CS 301 - Fall 2016
Instructor: Laura Hobbes LeGault

Midterm Exam 3 — 16.67%

(Last) Surname: _____ (First) Given name: _____

NetID (email): _____ @wisc.edu

IMPORTANT: Answers for Dual and Multiple Choice questions *must* be marked on a scantron. The answer marked on the scantron will be the only answer graded.

Fill in these fields (left to right) on the scantron form (use #2 pencil):
1. LAST NAME (surname) and FIRST NAME (given name), fill in bubbles
2. IDENTIFICATION NUMBER is your Campus ID number, fill in bubbles
3. Under ABC of SPECIAL CODES, write 001 (morning lecture), fill in bubbles
4. Under J of SPECIAL CODES, write A (exam version), fill in bubble 0

I certify that I will keep my answers covered and do my best to not allow my exam paper to be viewed by another student during the exam or prior to completion of their exam. I also certify that I have not viewed or in any way used another's work in completing my answers. I understand that being caught allowing another to view my work or being caught viewing another's work are both violations of this agreement and either will result in automatic failure of the course and an academic misconduct letter to the Deans Office for myself and any other individuals involved.

Signature: _____

---

The following exam has 21 questions and is worth a total of 84 points. You will have 50 minutes to complete the exam. **Be sure to read through every question completely.**

1. **Dual Choice** — 8 questions worth 2 points each.

2. **Multiple Choice** — 11 questions worth 4 points each. Choose the *best* answer.

3. **Fill-in-the-blank** — 2 questions worth 12 points each. Be complete.

You may not use notes or books, your neighbors, or calculators or any other electronic devices on this exam. **Turn off and put away** any portable electronics now.

Disclaimer: the following are provided for your reference only, and the inclusion of information here does not guarantee it will be used on the exam.

Operator Precedence Table:

level	operator	description
	(<expression>)	grouping with parentheses
	x[index:index]	slicing
higher	x[index]	indexing
	* / %	multiplicative
	+ -	additive
	< <= > >=	relational
	== !=	equality
	not	logical not
	and	logical and
lower	or	logical or
	= += *=	(compound) assignment

Built-in functions:

`raw_input(p)`	Displays prompt `p` and returns the user's input as a string.
`len(s)`	Return the length (the number of items) of an object.
`range(n)`	Returns a list of `n` consecutive integers beginning at `0`.
`min(x)`	Returns the smallest item in the iterable `x`.
`max(x)`	Returns the largest item in the iterable `x`.

Constants and functions from the math module:

`math.pow(x,y)`	Returns `x` raised to the power `y`. Converts both arguments to floats.
`math.pi`	The mathematical constant $\pi = 3.141592...$

Functions from the os module:

`os.path.exists(p)`	Returns `True` if the file at path `p` exists, `False` otherwise.

List and dictionary methods:

`list.append(x)`	Add the value `x` to the end of `list`.
`list.insert(i,x)`	Insert the value `x` at the `i`th index of `list`.
`dict.keys()`	Return a copy of `dict`'s list of keys.
`dict.values()`	Return a copy of `dict`'s list of values.

String methods:

`w.isalpha()`	Return true if all characters in string `w` are letters.
`w.isdigit()`	Return true if all characters in string `w` are numbers.
`w.isspace()`	Return true if all characters in string `w` are whitespace.
`w.lower()`	Return a copy of the string `w` with all letters lowercase.
`w.upper()`	Return a copy of the string `w` with all letters uppercase.

Files:

`open(p,m)`	Opens the file at path `p` in mode `m`, returning an object of `file` type.
`f.read()`	Returns the entire contents of the file object `f` as a string.
`f.readline()`	Returns the next line of the file object `f` as a string.
`f.readlines()`	Returns the entire contents of the file object `f` as a list of strings.
`f.write(s)`	Writes the string `s` to the file object `f`.
`f.close()`	Closes the file object `f`.

Functions from the numpy module (as np):

`np.array(L,t)`	Returns the list `L` as an array containing elements of type `t`.
`np.arange(n)`	Returns an array of `n` integers from 0 to `n-1`.
`np.mean(a)`	Returns the mean (average) of the elements of array `a`.
`np.std(a)`	Returns the standard deviation of the elements of array `a`.
`np.random.rand(n)`	Returns an array of `n` uniformly distributed floats between 0 and 1.

