Name:_____ **NetID:**_____ 

CSCI-UA.0002-008 – Midterm Exam #2

November 18th, 2015

Instructor: Joseph Versoza

Ask the person to your left for their first name
(leave blank if next to empty seat or wall):

_____

Ask the person to your right for their first name
(leave blank if next to empty seat or wall):

_____

Keep this test booklet closed until the class is prompted to begin the exam

- Computers, calculators, phones, textbooks, and notebooks are **not allowed** during the exam
- Please turn off your phone to avoid disrupting others during the exam
- The back of this cover sheet can be used as scratch paper

1. **Read the code sample** in the first column. **Answer the question** in the second column. (15 points)

Code	Question

`word = 'FREEZE'`
`letter = word[-1]`
`i = ord(letter)`
`c = chr(i - 3)`
`print(letter)`
`print(c)`
`print('Z' in word)`
	What is the output (shown on the screen) of this code? (3 points)
	
	`E`
	`B`
	`True`

`noise = 'buzz'`
`def make_electronic_music(sound):`
`  noise = 'bleep'`
`  print('%s... %s' % (noise, sound))`
 	
`result = make_electronic_music('bloop')`
`print(noise)`
`print(result)`
	What is the output (shown on the screen) of this code? (3 points)
	`bleep... bloop # these were the parameters`
	`# passed in`
	`buzz # noise is global, and is not`
	`# over-written`
	`None # nothing is returned, result is`
	`# None`

`a = [1, 2, 3]`
`b = a`
`c = a[:]`
`d = c.pop()`
`a.append([4, 5])`
`e = b.extend(c)`
	What values do the variables `a, b, c, d` and `e` contain? (5 points)
	
	`a:____[1, 2, 3, [4, 5], 1, 2]_____`
	
	`b:____[1, 2, 3, [4, 5], 1, 2]_____`
	
	`c:____[1, 2]_____`
	
	`d:_____3_____`
	
	`e:_____None_____`

`def find(needle, haystack):`
`  for item in haystack:`
`  if item == needle:`
`  return True`
`  else:`
`  return False`
 	
`print(find(4, [4, 3, 2, 1]))`
	(2 points)
	
	a. **How many times will the body of the for loop in the function run?**
	
	Once
	
	b. There 's a logical error in this program; it's supposed to return True if the needle exists in the list, haystack... and False otherwise. **Find a pair of arguments that reveal a logical error.**
	
	4, and [3, 4, 2, 1]

`def loop_with_return(x):`
`  print('starting...')`
`  for i in range(3, 6):`
`  print(i)`
`  return i`
`  print('done')`
 	
`print(loop_with_return(5))`
	What is the output (shown on the screen) of this code? (2 points)
	
	`starting...`
	`3`
	`3`

2. Using the following variable declaration, write out the output (error is possible) of the print statements below. (5 points)

```
animals = [['dog', 'cat'], ['bat'], ['goat'], ['cow', 'bee', 'ant']]
```

a) `print(animals[876])` (a) **error**

b) `print(animals[-3])` (b)`['bat']`

c) `print(animals[2] * 2)` (d)`['goat', 'goat']`

d) `print("animals"[-1])` (e) **s**

e) `print(animals[3][1:100])` (f)`['bee', 'ant']`

3. True or False (3 points)

a) (True / False) `'share' > 'scare'`

b) (True / False) `'foo' not in ['baz', 'bar', 'foo']`

c) (True / False)
```
squashed = [1, 2] + [3, 4]
squashed == [[1, 2], [3, 4]]
```

d) (True / False) Both strings and lists support the repetition operator (multiplication) .

e) (True / False) Local variables can be accessed outside of the function that they are defined in.

f) (True / False) A parameter is the name of the variable in a function definition that represents the value being passed in.

