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)
```

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.

```
grault
foofoofoo
```

```python
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))
```

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.

```
1
64
```

```python
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)
```

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.

```
[[], [], []]
0
0
0
```

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
[[] for i in range(k)]
```

```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(6)
```

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.

```
0
1
1
2
3
5
```

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?

The initial centroids are created by choosing random data points from the data set, without duplications

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

Go through all of the data points in each cluster, and create a new centroid by taking the average of each feature of each data point in the cluster.

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

Repeat a set number of times, repeat until the centroids stabilize or repeat until the centroids reach a point where they flip flop between two values constantly.

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).

```
def distance(p1, p2):
    sum_all = 0
    for i, value in enumerate(p1):
        diff_squared = (value - p2[i]) ** 2
        sum_all += diff_squared
    return math.sqrt(sum_all)

print(distance((5, 0), (2, 12)))

#or return math.sqrt(sum([(a - b) ** 2 for a, b in zip(p1, p2)]))
```

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

a) Define frequency and sampling rate.

Frequency is the number of cycles per second. Sample rate is how many samples per second.

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

-1 and 1 represent the position of a speaker's membrane.

c) Why can sound be described as a signal?

A signal is something that varies over time or space... or both. An audio signal is the variation of air pressure over time.

3. **Part 1 –** Classes and Objects: True or False

a)_**False**__ super() represents the parent class of the class that super is being used in

b)_**True**___ __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)_**True**___ 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)_**False**__ 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 you browser's URL bar, what is the 1st line of the http request that is made to the web server / your web application?

GET /hello/there HTTP/1.1

b) Name two HTTP methods.

GET and POST

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

Browser, curl, netcat, requests library

Part 3 – Sockets: True or False

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

b)_**False**_ a socket represents a connection between a client and a server

c)_**False**_ you cannot create your own protocol when communicating via sockets; you must use a 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?

Any "static" files, that is files that don't have any dynamic data in them, such as CSS, images, and other "static" resources. They're accessed by /static/resource.name

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

Use the app.route(...) decorator before the function that you want to handle that url.

c) Why use templating rather than serving html files directly (that is, without render_template)?

There may be dynamic elements on your page (that is, data filled by variables)

4. Read the program in the left column. Write the output of the program in the right column.

```
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))
```

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.

```
Alice ambled
Bob bellowed!!!!!
```

```
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)
```

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.

```
ACE
```

```
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'))
```

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.

```
ca
KeyError
```

```
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()
```

What response will the server output to the client if the client is netcat using the following commands.

a) nc localhost 5000
what

```
whatxwhat
```

b) nc localhost 5000
hello there how are you

```
helloxyou
```

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.

Drawing a square with PIL	**Drawing a square with turtle**

`def draw_square(img, left_x, top_y, size, color):`
    `// TODO: FINISH FUNCTION BODY`
`from PIL import Image`
`img = Image.new('RGB', (400, 400))`
`draw_square(img, 100, 100, 200, (255, 255, 255))`
`img.show()`
	`def draw_square(t, left_x, top_y, size):`
	    `// TODO: FINISH FUNCTION BODY`
	`import turtle`
	`my_turtle = turtle.Turtle()`
	`wn = turtle.Screen()`
	`draw_square(my_turtle, -100, -100, 200)`
	`wn.mainloop()`

```
def draw_square(img, left_x, top_y, size, color):
    pixels = img.load()
    for y in range(top_y, top_y + size):
        pixels[left_x, y] = color
        pixels[left_x + size - 1, y] = color
    for x in range(left_x, left_x + size):
        pixels[x, top_y] = color
        pixels[x, top_y + size - 1] = color
```

```
def draw_square(t, left_x, top_y, size):
    t.up()
    t.goto(left_x, top_y)
    t.setheading(0)
    t.down()
    for i in range(4):
        t.forward(size)
        t.right(90)
```