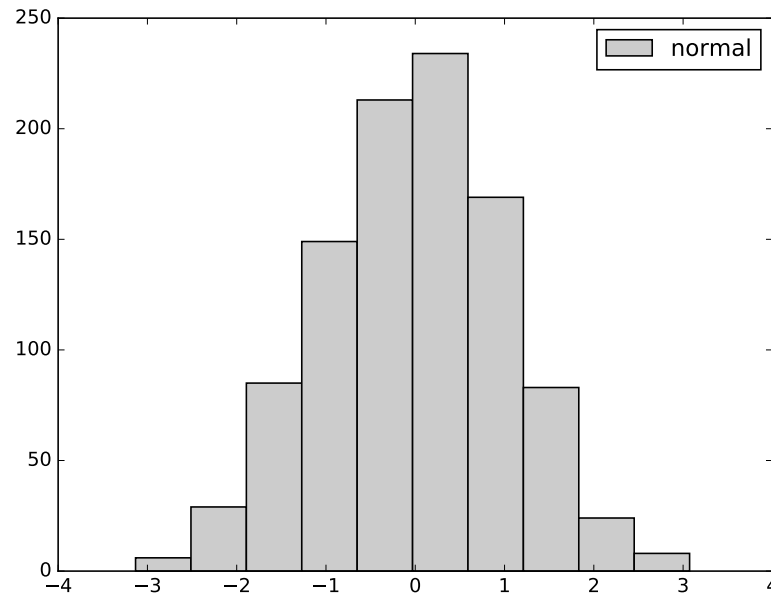## Functions from matplotlib.pyplot (as plt):

`plt.plot(x,y)`	Plots `(x,y)` coordinates using default line style and color.
`plt.xlabel(s)`	Sets the x-axis label of the current plot to `s`.
`plt.ylabel(s)`	Sets the y-axis label of the current plot to `s`.
`plt.title(s)`	Sets the title of the current plot to `s`.
`plt.legend()`	Places a legend on the axes for all labelled lines.
`plt.hist(a,n)`	Divides the values in `a` into `n` bins and plots a histogram.
`plt.pie(a)`	Creates a pie chart with wedges proportional to the values in `a`.
`plt.show()`	Display a figure and pause until the figure has been closed.

Dual Choice: Terminology

1. The code   (2)

    ```
    total += a.vol
    ```

    adds _____ to the variable `total`.

    A. the value of `a`'s `vol` property

    B. the return value from `a`'s `vol` method

2. The `__repr__()` method must _____ an object's string representation.   (2)

    A. return

    B. print

3. Array elements must all have _____ types.   (2)

    A. the same

    B. numeric (int or float)

4. A `try` block is _____ followed by an `except` block.   (2)

    A. sometimes

    B. always

5. `random.shuffle(deck)` is an example of a _____.   (2)

    A. function

    B. method

6. `plt.show()` should be called _____ any other calls to modify a plot.   (2)

    A. before

    B. after

7. `../Documents/weather.csv` is an example of a(n) _____ path.   (2)

    A. absolute

    B. relative

8. If a file **does not** exist, it may still be opened for _____.   (2)

    A. writing

    B. reading

Multiple Choice: Reading code

9. Which of the following commands was **not** used to create the plot below? (4)

    A. `plt.show()`

    B. `plt.hist()`

    C. `plt.title()`

    D. `plt.legend()`

10. What sort of graph is displayed by the following complete program, when run on the command line? (4)

```
import matplotlib.pyplot as plt

plt.plot([0,1,2], [1,2,3])
plt.plot(range(12))
```

    A. A line graph

    B. A pie graph

    C. A histogram

    D. No output is displayed

11. If a TypeError occurs on `LINE Z` in the following code, which line will **not** be executed? (4)

```
try:
    LINE Z
    LINE A
except TypeError:
    LINE B
LINE C
```

    A. `LINE A`

    B. `LINE B`

    C. `LINE C`

    D. None of these will run, the program will crash.

12. What is the *type* of the variable `x` after the following code has run? Assume all modules have been imported properly. (4)

```
x = os.path.exists("exam3.txt")
```

    A. file

    B. string

    C. boolean

    D. list

13. What is the *type* of the variable `x` after the following code has run? Assume numpy has been imported as `np`. (4)

```
x = np.array(["1","2","3","4","5"], int)[0]
```

    A. int

    B. array

    C. string

    D. This code will produce a ValueError.

14. A user runs a program from the command line by typing (4)

```
$ python exam3.py 50
```

Which of the following puts the value 50 into a variable `x`?

