 CMPSCI 119
 Fall 2019
 Monday, November 18, 2019
 Midterm #2 Solution Key
 Professor William T. Verts
25 Points – What is the value of each expression below? Answer any 25; answer more for extra credit. Answer “Error” if an expression cannot be computed for any reason. Scoring:

> +1: Completely correct answers
> +½: Incorrect data types, lists without square brackets, strings without quotes, etc.
> 0: Blank answers
> −½: Incorrect answers (better to leave it blank than to guess)

Ruby = 13
Weiss = 4.8
Blake = [3, 7, "Cat", 7.5, 4]
Yang = {"Age":17, "Color":"Yellow", "Semblance":"Burn"}
RWBY = ("Red", "White", "Black", "Yellow")

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>6.5</td>
<td>float</td>
<td>Ruby / 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>6</td>
<td>int</td>
<td>Ruby // 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>5</td>
<td>int</td>
<td>round(Weiss)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>4</td>
<td>int</td>
<td>int(Weiss)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>4.8</td>
<td>float</td>
<td>float(Weiss)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>13.0</td>
<td>float</td>
<td>float(Ruby)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>&quot;13&quot;</td>
<td>string</td>
<td>str(Ruby)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>5</td>
<td>int</td>
<td>len(Blake)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>4</td>
<td>int</td>
<td>len(RWBY)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>ERROR</td>
<td></td>
<td>len(Ruby)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>3</td>
<td>int</td>
<td>len(Blake[2])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>6</td>
<td>int</td>
<td>len(Yang[&quot;Color&quot;])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>ERROR</td>
<td></td>
<td>len(Yang[Color])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>1</td>
<td>int</td>
<td>len(str(Blake[-1]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>True</td>
<td>bool</td>
<td>Yang[&quot;Color&quot;] == RWBY[3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>ERROR</td>
<td></td>
<td>Yang[&quot;Frog&quot;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td>[7,&quot;Cat&quot;,7.5,4]</td>
<td>list</td>
<td>Blake[1:]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>[13,14,15,16]</td>
<td>list</td>
<td>list(range(Ruby, Yang[&quot;Age&quot;]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>[3,5,7,9,11]</td>
<td>list</td>
<td>list(range(Blake[0], Ruby, 2))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>[0,1,2,3]</td>
<td>list</td>
<td>list(range(Blake[-1]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td>[0,4,8]</td>
<td>list</td>
<td>list(range(0, 10, Blake[4]))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>[]</td>
<td>list</td>
<td>list(range(Ruby, 5))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.</td>
<td>[0,0,0]</td>
<td>list</td>
<td>[0 for N in range(Ruby // 4)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.</td>
<td>[13,14,15,16,17]</td>
<td>list</td>
<td>[N+Ruby for N in range(5)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.</td>
<td>ERROR</td>
<td></td>
<td>[Frog+Ruby for N in range(5)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.</td>
<td>[0,0,0,0]</td>
<td>list</td>
<td>[0] * 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.</td>
<td>[13,4.8,13,4.8,13,4.8]</td>
<td>list</td>
<td>[Ruby, Weiss] * 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<2> 20 Points – (2 points each answer) When `Main()` is called there will be exactly ten lines of output printed. What are they?

```python
def Chalk(A, B, C=5):
    Blackboard = A + B
    print(Blackboard)
    return Blackboard + C

def Erasers(X, Y, Z):
    Q = Y + Chalk(X, Z)
    print(Q)
    return

def Main():
    print(Chalk(2, -5, 1))
    Erasers(5, 3, -4)
    print(Chalk(3, 1))
    Erasers(-4, 2, 1)
    print(Chalk(1, 1, 1))
    return
```

# Line 1 ___-3___
# Line 2 ___-2___
# Line 3 ___1___
# Line 4 ___9___
# Line 5 ___4___
# Line 6 ___9___
# Line 7 ___-3___
# Line 8 ___4___
# Line 9 ___2___
# Line 10 ___3___
15 Points – Complete the `Process` function below to flip a coin \( N \) times. Use the `random.random()` function to generate each coin flip; print out `HEADS` if the random value is less than 0.5 but print out `TAILS` otherwise. The command `import random` is already at the top of the program. Remove 1 point per error in each section.

