In writing code you do not need to worry about specifying the proper `import statements`. You do not need to worry about getting function or method names exactly right. Assume that all libraries and packages we have discussed are imported in any code you write.
PROBLEM 1:  (*Type Casting*)

Each of the variables below has a *type* and a *value*. The type is one of: list, boolean, int, string, or float. As an example, the type and value are shown in the first row of the table below. Fill in the other type and value entries based on the variable/expression in the first column.

<table>
<thead>
<tr>
<th>variable/expression</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>len([5,3,1])</code></td>
<td>int or integer</td>
<td>3</td>
</tr>
<tr>
<td><code>30 % 15</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>len(&quot;cat&quot;) &lt; len(&quot;cheeseburger&quot;)</code></td>
<td>bool</td>
<td></td>
</tr>
<tr>
<td><code>sum(range(1,5))</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;abcdef&quot;[1:3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>max(&quot;zebra&quot;)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&quot;a&quot;, [0], 3[0]</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;compsci &quot;.upper() + &quot;101&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>4.0 / 8</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 2:  (It's Turtles all the way Down)

Draw the picture resulting from the following turtle commands:

```python
snap = turtle.Turtle()
length = 50
for x in range(10):
    snap.forward(length)
    snap.left(90)
    length += 10
```
The function `change` below is intended to reverse the last and first names when in the format `last, first`, so that the call below returns "bob jones".

```
change("jones, bob")
```

Here is one solution that passes all given tests. You will be asked two questions about this code.

```python
def change(name):
    index = name.find(",")
    return name[index+2:] + " " + name[:index]
```

**Part A**

Explain in words why the first slice used in the return uses `index+2` and why the second slice uses `index`.

**Part B**

Pat looks at the code and says it will generate an error message if it’s called with a string without commas (not allowed in the APT) so that `change("bob jones")` will generate an error. Ryan says no, it won’t generate an error, runs the code, and the call `change("bob jones")` returns the string below (no error is generated).

```
ob jones bob jone
```

Explain why the function generates this return value and does not result in an error.
PROBLEM 4:  (Multiple Choice)

Part A
Consider the following loops:

I. for x in range(1, 20):
II. for x in range(0, 200, 10):

Which of the following statements is true?

A. The loops each run 20 times.
B. The loops each run 21 times.
C. Loop I runs more times than Loop II.
D. Loop II runs more times than Loop I.

Part B
The following function is supposed to be an implementation of the Python function isupper that returns True only if the given string contains nothing but capital letters in it.

```python
def isupper(word):
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    for char in word:
        if char in alphabet:
            return True
    return False
```

Which of the following statements about the code is true?

A. It works as intended.
B. It always returns false.
C. It returns whether or not the first letter only is uppercase.
D. It returns whether or not the last letter only is uppercase.
PROBLEM 5 :  (Play that Funky Music)

Part A
A number is abundant if it is greater than the sum of its proper divisors, that is its divisors other than itself. For example 12 is abundant because \(1 + 2 + 3 + 4 + 6 = 16 > 12\). The first 10 abundant numbers are 12, 18, 20, 24, 30, 36, 40, 42, 48, 54.

Write a boolean function `isAbundant` to return True if its parameter is abundant and False otherwise.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>isAbundant(4)</td>
<td>False</td>
</tr>
<tr>
<td>isAbundant(12)</td>
<td>True</td>
</tr>
<tr>
<td>isAbundant(24)</td>
<td>True</td>
</tr>
<tr>
<td>isAbundant(28)</td>
<td>False</td>
</tr>
</tbody>
</table>

Complete the function `isAbundant` below:

```python
def isAbundant(num):
    """
    return True if int parameter num is abundant and
    returns False otherwise
    """
```
Part B
Write a function `abundantCount` that returns the number of abundant numbers between (and including) parameters first and last. You should call `isAbundant` and assume it works correctly.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>abundantCount(1,11)</td>
<td>0</td>
</tr>
<tr>
<td>abundantCount(1,20)</td>
<td>3</td>
</tr>
<tr>
<td>abundantCount(20,30)</td>
<td>3</td>
</tr>
<tr>
<td>abundantCount(70,80)</td>
<td>4</td>
</tr>
</tbody>
</table>

Complete the function `abundantCount` below:

```python
def abundantCount (start, end):
    """
    return how many numbers between start and end (inclusive) are abundant
    """
```
The `lawFirm` function takes a string of partner names and returns the name of the law firm composed from their last names. This table illustrates what the function `lawFirm` is supposed to return for each given input.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lawFirm(&quot;Robert Duvall&quot;)</code></td>
<td>&quot;Duvall, LLC&quot;</td>
</tr>
<tr>
<td><code>lawFirm(&quot;Ivan the Great and Terrible, Peter the Great&quot;)</code></td>
<td>&quot;Terrible &amp; Great, LLC&quot;</td>
</tr>
<tr>
<td><code>lawFirm(&quot;Jessica Pearson, Daniel Hardman, Harvey Specter&quot;)</code></td>
<td>&quot;Pearson Hardman &amp; Specter, LLC&quot;</td>
</tr>
</tbody>
</table>

Consider the following solutions that are not correct. Each both passes and fails at least one of the examples shown above. Complete the table below for each solution, saying whether the given solution passes or fails the referenced test case and, if it fails, what the actual return value is. After the table explain, in general terms, what is wrong with the solution.

```python
def lawFirm (partners):
    partnerList = partners.split(",")
    firmName = ""
    for idx in range(len(partnerList)):
        lastName = partnerList[idx].split()[-1]
        partnerList[idx] = lastName
    partnerList.insert(-1, ",&")
    for p in partnerList:
        firmName += p + " \\
    return firmName.strip() + ", LLC"
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Pass/Fail</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why it fails:

```python
def lawFirm (partners):
    partnerList = partners.split(",")
    firmName = ""
    for p in partnerList[:-1]:
        firmName += p.split()[-1] + " \\
    if len(partnerList) > 1:
        lastName = partnerList[-1]
        firmName += ",& " + lastName.split()[-1]
    return firmName + ", LLC"
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Pass/Fail</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why it fails:
PROBLEM 7: *(Files, Formatted Data, Extreme Values)*

For these problems, consider the following data stored in a file in the format shown below. Each line contains
data for one animal giving the animal’s name (string), gestation period in days (int), and estimated longevity
in years (int). The information on a line is delimited by commas as shown, for example the file below shows
information for eight animals in the format used in this problem.

```
bear,180,15
cat,52,10
dog,53,10
hamster,15,2
elephant,510,30
hippopotamus,220,30
human,253,65
lion,106,10
```

Part A

Write the function `getAgeList` that returns a list of those animals whose estimated longevity is between the
values given by its two int parameters: `low` and `high`. The name of the file holding the data to be read and
processed is given by parameter `filename`.

For example, if "data.txt" is the name of the sample data file above, then the call
`getAgeList("data.txt",15,30)` should return the list `["bear","elephant","hippopotamus"]`,
the call `getAgeList("data.txt",1,8)` should return the list `["hamster"]` and the call
`getAgeList("data.txt",70,100)` should return the empty list `[]`

```python
def getAgeList (filename, low, high):
    file = open(filename)
```

```
Part B

Write the function getLongestGestation that returns a list of two values, a string, the name of the animal with the longest gestation period, and an int, representing its gestation period. The name of the file holding the data to be read and processed is given by parameter filename.

For example, if "data.txt" is the name of the sample data file above, then the call getLongestGestation("data.txt") should return the list ["elephant",510].

def getLongestGestation (filename):
    file = open(filename)