

CS 301 - Fall 2017
Instructor: Laura Hobbes LeGault and Alexi Brooks
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 for those portions. Answers for Fill-in-the-blank questions must be marked on this exam.

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 your **lecture number**, fill in bubbles:
 001 - MWF 8:50a (Hobbes morning)
 002 - MWF 1:20p (Hobbes afternoon)
 003 - TR 9:30a (Alexi)
4. Under *F* 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 22 questions and is worth a total of 43 points. You will have 50 minutes to complete the exam. **Be sure to read through every question completely.**

1. **Dual Choice** — 10 questions worth 1 point each. Choose the *best* answer.
2. **Multiple Choice** — 9 questions worth 2 points each. Choose the *best* answer.
3. **Fill-in-the-blank** — 3 blanks worth 3 points each, 1 worth 6 points. 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
higher	(<expression>)	grouping with parentheses
	x[index:index]	slicing
	x[index]	indexing
	* / %	multiplicative
	+ -	additive
	< <= > >=	relational
	== !=	equality
	not	logical not
lower	and	logical and
	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)` Returns 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`.

The random module:

`random.randint(a,b)` Returns a random integer in `[a, b]`.
`random.shuffle(s)` Randomly changes the order of the sequence `s` in place.

The os and sys modules:

`os.path.exists(p)` Returns `True` if the file at path `p` exists, `False` otherwise.
`sys.argv` The list of command line arguments passed to a Python script.

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()` Returns a copy of `dict`'s list of keys.
`dict.values()` Returns a copy of `dict`'s list of values.

String methods:

<code>w.isalpha()</code>	Returns <code>True</code> if all characters in string <code>w</code> are letters.
<code>w.isdigit()</code>	Returns <code>True</code> if all characters in string <code>w</code> are numbers.
<code>w.isspace()</code>	Returns <code>True</code> if all characters in string <code>w</code> are whitespace.
<code>w.lower()</code>	Returns a copy of the string <code>w</code> with all letters converted to lowercase.
<code>w.upper()</code>	Returns a copy of the string <code>w</code> with all letters converted to uppercase.

Files:

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

The numpy module (as np):

<code>np.append(a,x)</code>	Returns a copy of array <code>a</code> with value <code>x</code> appended to it.
<code>np.array(L,t)</code>	Returns the list <code>L</code> as an array containing elements of type <code>t</code> .
<code>np.arange(n)</code>	Returns an array of <code>n</code> integers from 0 to <code>n-1</code> .
<code>np.mean(a)</code>	Returns the mean (average) of the elements of array <code>a</code> .
<code>np.std(a)</code>	Returns the standard deviation of the elements of array <code>a</code> .
<code>np.random.rand(n)</code>	Returns an array of <code>n</code> uniformly distributed floats in <code>[0,1)</code> .
<code>np.random.rand(a,b)</code>	Returns an <code>a×b</code> array of uniformly distributed floats in <code>[0,1)</code> .

The pandas module (as pd):

<code>pd.DataFrame(d)</code>	Returns a 2-dimensional data structure from data <code>d</code> . Optional argument: <code>columns</code> with labels for each column
<code>df.plot(x=x,y=y)</code>	Plots <code>(x,y)</code> coordinates as a line plot. Optional argument: <code>label</code> to label the series Optional argument: <code>title</code> as a title for the graph Optional argument: <code>color</code> to control the series' color
<code>df.plot.scatter(x=x,y=y)</code>	Plots <code>(x,y)</code> coordinates as a scatter plot.
<code>df.plot.hist()</code>	Plots the values in DataFrame <code>df</code> as a histogram.
<code>ax.get_figure()</code>	Returns a Figure object containing the current graph.
<code>fig.savefig(n)</code>	Saves the figure to a file named <code>n</code> .