- 5 Points max for a properly constructed loop of some kind
- 5 Points max for correct use of `random.random()`
- 5 Points max for testing and printing

```python
def Process (N):
    for I in range(N):
        Value = random.random()
        if Value < 0.5:
            print ("HEADS")
        else:
            print ("TAILS")
    return

-or-

def Process (N):
    I = 0
    while I < N:
        Value = random.random()
        if Value < 0.5:
            print ("HEADS")
        else:
            print ("TAILS")
        I = I + 1
    return

-or-

def Process (N):
    for I in range(N):
        if random.random() < 0.5: print ("HEADS")
        else: print ("TAILS")
    return

-or-

def Process (N):
    I = 0
    while I < N:
        if random.random() < 0.5: print ("HEADS")
        else: print ("TAILS")
        I = I + 1
    return
```
15 Points – Write a counter-loop code fragment (not a complete function) using Index as the counter variable, which starts at 37, goes up to but does not include 914, and counts by 7s. The payload of the loop is to call the Strange function with Index as its (only) parameter.

10 Points max for a properly constructed loop of some kind
5 Points max for calling Strange(Index)
Accept a for-loop solution, even though the first version is what is expected
Remove 1 point per error in each section

```python
Index = 37
while Index < 914:
    Strange(Index)
    Index = Index + 7
-or-
for Index in range(37, 914, 7):
    Strange(Index)
-or-
for Index in range(37, 914, 7): Strange(Index)
```

10 Points – The following function attempts to write the square roots of all integers from 0 up to and including N, one per line, to the text file indicated by file name F. However, there are both syntax errors and logic errors in the code. Locate and fix all the errors.

I count 8 unique errors. Start with 10 points and remove 1 point for each error not found, and remove 1 point for each correct item miss-identified as an error, but do not go below zero.

```python
import math
# Math \rightarrow math

def WriteSQRT (F,N):
    Handle = open(F, "wb")
    # "wb" \rightarrow "w"
    for I in range(N+1):
        # N \rightarrow N+1
        handle.write(str(math.sqrt(I)) + "\n")
            # Missing )
            # Missing +"\n"
            # handle \rightarrow Handle
Handle.close()
    return
```
10 Points – I have a variable $L$ containing a list of floats that represent an audio sound (to be saved with the `WriteWAV` function in the book). $L$ contains a bunch of sound samples (that is, it isn’t empty). The variable `SamplesPerSecond` is already defined and contains the number of sound samples needed for each second the sound will play. Write a code fragment (not a function!) to add five seconds of silence to the end of $L$ (silence is where the sample value is zero).

```python
for I in range(SamplesPerSecond * 5):
    L = L + [0.0]
```

-or-

```python
for I in range(SamplesPerSecond * 5):
    L.append(0.0)
```

-or-

```python
TotalSamples = SamplesPerSecond * 5
for I in range(TotalSamples):
    L = L + [0.0]
```

-or-

```python
TotalSamples = SamplesPerSecond * 5
for I in range(TotalSamples):
    L.append(0.0)
```

-or-

```python
TotalSamples = SamplesPerSecond * 5
I = 0
while (I < TotalSamples):
    L.append(0.0)
    I = I + 1
```

-or-

```python
TotalSamples = SamplesPerSecond * 5
I = 0
while (I < TotalSamples):
    L = L + [0.0]
    I = I + 1
```

Scoring:

- 3 Points max for computing the number of new samples needed
- 4 points max for an appropriate loop, either a for-loop or while-loop
- 3 points max for adding the 0.0 (int 0 is allowed) to the end of the list L
- Remove 1 point per error in each section (do not go below 0 in each section)

In talking with students after the exam, I found that a number of them had used the `makeEmptySound` function referred to on page 430 of the Companion. The description on that page clearly says that this function is specific to the Python 2 JES environment, which we are not using. The Python 2/3 code on page 431 is far more relevant. Give those students 4 points max for this approach, but again remove 1 point per error (syntax, mostly). Do not go below zero.
There are two major ways we can represent polynomials in Python. One is to create a list where each item in the list is a two-element list containing the coefficient and the exponent. For example, the polynomial $3x^{10} - 6x^4 + 2$ would be encoded in Python as $[[3,10], [-6,4], [2,0]]$. The other way is to create a list where each entry is the coefficient and the index is the exponent. The same polynomial $3x^{10} - 6x^4 + 2$ would be encoded as $[2, 0, 0, 0, -6, 0, 0, 0, 0, 3]$. What are the advantages and disadvantages of each approach?

The first way (a list of `[coefficient,exponent]` lists) is very good for large, sparse polynomials, such as $3x^{1000000} - 6x^{4000} + 2$ as this approach stores information for only the needed non-zero terms. The Python code would be somewhat complicated.

However, the second way (a simple list of coefficients) is far simpler for the large majority of expected cases, both to represent the polynomials and to process them in Python code. It is less efficient for large, sparse polynomials, as a lot of zero coefficients would be present in the list.

Scoring:

5 points for correctly discussing both the efficiency of the representation and the complexity of the Python code.

3 points for correctly describing one or the other (efficiency or complexity), but either omitting the other discussion or getting it wrong.

1 point for anything marginally reasonable.

0 points for any answer that is way off base.