Name:_____ **Date:**____/____/_____

CSCSCI-UA.0002 – Midterm #2 – Practice Questions

1. If the following code is executed ...

```
a = [4, 5, 6]
b = [3, 1, 2]
a.extend([20, 15])
c = b.sort()
d = [1,2]
a.append(b.pop())
d.append(b)
```

What are the values of a, b, c, and d **after** running the code? (2 points)

(a)____[4, 5, 6, 20, 15, 3]_____ (b)____[1, 2]_____

(c)____None_____ (d)_____[1, 2, [1, 2]]_____

2. What is a **list**? Name two **operations** or **constructs** that lists and strings share. (2 points)

A list is an ordered sequence of values (any value). Lists and strings both support +, *, [], [:]....

3. The following program **draws a dashed line** horizontally across the window. There are five dashes. Each **dash is 20 pixels** long and the **gaps between each dash is 10 pixels** long. **Complete the missing portions** of the code below. (3 points)

```
import ___turtle_______

raphaelle = __turtle.Turtle()_________
wn = turtle.Screen()

for i in ____range(5)____:
    raphaelle.forward(20)

    raphaelle.__up()________

    raphaelle.forward(__10_____)

    raphaelle.___down()_________
wn.mainloop()
```

4. Name 3 **list methods** and 3 **string methods**: (3 points)

List Methods	String Methods
append(object) extend(object) insert() sort() pop() index() count() remove(object) etc ...	upper() lower() strip() isupper() islower() isdigit() isalpha() find(char) etc ...

5. What's the difference between a **method** and a **function**? (1 point)

A method must be called on an object, a function does not.

6. Cross out all of the statements that are false or evaluate to False. (3 points)

(a) ~~`7 not in ['hi', None, 7, 3]`~~

(b) `['a', 6, 4, 5] > ['a', -2, 4, 8]`

(c) ~~strings are an unordered sequence of characters~~

(e) strings are immutable

(f) ~~the last index in a list is the length of that list~~

(g) ~~`'twelve'.isdigit()`~~

7. Write the output of each line of code. If there's an error, write Error. (3 points)

```
vegetable = "turnip",
i = len(vegetable) - 1
```

a) `print(vegetable[2] + vegetable[-2] + vegetable[i])` (a)_____rip_____

b) `print(vegetable[0])` (b)_____t_____

c) `print(vegetable[i + 1])` (c)_____Error___

d) `print("vegetable"[4:100])` (d)_____table___

e) `print(vegetable[:3])` (e)_____tur_____

f) `print(vegetable[0:(i-1)])` (f)_____turn____

8. There's an error in both code samples below. Circle the line where the error occurs. To the right of each code sample, explain why there's an error: (3 points)

a)
```
s = 'hello'
s[0] = 'C'
s = s.upper()
print(s)
```
strings are immutable (TypeError)

b)
```
c = '1'
numbers = [1, 3, 2] + [4, 5, 6]
numbers[5] = 9
numbers[c] = 12
```
indexes are integers (TypeError)

c)
```
a = say_hello('yo yo meow')
def say_hello(name):
    print('hello %s' % (name))
```
function doesn't exist yet (NameError)

9. Create a function called it **`unique_and_filter`**... (4 points)

a) it will expect a **list of *only* strings** as an input
b) it will filter that list by:
   - removing duplicates
   - ignoring any string that's three letters or less
   - ignoring any string that only consists of numbers (is numeric)
c) an empty list returns an empty list
d) write two assertions to test your code
e) Sample output:

```
>>> unique_strings = unique_and_filter(['cat', '23', 'four', 'four', 'hello', 'dog'])
>>> print(unique_strings)
['four', 'hello']
```

```
def unique_and_filter(words):
    filtered_words = []
    for w in words:
        if w not in filtered_words and len(w) > 3 and not w.isdigit():
            filtered_words.append(w)
    return filtered_words

assert ['four', 'hello'] == unique_and_filter(['cat', '23', 'four', 'four', 'hello', 'dog'])
assert [] == unique_and_filter([])
```

10. ~~Name these following methods:~~ these methods will not be on the exam

~~(1) creates a string from a list: ____join(list)_____ (2) creates a list from a string _____split(string)_____~~

11. Using the code in the 1st column, answer the questions in the second and third columns. If the question asks for output, **error** is always a possible answer. (3 points)

Code	**Question #1**	**Question #2**

`def say_cheese(n):`
`  s = n * 'cheese'`
`  print(s)`
`talk = say_cheese(3)`
	What is the **output** of this program?
	`cheesecheesecheese`
		What is the **value** of the variable called **talk**?
		None

`exclamation = 'boo'`
`whisper = 'shhh'`
`def do_something():`
`  exclamation = 'bye!'`
`  print(whisper)`
`do_something()`
`print(exclamation)`
	What is the **first** line of **output** for this program?
	
	shhh
		What is the **second** line of **output** for this program?
		
		boo

`def join_three(a, b, c):`
`  return '%s, %s, %s' % (a, b, c)`
`c, b, a = 3, 2, 1`
`res = join_three(c, b, a)`
`print(res)`
	What is the **output** of this program?
	
