
<tr>
<td>19 fail</td>
</tr>
</tbody>
</table>

Test Results Follow (scroll to see all)

<table>
<thead>
<tr>
<th># of correct: 19 out of 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pass</td>
</tr>
<tr>
<td>2 pass</td>
</tr>
<tr>
<td>3 pass</td>
</tr>
<tr>
<td>4 pass</td>
</tr>
<tr>
<td>5 pass</td>
</tr>
<tr>
<td>6 pass</td>
</tr>
<tr>
<td>7 pass</td>
</tr>
<tr>
<td>8 pass</td>
</tr>
<tr>
<td>9 pass</td>
</tr>
<tr>
<td>10 pass</td>
</tr>
<tr>
<td>11 pass</td>
</tr>
<tr>
<td>12 pass</td>
</tr>
<tr>
<td>13 pass</td>
</tr>
<tr>
<td>14 pass</td>
</tr>
<tr>
<td>15 pass</td>
</tr>
<tr>
<td>16 pass</td>
</tr>
<tr>
<td>17 pass</td>
</tr>
<tr>
<td>18 pass</td>
</tr>
<tr>
<td>19 pass</td>
</tr>
</tbody>
</table>
APT Correct → The Green Dance (Fall 2020)