    A. `x = sys.argv()`

    B. `x = argv`

    C. `x = os.argv[0]`

    D. `x = sys.argv[1]`

15. Given the following list of imported modules from the beginning of a file, which function call is *not* legal? (4)

    ```
    import numpy
    import matplotlib.pyplot as plt
    import random
    ```

    Pay attention to **module names** - all functions are otherwise called and used correctly.

    A. `a = np.array(num,int)`

    B. `num = [ random.randint(1,10) for i in range(500) ]`

    C. `a = numpy.random.randint(1,11,size=500)`

    D. `plt.plot(a)`

16. Why may the following code cause an error? **Pay attention to details!** (4)

    ```
    f = open('doesnotexist.txt', 'w')
    s = f.read()
    ```

    A. The file was opened in write mode and cannot be read.

    B. The file `doestnotexist.txt` does not exist.

    C. Bad syntax; the first line should be `f.open('doesnotexist.txt', 'w')`.

    D. If `doesnotexist.txt` is too large, `read()` will cause a memory overflow.

17. NumPy's documentation includes the following entry for `numpy.random.randint()`: (4)

    `numpy.random.randint(low, high=None, size=None)`

    Returns a *size*-shaped array of random integers (or a single such random int if *size* not provided) from the "discrete uniform" distribution in the "half-open" interval [low, high). If high is None (the default), then results are from [0, low).

    Which of the following is **not** a legal call to `numpy.random.randint()`?

    A. `numpy.random.randint()`

    B. `numpy.random.randint(5, size=(2,4))`

    C. `numpy.random.randint(10)`

    D. `numpy.random.randint(low=2, high=5)`

18. A program you are running displays the following output in the console: (4)

    ```
    <__main__.Animal instance at 0x02855328>
    ```

    Which of the following statements about this program is necessarily **true**?

    A. The program crashed.

    B. The program stores Animal objects in a dictionary.

    C. The Animal class does not contain a __repr__() method.

    D. The Animal class __init__() method expects a numeric argument.

19. Which of the following if statements could be used instead of try in this case to prevent (4)
    an IndexError from occurring?

    ```
    x = raw_input("Enter a word: ")
    try:
        print x[index]
    except IndexError:
        print "Not enough letters!"
    ```

    A. if len(x) < index:

    B. if len(x) > index:

    C. if len(x) == index:

    D. No if necessary, this code will never cause an IndexError.

Fill-in-the-blank: Writing code

For each of the questions on this page, fill in the value, operator, or statement needed to produce the indicated output. Each line is worth **6 points**.

20.
```
def normalize_file( text ):
    """ As in I'm Feeling Lucky, you'll be normalizing text from a file.
        Instead of reading the file, the contents of the article is the
        single string argument.  Transform all letters to lowercase,
        and remove all punctuation and numbers.

            input:  "I am Lord Voldemort!!111eleven"
            output:  "i am lord voldemorteleven"
    """

    newstring = ""          # will contain the normalized string

    for letter in _____:                                    (6)
        if letter.isalpha()):

            c = _____        (6)
        elif letter.isspace():
            c = letter
        else:
            c = ""          # empty string
        newstring += c      # add c to the end of newstring
    return newstring
```

For the last question, you will write a complete function.

21. Your job is to implement a function to add two lists as though they were arrays **without using NumPy**. (12)

Write a function named `question_21` which expects two **lists** as arguments and returns a **list** that contains their **element-wise** sum (**not** the `sum()` of their elements). Be careful to add elements at matching indexes in the two lists, and don't cause an IndexError!

For example:

```
>>> print question_21([1,2,3], [4,5,6])
[5, 7, 9]
>>> print question_21([1,2,3], [4,5])
[5, 7, 3]
```

You do not *need* to comment your code, but recall that your grade is based on how well I can understand what you're trying to do.

This page intentionally left blank.