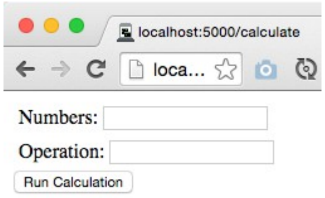
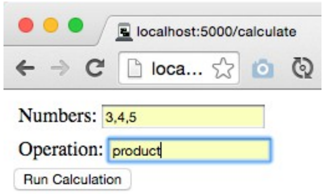
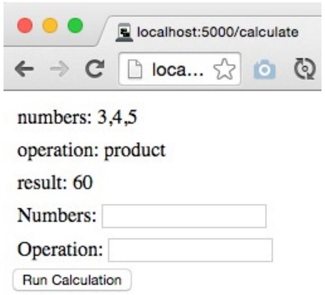
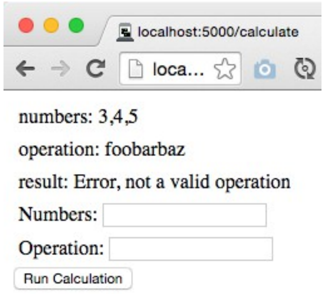
6. Fill in the missing parts (marked `// TODO: FILL IN THIS CODE`) of the flask application below. The flask application allows you to enter comma separated numbers and an operation (`sum` to sum all numbers, `product` to multiply all numbers, and `max` to get the largest number) through a form. When 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>
```

Going to /calculate	Filling out form	After form Submission	If operation doesn't exist
 localhost:5000/calculate loca... Numbers: Operation: Run Calculation	 Numbers: 3,4,5 Operation: product 	 localhost:5000/calculate loca... numbers: 3,4,5 operation: product result: 60 Numbers: Operation: Run Calculation	 localhost:5000/calculate loca... numbers: 3,4,5 operation: foobarbaz result: Error, not a valid operation Numbers: Operation: Run Calculation

```
def product(numbers):
    result = 1
    for num in numbers:
        result *= num
    return result

def error(numbers):
    return 'Error, not a valid operation'

funcs = {'product':product, 'sum':sum, 'max':max}

@app.route('/calculate', methods=['GET', 'POST'])
def calculate():
    if request.method == 'GET':
        return render_template('calculate_form.html')
    if request.method == 'POST':
        nums = request.form['nums'].strip()
        op = request.form['op'].strip()
        f = funcs.get(op, error)
        res = f(map(int, nums.split(',')))
        return render_template('calculate_form.html', op=op, nums=nums,
res=res)
```

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.

Contents of `names.txt`	**Code Using `FixedWidthFile` Class**	**Output**

`FIRST   LAST    MIDDLE`
`Alice   AndersonA`
`BenjaminBerger  B`
	`f = FixedWidthFile('names.txt', 8)`
	`print(f)`
	`print(f.get(0, 'LAST'))`
	`print(f.get(1, 'FIRST'))`
		`FIRST,LAST,MIDDLE`
		`Anderson`
		`Benjamin`

```
class FixedWidthFile:

    def __init__(self, fn, col_width):
        self.filename = fn
        self.col_width = col_width
        self.col_names = ''
        self.data = []
        self._load()

    def _load(self):
        with open(self.filename, 'r') as f:
            first_line = f.readline()
            self.col_names = [first_line[i: i + self.col_width].strip() for i
in range(0, len(first_line), self.col_width)]
           for line in f:
                d = {}
                for i, col_name in enumerate(self.col_names):
                    start = i * self.col_width
                    end = start + self.col_width
                    d[col_name] = line[start:end].strip()
                self.data.append(d)

    def get(self, row_num, col_name):
        return self.data[row_num][col_name]

    def __str__(self):
        return "file with columns: {}".format(','.join(self.col_names))
```

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>
<tr><th>Example</th><th>Hints</th></tr>
<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))     # --> 2
print(product(2, 99, 3)) # --> 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("foo", "bar", "baz")

# prints out the tuple:
# ("foo", "bar", "baz")


# 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>
</table>

