**Practice Questions for CSCI-UA.0380 Final**

1. Read the code in the right column. Answer questions about it in the left column.

```python
class Foo(object):
    def __init__(self, n):
        self.x = n
    def bar(self):
        print('qux')
    def __str__(self):
        return self.x * 'foo'

class Corge(Foo):
    def __init__(self, n):
        super().__init__(n)
    def bar(self):
        print('grault')
c = Corge(3)
c.bar()
print(c)

def what_is_this(a, b):
    if b == 0:
        return 1
    else:
        return a * what_is_this(a, b - 1)
print(what_is_this(5, 0))
print(what_is_this(4, 3))

def create_initial_clusters(k):
    clusters = []
    for i in range(k):
        clusters.append([])
    return clusters
clusters = create_initial_clusters(3)
print(clusters)
for result in map(len, clusters):
    print(result)
```

### Questions:

What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.

What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.

What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.

### Code Modifications:

Change the code above so that the body of `create_initial_clusters` is a list comprehension instead of a regular for loop. Write your list comprehension below:

```python
def mystery(n):
    nums = [0, 1]
    for i in range(n):
        val = nums[0]
        del(nums[0])
        nums.append(val + nums[0])
        yield val
mysterious = mystery(7)
```

What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.
2. **Part 1** – K-means clustering: write one or two sentence answers to the following questions:

a) How are the initial centroids generated before any data points are put into clusters?

b) How are the subsequent centroids determined (once data points are distributed into clusters)?

c) Recalculating centroids and clustering data points based on new centroids is a process that is repeated applied. When should the repetition of the process above stop?

**Part 2** – Write a Euclidean distance function where that can handle an arbitrary number of data points. For example: `distance((5, 0, 1), (2, 12, 5)) # --> 12`. Euclidean distance is found by taking summing the square of the difference of each feature between each data point, and taking the square root of the resulting sum. Here’s an example where there are only two feature, x and y:

\[
d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
\]

Your function must work for an arbitrary number of features (for example, x, y and z).

**Part 3** – Audio: write one or two sentence answers to the following questions:

a) Define frequency and sampling rate.

b) Why do the numbers that our pyaudio program create have to be within -1 and 1?

c) Why can sound be described as a signal?
3. **Part 1 – Classes and Objects: True or False**

   a) __________ super() represents the parent class of the class that super is being used in

   b) __________ __init__ is automatically called when you create a new object by calling a function that's named the same name as the class of the object you're creating

   c) __________ when a method is called on an object, the first argument that is passed in to the method is the object that the method was called on

   d) __________ when creating a class, you have to specify a method called __str__

**Part 2 – HTTP:** write one or two sentence answers to the following questions:

   a) If you type in http://localhost:5000/hello/there in your browser's URL bar, what is the 1st line of the http request that is made to the web server / your web application?

   b) Name two HTTP methods.

   c) Name three tools or applications that you can use to create an HTTP request.

**Part 3 – Sockets: True or False**

   a) __________ when communicating using Python's socket module... data received from a client arrives in a bytes object, not a string

   b) __________ a socket represents a connection between a client and a server

   c) __________ you cannot create your own protocol when communicating via sockets; you must use a defined protocol, such as HTTP, FTP, SMTP, etc.

**Part 4 – Flask:** write one or two sentence answers to the following questions:

   a) What is placed in the static directory? How are those resources accessed via url?

   b) How do you tell your application to associate a function with a specific URL?

   c) Why use templating rather than serving html files directly (that is, without render_template)?
4. Read the program in the left column. Write the output of the program in the right column.

<table>
<thead>
<tr>
<th>Program</th>
<th>Output</th>
</tr>
</thead>
</table>
| def zippy(li_1, li_2, li_3):
  s = ''
  for a, b, c in zip(li_1, li_2, li_3):
    s += '{} {}'.format(a, b)
    if c == 'loudly):
      s += '!!!!!!'
    s += '\n'
  return s

names = ['Alice', 'Bob']
actions = ['ambled', 'bellowed']
adverbs = ['anxiously', 'loudly']
print(zippy(names, actions, adverbs))

<table>
<thead>
<tr>
<th>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What response will the server output to the client if the client is netcat using the following commands.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>a) nc localhost 5000 what</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>b) nc localhost 5000 hello there how are you</td>
</tr>
</tbody>
</table>

| def letter_me():
  num = 65
  while True:
    if num > 90:
      num = 65
    val = num
    num += 1
    yield chr(val)

lettered = letter_me()

for i in range(1, 6):
  if i % 2 == 1:
    print(next(lettered), end='')
  else:
    next(lettered)

| def half(s):
  return s[:len(s) // 2]

def double(s):
  return s * 2
d = {'a': double, 'b': half}
def wordy(letter, s):
 funcy = d[letter]
  return funcy(s)

print(wordy('b', 'cats'))
print(wordy('c', 'cats'))
print(wordy('a', 'cats'))

