Closed book, closed notes, log out of computer! Please write neatly!

4.1 For the following, indicate whether the expression will evaluate to True or False.

<table>
<thead>
<tr>
<th>Expression</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>not True and True</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>True or value &gt; 3</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>&quot;abc&quot;[1] == &quot;b&quot;</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>(1.0 &gt;= 1.0) and (1.0 &gt; 1.0)</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

4.2. For each of the following code snippets, enter the final value for \( y \) on the line:

**a.**
```python
x = 6
if x >= 2:
    y = 2
elif x >= 4:
    y = 4
elif x >= 8:
    y = 8
```

**b.**
```python
x = 6
if x >= 2:
    y = 2
if x >= 4:
    y = 4
if x >= 8:
    y = 8
```

4.3 For each of the following while loops indicate whether the loop is guaranteed to terminate or not execute (indicate with “T”), is guaranteed to be an infinite loop (indicate with “I”) or depends (indicate with “D”).

**a.**
```python
i = 0
while i < 5:
    i = i - 1
```

**b.**
```python
i = 1
while i < 5:
    i = i * 2
```

**c.**
```python
i = 3
while i < 5:
    a = i + 1
```

**d.**
```python
from random import randint
i = 0
while i < 5:
    i = i + randint(1, 5)
```
1.1. Which of the following instructions in a recipe show why recipes can be an imperfect analogy for an algorithm in a Computer Science context? Select all that apply.

- Bake until the internal temperature is 165 °F.
- Cut into strips.
- Add 10 grams of all-purpose flour.
- Preheat the oven to 325 °F.
- Mix thoroughly.

1.2. Write (in the box) the value of x after the code below executes?

```python
x = 4
y = x
y = 3
# x = 3 * x
x = y - x
2 * x
x = 2 * x
```

1.3. Evaluate the following expressions and indicate if an error would occur, or if it is a valid Python expression, indicate what the value would be. Make the type of any values clear by showing quotes or decimal portions as relevant.

(a) \(3 \mod 2\)

(b) "3" * 2

(c) \(3 / 2 + 1\)
2.2. Write an appropriate docstring for the following function:
   ```python
   import turtle as t
   def mystery(side):
   
   for i in range(3):
       t.forward(side)
       t.left(120)
   ```

2.3. Below each snippet report the value you would assign to the variable \( n \) to so that \( \text{total} \) has the specified value. Each snippet should be considered independently. The value for \( n \) must be an integer that results in valid Python code (i.e., not cause an error).

   a) \( \text{total} \) is 6
   ```python
   total = 0
   for i in range(2):
       for j in range(n):
           total = total + 1
   ```

   b) \( \text{total} \) is 8
   ```python
   total = 0
   for i in range(2):
       total = total + 1
   for j in range(n):
       total = total + 1
   ```

2.1. Write a function named `iso_height` that has two parameters `base, side`, representing the length of the base and sides of an isosceles triangle and returns the height (as shown in the figure). Recall that the height of an isosceles triangle is \( \sqrt{\text{side}^2 - \frac{\text{base}^2}{4}} \). You do not need to include comments or docstrings.

   ```python
   >>> iso_height(2, 3)
   2.828
   ```
3.1 Enter in the box (right) what each code block would print when run via the green arrow in Thonny. Each block should be considered independently.

| l = [1, "a", 2] |
| l.remove("a") |
| print(l[1]) |

| s = "Midd" |
| print(s.replace("dd","M").lower()) |

| print("a"*2 + 'b'*3) |

3.2. Evaluate the following expressions, writing the resulting string into the boxes, one character per box (recall a space is a character), and shading in unused boxes to indicate the end of the string (the empty string would have all boxes shaded). The value of `astring` is "cs for me!".

For example: `astring`

```
c s f o r m e !
```

`astring[:2]`

```

```

`astring[0::3]`

```

```

`astring[7:9]`

```

```

3.3 For the following, indicate whether each block is valid Python if `value` is both a string and a list, only a string, only a list or valid for neither a list nor a string. You can assume `value` is already assigned and not empty.

<table>
<thead>
<tr>
<th></th>
<th>Both</th>
<th>Only string</th>
<th>Only list</th>
<th>Neither</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>len(value[2:]);</code></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>sorted(value);</code></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>value.append(&quot;a&quot;);</code></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>