4. The intention of the following code is to create a function that would **take a non-empty list of ints (positive, negative, and mixed positive and negative ints), and return the largest int in that list**. For example, given [1, 2, 3], the function would return 3. Unfortunately, the implementation is slightly off! Test the program below. (2 points)

```
def max_int(numbers):
    largest_number = 0
    for n in numbers:
        if n > largest_number:
            largest_number = n
    return largest_number
```

Write assertions that will cover three unique test cases. There is a logical error in the program; find the error based on the implementation and description. Make your **last assertion** the one that uncovers the logical error.

a) `assert 9 == max_int([1, 5, 9]), 'only positive numbers'`

b) `assert 1 == max_int([1, -5, -9]), 'mixed positive and negative'`

c) `assert -1 == max_int([-1, -5, -9]), 'negative numbers only'`
(This assertion should uncover a logical error)

5. 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 (an empty box implies a space)**. You do not have to use all of the boxes. (4 points)

```
def make_pattern(n):
    for i in range(n):
        row = ''
        for j in range(n):
            if j == i or i == 0:
                row += str(j)
            else:
                row += " " # add a space
        print(row)

make_pattern(4)
```

0	1	2	3				
	1						
		2					
			3				

6. Create a function called **`uppercase_last`**. It should take two arguments: (6 points)

a) a **string** that specifies what character will separate words (for example, a hyphen: '`-`')
b) another **string** composed of words **separated by the character specified** (for example: '`word1-word2-word3`')
c) it will **give back the last word**, **all uppercase** (in the example above, `WORD3`)
d) assume there are at least two words in the string (you don't have to worry about an invalid argument being passed in)
e) Example usage:

```
print(uppercase_last('-', 'cabeza-mano-pie')) # gives back PIE
print(uppercase_last(',', 'cabeza,mano'))     # gives back MANO
```

```
def uppercase_last(separator, s):
    i = 0
    start = 0
    for ch in s:
        if ch == separator:
            start = i
        i += 1
    return s[start + 1:].upper()
```

```
def uppercase_last3(separator, s):
    word = ""
    for ch in s:
        if ch == separator:
            word = ""
        else:
            word += ch
    return word.upper()

def uppercase_last_3(s):
    parts = s.split(',')
    parts[-1] = parts[-1].upper()
    return ','.join(parts)
```

7. Your email account has recently been hacked because you used a terribly simple password (`pizza`). To prevent hackers from breaking into your email (and seeing your embarrassingly liberal use of emoji 😂😂😂) again, you decide to write a program to help you create stronger passwords. You'll do this in two parts: first, you'll create a function... and then you'll use that function in a short program. (10 points)

   a) Start off by creating a function called **`is_strong_password`**
   b) parameters: a string representing the password to test
   c) processing: determine if the password is strong by ensuring that it has:
      - at least one letter
      - at least one number
      - at least one of the following punctuation characters: period (.), comma (,), dollar($), exclamation (!) or hash (#)
      - is at least 10 characters long
   d) return: gives back true if the password meets the above requirements, false otherwise
   e) Then... use this function in a program that:
      - continually asks the user for a password (prompt the user by saying: '`Create a password`')
      - stops asking once they enter a password that meets the conditions outlined above for a 'strong' password

   Example function usage:

```
print(is_strong_password('asdf'))            # False
print(is_strong_password('asdfasdfasdf'))    # False
print(is_strong_password('asdf12341234'))    # False
print(is_strong_password('asdf1234^^^^'))    # False
print(is_strong_password('asdf1234!#!#'))    # True
```

   Example program interaction:

```
Create a password
> pizza
Create a password
> pizza1?
Create a password
> pizz4p!zza
```

```
def is_strong_password(s):
    letters, numbers, punctuation = 0, 0, 0
    for c in s:
        if c.isalpha():
            letters += 1
        elif c.isdigit():
            numbers += 1
        elif c in ['.', ',', '$', '!', '#']:
            punctuation += 1
    if letters > 0 and numbers > 0 and punctuation > 0 and len(s) >= 10:
        return True
    else:
        return False


while not is_strong_password(input('Create a password\n> ')):
    continue
```

8. You're a publisher of Star Trek fan fiction. Printed fan fiction doesn't sell in high numbers, so you decide to find creative ways to save money. One solution that you've come up with is to reduce the number of pages of each story by removing every vowel (regardless of case) in the works that you publish (it's totally still understandable without vowels, right?). So... you write a program to do it for you. (6 points total)

   Create a function called **`remove_vowels`**.