<table>
<thead>
<tr>
<th>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What's the output of the program on the left? If the output includes an error, show all output before the error and the type of error that occurs.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>What response will the server output to the client if the client is netcat using the following commands.</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>a) nc localhost 5000 what</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>b) nc localhost 5000 hello there how are you</td>
</tr>
</tbody>
</table>

import socket
# socket setup omitted to save space
s.listen(queue)

while True:
  client, address = s.accept()
  data = client.recv(4096)
  if data:
    req = data.decode('utf-8')
    parts = req.split(' ')
    res = parts[0].strip() + 'x'
    res + parts[-1].strip()
    client.send(bytes(res, 'utf-8'))
  client.close()
5. Fill in the bodies (the two areas marked // TODO: FINISH FUNCTION BODY) of the following two functions that draw squares – one with PIL, the other with turtle. The programs below `draw_square` functions to create a square that's 200 pixels. The function headers and example usage are below. Note that in the PIL version, the color can be specified, and the example has a white outline of a square on a black background.

<table>
<thead>
<tr>
<th>Drawing a square with PIL</th>
<th>Drawing a square with turtle</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def draw_square(img, left_x, top_y, size, color):</code></td>
<td><code>def draw_square(t, left_x, top_y, size):</code></td>
</tr>
<tr>
<td>// TODO: FINISH FUNCTION BODY</td>
<td>// TODO: FINISH FUNCTION BODY</td>
</tr>
<tr>
<td>from PIL import Image</td>
<td>import turtle</td>
</tr>
<tr>
<td>img = Image.new('RGB', (400, 400))</td>
<td>my_turtle = turtle.Turtle()</td>
</tr>
<tr>
<td><code>draw_square(img, 100, 100, 200, (255, 255, 255))</code></td>
<td>wn = turtle.Screen()</td>
</tr>
<tr>
<td>img.show()</td>
<td><code>draw_square(my_turtle, -100, -100, 200)</code></td>
</tr>
<tr>
<td></td>
<td>wn.mainloop()</td>
</tr>
</tbody>
</table>
6. Fill in the missing parts (marked // TODO: FILL IN THIS CODE) of the flask application below. The flask application allows you to enter comma separated numbers and an operation (sum to sum all numbers, product to multiply all numbers, and max to get the largest number) through a form. When the form is submitted, the page that gets rendered should show the original operation and numbers as well as the result. The form is also displayed underneath the results. If the operation is unrecognized, then the result should be a message that says: “Error, operation not supported”.

app.py
from flask import Flask
from flask import request
from flask import render_template
app = Flask(__name__)
app.debug = True
// TODO: FILL IN THIS CODE
app.run()

calculate.html
{% if res %}
<div>numbers: {{nums}}</div>
<div>operation: {{op}}</div>
<div>result: {{res}}</div>
{% endif %}
<form method="POST" action="">
<div>Numbers: <input type="text" name="nums"></div>
<div>Operation: <input type="text" name="op"></div>
<input type="submit" value="Run Calculation">
</form>

<table>
<thead>
<tr>
<th>Going to /calculate</th>
<th>Filling out form</th>
<th>After form Submission</th>
<th>If operation doesn't exist</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Going to /calculate" /></td>
<td><img src="image2" alt="Filling out form" /></td>
<td><img src="image3" alt="After form Submission" /></td>
<td><img src="image4" alt="If operation doesn't exist" /></td>
</tr>
<tr>
<td>Numbers:</td>
<td>Operation:</td>
<td>numbers: 3,4,5</td>
<td>numbers: 3,4,5</td>
</tr>
<tr>
<td>Run Calculation</td>
<td>Run Calculation</td>
<td>operation: product</td>
<td>operation: fooobarbaz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>result: 60</td>
<td>result: Error, not a valid operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numbers:</td>
<td>Numbers:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operation:</td>
<td>Operation:</td>
</tr>
</tbody>
</table>
7. Create a class called `FixedWidthFile`. This class can be used to access data within a file that has one row of data per line, with each column in the row being a fixed width of characters. The first line of a fixed width file will always contain column names, and each subsequent row will contain actual data. See the example fixed width file in the first column of the table below ("Contents of names.txt").