```
def constrain(old_f):
    def new_f(*args, **kwargs):
        for i in range(len(args)):
            args = list(args)
            if args[i] > 10:
                args[i] = 10
            elif args[i] < 1:
                args[i] = 1
        return old_f(*args, **kwargs)
    return new_f
```

Net ID:______________________________________ **Name:**____

Reference Material / Scrap Paper

Reference for Flask, socket, etc. to be added

Built-in	String Methods	Turtle and Screen	Math Module	PIL	List Methods	File Object Methods

abs
bool
chr
dict
enumerate
filter
float
format
input
int
len
list
map
max
min
open
ord
pow
print
range
round
set
sorted
str
sum
	capitalize
	count
	endswith
	find
	format
	index
	isalnum
	isalpha
	isdecimal
	isdigit
	islower
	isnumeric
	isprintable
	isspace
	istitle
	isupper
	join
	lower
	replace
	split
	startswith
	strip
	title
	upper
		Turtle Object
		back(distance)
		begin_fill()
		circle(radius)
		clear()
		color(color)
		color(colorstring)
		down()
		end_fill()
		forward(distance)
		goto(x, y)
		hideturtle()
		left(angle)
		pensize(size)
		right(angle)
		setheading(angle)
		up()
		Screen Object
		bgcolor(colorstring)
		listen()
		onkeypress(func, key
		ontimer(func, time_ms)
		setup(width, height)
		tracer(0)
		update()
			acos
			acosh
			asin
			asinh
			atan
			atan2
			atanh
			ceil
			cos
			cosh
			degrees
			floor
			log
			log10
			log2
			pi
			pow
			radians
			sin
			sinh
			sqrt
			tan
			tanh
				Image Module
				Image.new(...)
				Image.open(...)
				Image Object
				img.size
				img.load()
				PixelAccess Object
				(like a dict)
				get using []'s
				set using []'s
					append
					count
					extend
					index
					insert
					pop
					remove
					reverse
					sort
						read
						readline
						readlines
						write

					Dictionary Methods	**Random Module Functions**
					get items keys pop popitem update values	choice randint sample shuffle

ASCII Chart

```
Char Dec  | Char Dec | Char Dec | Char Dec
 -------------------------------------------
 (nul)  0  | (sp) 32  | @    64  | `     96
 (soh)  1  | !    33  | A    65  | a     97
 (stx)  2  | "    34  | B    66  | b     98
 (etx)  3  | #    35  | C    67  | c     99
 (eot)  4  | $    36  | D    68  | d    100
 (enq)  5  | %    37  | E    69  | e    101
 (ack)  6  | &    38  | F    70  | f    102
 (bel)  7  | '    39  | G    71  | g    103
 (bs)   8  | (    40  | H    72  | h    104
 (ht)   9  | )    41  | I    73  | i    105
 (nl)  10  | *    42  | J    74  | j    106
 (vt)  11  | +    43  | K    75  | k    107
 (np)  12  | ,    44  | L    76  | l    108
 (cr)  13  | -    45  | M    77  | m    109
 (so)  14  | .    46  | N    78  | n    110
 (si)  15  | /    47  | O    79  | o    111
 (dle) 16  | 0    48  | P    80  | p    112
 (dc1) 17  | 1    49  | Q    81  | q    113
 (dc2) 18  | 2    50  | R    82  | r    114
 (dc3) 19  | 3    51  | S    83  | s    115
 (dc4) 20  | 4    52  | T    84  | t    116
 (nak) 21  | 5    53  | U    85  | u    117
 (syn) 22  | 6    54  | V    86  | v    118
 (etb) 23  | 7    55  | W    87  | w    119
 (can) 24  | 8    56  | X    88  | x    120
 (em)  25  | 9    57  | Y    89  | y    121
 (sub) 26  | :    58  | Z    90  | z    122
 (esc) 27  | ;    59  | [    91  | {    123
 (fs)  28  | <    60  | \    92  | |    124
 (gs)  29  | =    61  | ]    93  | }    125
 (rs)  30  | >    62  | ^    94  | ~    126
 (us)  31  | ?    63  | _    95  | (del)127
```