   a) It should take **two arguments**, the **string** that will have vowels removed from it, and an additional **boolean** value that determines whether or not the letter y should be considered a vowel (1 point)
   b) It will **give back a new string** with all vowels removed (2 points)
   c) If the second argument is True, it will count y as a vowel, and remove it from the incoming string. (1 point)

   Example output:

```
>>> print(remove_vowels('Typical Picard', True))
Tpcl Pcrd
>>> print(remove_vowels('Typical Picard', False))
Typcl Pcrd
```

```
def remove_vowels(s, include_y):
    new_s = ''
    vowels = 'aeiouAEIOU'
    if include_y:
        vowels = vowels + 'yY'
    for c in s:
        if c not in vowels:
            new_s += c
    return new_s
```

9. Create a function called **`in_nested_list`**. It will give back the indexes (as a 2-element list) of the first occurrence of a value within the sub lists of a list of lists. (6 points)

   a) parameters: **a value** to search for and a **list of lists** to search in; you can **assume that every value in the outer list is a list** and that there will only be **2 levels of nesting** (that is, 1 outer list with 1 or more inner lists)
   b) processing: searches each sub list for the specified value
   c) return: it will **give back a 2-element list** that represents the indexes to use to retrieve the specified value (the index of the outer first, the index of the inner second); **give back an empty list** if the value is not found.
   d) Example usage:

```
numbers = [[1, 2], [6, 1, 7, 15, 23, 24], [12, 15, 0]]

# 23 can be retrieved by indexing into the list of lists, numbers, using 1... then 4
print(in_nested_list(23, numbers))   # --------> prints out [1, 4]

# 1000 does not exist in the list of lists, numbers
print(in_nested_list(1000, numbers)) # --------> prints out []
```

```
def in_nested_list(v, numbers):
    for index_1 in range(len(numbers)):
        for index_2 in range(len(numbers[index_1])):
            if numbers[index_1][index_2] == v:
                return [index_1, index_2]
    return []
```

10. It's 1984, and you've been hired by the Ministry of Truth to redact information from lists of words. You have a list of illegal words, and you have an incoming list of uncensored words. If any word in the uncensored list is in the list of illegal words, the first three characters of the illegal word must be replaced with dashes (-'s). If the word is three letters or less, then the entire word must be replaced with a series of dashes equal to the length of the word. For example: '`hello`' is converted to '`---lo`', '`hey`' is converted to '`---`', and '`hi`' is converted to '`--`'. You decide to make a **function called `redact`** to do this work for you. (6 points)

    Your function takes takes **two arguments, `words` and `illegal_words`**. It returns:

    a) a new list of strings composed of all of the strings in the original list, words
    b) but if a string in word is in the list of illegal words, it must be partially replaced by -'s
    c) if the word is three letters or less, the whole word is replaced by a series of -'s equal to the number of characters in the string; otherwise, the first three letters are replaced by -'s
    d) example output below:

```
>>> print(redact(['hi', 'hello', 'there'], ['hi', 'hello']))
['--', '---lo', 'there']
>>> print(redact(['cute', 'puppies', 'eating', 'a', 'cake'], ['puppies', 'cake', 'a'])
['cute', '---pies', 'eating', '-', '---e']
```

```
def redact(words, illegal_words):
    redacted = []
    for word in words:
        if word in illegal_words:
            if len(word) <= 3:
                redacted.append('-' * len(word))
            else:
                redacted.append('---' + word[3:])
        else:
            redacted.append(word)
    return redacted
```

11. **Extra Credit (2 points total)**

    a) What is the name of a function that is defined in terms of itself?

    recursive

    b) What's the output of the code below?

```
def foo(nums):
    if len(nums) == 1:
        return nums[-1]
    else:
        return nums[0] + foo(nums[1:])

print(foo([1, 2, 3, 4]))
```

    10

Name:_____ **NetID:**_____

Scratch Paper and Reference Material

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

String Methods

```
capitalize
count
endswith
find
format
index
isalnum
isalpha
isdecimal
isdigit
islower
isnumeric
isprintable
isspace
istitle
isupper
join
lower
replace
split
startswith
strip
title
upper
```

List Methods

```
append
count
extend
index
insert
pop
remove
reverse
sort
```