Your class will allow you to create a new `FixedWidthFile` object given a file name. Once you have an object, you can call the get method on that object to retrieve a piece of data at a given row number (0 for the 1st, 1 for the 2nd, etc.) and a column name (again, in the example below, First would be a column name). Lastly, when `str` is print is used with the object, its string representation is just the file's headers separated by commas.

<table>
<thead>
<tr>
<th>Contents of names.txt</th>
<th>Code Using FixedWidthFile Class</th>
<th>Output</th>
</tr>
</thead>
</table>
| FIRST                | f = FixedWidthFile('names.txt', 8)
Alice               | print(f)                        | FIRST, LAST, MIDDLE |
Anderson             | print(f.get(0, 'LAST'))         | Anderson         |
Benjamin             | print(f.get(1, 'FIRST'))        | Benjamin         |
Berger               |                                 |                 |
8. Write a decorator called `constrain`. It will:

(a) take all incoming arguments of a function that it decorates and constrain the values of those arguments to values between 1 and 10, inclusive
(b) if an argument is less than or greater than this range, then its value will be changed to 1 or 10 respectively
(c) it will then call the old function with the constrained arguments. The constructor must be able to deal with functions that take arbitrary number of arguments
(d) see the example and hints below:

<table>
<thead>
<tr>
<th>Example</th>
<th>Hints</th>
</tr>
</thead>
<tbody>
<tr>
<td># in this example, the function, # product is decorated with # @constrain @constrain def product(*args): p = 1 for n in args: p *= n return p print(product(0, 2)) # --&gt; 2 print(product(2, 99, 3)) # --&gt; 60 # note that the product in the 1st # print is 2 and not 0 because the # decorator changes the 1st argument # argument to 1 (a similar change # occurs in the 2nd print as well)</td>
<td># remember that all arguments passed # in to a function as *args shows up # as elements in a tuple def show_args(*args): print(args) show_args(&quot;foo&quot;, &quot;bar&quot;, &quot;baz&quot;) # prints out the tuple: # (&quot;foo&quot;, &quot;bar&quot;, &quot;baz&quot;) # you can unpack a list or a tuple # into separate arguments to a # function by prefixing with a * nums = [0, 10, 3] print(list(range(*nums)))</td>
</tr>
</tbody>
</table>
## Reference Material / Scrap Paper

**Reference for Flask, socket, etc. to be added**

### Built-in String Methods
- `abs`  
- `bool`  
- `count`  
- `dict`  
- `enumerate`  
- `format`  
- `find`  
- `isalpha`  
- `isdecimal`  
- `isdigit`  
- `islower`  
- `isspace`  
- `isnumeric`  
- `istitle`  
- `isupper`  
- `join`  
- `lower`  
- `replace`  
- `split`  
- `startswith`  
- `strip`  
- `title`  
- `upper`  

### Turtle and Screen Methods
- `Turtle Object`  
- `Turtle Object`  
- `Screen Object`  

### Math Module
- `acos`  
- `acosh`  
- `asin`  
- `asinh`  
- `atan`  
- `atan2`  
- `atanh`  
- `ceil`  
- `cos`  
- `cosh`  
- `degrees`  
- `floor`  
- `log`  
- `log10`  
- `log2`  
- `pi`  
- `pow`  
- `radians`  
- `sin`  
- `sinh`  
- `sqrt`  
- `tan`  
- `tanh`  

### PIL Methods
- `Image Module`  
- `Image.new(...)`  
- `Image.open(...)`  
- `open`  

### List Methods
- `append`  
- `count`  
- `extend`  
- `index`  
- `insert`  
- `pop`  
- `remove`  
- `reverse`  
- `sort`  

### File Object Methods
- `read`  
- `readline`  
- `readlines`  
- `write`  

### Dictionary Methods
- `get`  
- `items`  
- `keys`  
- `pop`  
- `popitem`  
- `update`  
- `values`  

### Random Module Functions
- `choice`  
- `randint`  
- `sample`  
- `shuffle`  

### ASCII Chart

<table>
<thead>
<tr>
<th>Char Dec</th>
<th>Char Dec</th>
<th>Char Dec</th>
<th>Char Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>0</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(eng)</td>
<td>5</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(cr)</td>
<td>12</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(fs)</td>
<td>14</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(gs)</td>
<td>15</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(rs)</td>
<td>16</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(us)</td>
<td>17</td>
<td>(sp)</td>
<td>32</td>
</tr>
<tr>
<td>(del)</td>
<td>18</td>
<td>(sp)</td>
<td>32</td>
</tr>
</tbody>
</table>

---

**Net ID:** ______________________  **Name:** ______________________