	3, 2, 1
		What data **type** is returned from join_three?
		
		str

12. Implement the following function (note that there are already built-in constructs, functions and methods in Python that provide similar functionality, but we'll be writing our own):

a) Create a function called **is_in_list** (½ point)
b) The function should take **two arguments**, an **integer** named **n** and a **list** named **numbers** (½ point)
c) The function should **return True or False** depending on whether or not the number is in the list (2 points)
d) Ignore the case where the function either receives non-integer values for the first argument or a value that is not a list of integers for the second argument.
e) Create two assertions to test your function.
f) Example Output:

```
>>> print(is_in_list(1, [1, 2, 3]))
True
>>> print(is_in_list(4, [1, 2, 3]))
False
>>> print(is_in_list(4, []))
False
```

```
def is_in_list(v, items):
    in_list = False
    for item in items:
        if v == item:
            in_list = True
            break
    return in_list

assert True == is_in_list(1, [1, 2, 3])
assert True == is_in_list(1, [3, 2, 1])
assert False == is_in_list(1, [2, 2, 2])
assert False == is_in_list(1, [])
```

13. What is the output of the following lines of code (error is possible)? (1 point)

```
template = '{0} foxes and {0} {1}'
print(template.format('four', 'flamingoes'))

print('%s foxes and %s %s' % ('four', 'flamingoes'))
```

```
four foxes and four flamingoes

Error (TypeError: not enough arguments for format string)
```

14. Use the following list to answer the questions below...

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15]]
```

Part 1 - write the code described in the table below based on the list above (2 points)

Using **list indexing**, write code below to change the number **15** to the word '**fifteen**' and the number **4** to the word '**four**'.	Write a loop below that would change the 3rd element in every inner list to an exclamation point: [[1, 2, '!'], [4, 5, '!'] ...]
`a[-1][-1] = 'fifteen'` `a[1][0] = 'four'`	`for inner_list in a:` `  inner_list[2] = '!'`

Part 2 - write the code described in the instructions below based on the list above (3 points)

Loop through every element in every nested list in the variable **a**.

a) Print out the inner list index before printing out every element of the inner list
b) Example output:

```
List at index 0:
1
2
3
List at index 1:
4
5
6
```

```
count = 0
for inner_list in a:
  print('List at index %s' % count)
  for list_item in inner_list:
    print(list_item)
  count += 1
```

15. Write the following program... (2 points)

a) Stores a list of 4 words: `'dog', 'cat', 'bat', 'frog'`

b) It will then continually ask the user for a word. If the word, **in any casing, lower or upper,** appears in the stored list of words, stop asking for a word and prints out `'Done!'`.

c) Example output:

```
Word please!
> burrito
Word please!
> bat
Done!
```

```
words = ['dog', 'cat', 'bat', 'frog']
answer = ''
while answer.lower() not in words:
    answer = input('Word please!\n>')
print('Done!')
```

16. Write a program that asks for exactly 5 words. After all five words are entered, print them out in alphabetical order (hint: there's a list method to do this). (2 points)

```
words = []
for num in range(5):
    words.append(input('Word please!'))
words.sort()
for word in words:
    print(word)
```

17. What is the output of the following program? Use the grid to the right of the program as a guide; **each individual character of output can be placed in a single box**. **Leave a box blank to represent a space character**. You do not have to use all of the boxes. (3 points)

```
def create_table(size, letters):
    table = ''
    for i in range(1, size + 1):
        row = ''
        for c in letters:
            row += str(i) + c + ' '
        table += row + '\n'
    return table

def main():
    num, s = 4, 'abc'
    print(create_table(num, s))
main()
```

1	a		1	b		1	c		
2	a		2	b		2	c		
3	a		3	b		3	c		
4	a		4	b		4	c		

18. Create a **function called `make_acronym`**. It takes a string as an argument. It extracts words from the string by using space as a word separator. It takes the first letter of each word that **starts in uppercase** to create a new word. (4 points)
See the following examples below (the first one omits 'of' because it does not start with an uppercase letter).

```
>>> print(make_acronym("Bachelor of Arts"))
'BA'

>>> print(make_acronym("Too Long Didn't Read"))
'TLDR'
```

```
def make_acronym(s):
    acronym = ''
    words = s.split(' ')
    for word in words:
        if word[0].isupper():
            acronym += word[0]
    return acronym
```