Dual Choice: Terminology

1. A **relative** path always begins at the _____ . (1)
A. current working directory
B. root directory
2. If `A.function()` is a **method** call, then `A` must be a _____ . (1)
A. module
B. variable
3. Appending to a **NumPy array** adds the new element _____ . (1)
A. to a deep copy
B. in place
4. Given the **method call** `A.do_thing(2, 3)`, which of the following method headers has the correct parameter list? (1)
A. `def do_thing(self, x, y):`
B. `def do_thing(x, y):`
5. When calling `df.plot()` on a `DataFrame` with two columns, the default x values are _____ . (1)
A. column 0
B. indexes
6. Looping through a file object `f`, opened in read mode, is equivalent to looping through the value returned by _____ . (1)
A. `f.readlines()`
B. `f.read()`
7. The code `ditto = Ditto()` calls the _____ method. (1)
A. `Ditto()`
B. `__init__()`
8. The value at `sys.argv[0]` is _____ . (1)
A. The name of the .py file as a string
B. "python"

9. Code that runs **after an error occurs** should be placed in a _____ block. (1)
- A. `try`
 - B. `except`
10. The legend labels on a `df.plot.scatter()` plot are from _____. (1)
- A. the DataFrame's column labels
 - B. the optional `label` argument

Multiple Choice: Reading code

11. Which of the following `if` statements could be used in place of the `try` statement to prevent an `IndexError` from occurring? Assume `index` is some positive integer. (2)

```
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, `print x[index]` will never cause an `IndexError`.
12. What *data type* is returned by a correctly implemented `__str__(self)` method? (2)
- A. `object`
 - B. `<__main__.Object instance at 0x102844cd0>`
 - C. `str` (**string**)
 - D. `NoneType`
13. What is the *data type* of `x` after the following code executes? (2)

```
import numpy as np
x = np.array([1,2,3], float)[1]
```

- A. `float`
- B. `array`
- C. `int`
- D. None, this code causes a `TypeError`.

14. Which of the following import statements will cause this line to work without error? (2)

```
nums = rand(5)
```

- A. `from numpy.random import rand`
- B. `import numpy.random as rand`
- C. `import numpy as np`
- D. No import syntax allows NumPy's `rand()` function to be called like this.

15. What is the *value* in `x` after the following complete program is executed? (2)

```
import sys
x = sys.argv[-1]
```

Assume this program is contained in a file called `example_code.py` and was called on the command line with

```
$ python example_code.py command line argument
```

- A. `example_code.py`
- B. `command`
- C. `argument`
- D. This code causes a `SyntaxError`, line 2 should be `x = sys.argv(-1)`.

16. Which of the following lines will **not** run if **ERROR LINE** **does not** cause any error? (2)

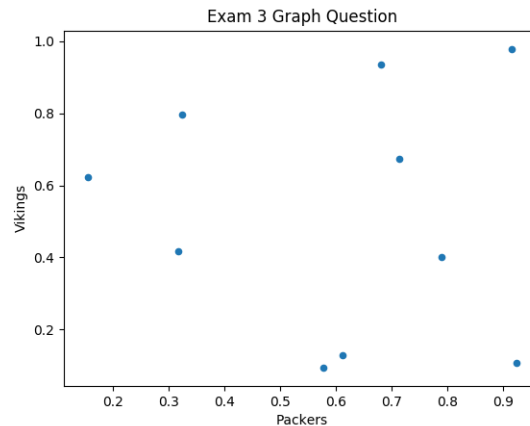
```
try:
    LINE A
    ERROR LINE
except TypeError:
    LINE B
LINE C
```

- A. The program will crash without running either `LINE B` or `LINE C`.
- B. `LINE A`
- C. `LINE B`
- D. `LINE C`

17. What is the *value* in `x` after the following code executes, assuming that the file `exam3.txt` exists? (2)

```
f = open("exam3.txt", "w")
x = f.readlines()[0]
f.close()
```

- A. `str`
 - B. `list`
 - C. `NoneType`
 - D. This code causes an error, a file opened in "w" mode cannot be read.
18. Which of the following statements was **not** present in the code which created the image file `figure.png`, shown here? Assume `numpy` and `pandas` were imported as `np` and `pd`, respectively. (2)



- A. `ax = df.plot(title="Exam 3 Graph Question")`
 - B. `ax.get_figure().savefig("figure.png")`
 - C. `data = np.random.rand(10,2)`
 - D. `df = pd.DataFrame(data, columns=["Packers", "Vikings"])`
19. What is the *value* in `x` after the following code executes? (2)

```
import numpy as np
x = np.array([1,2]) + np.array([5,10])
```

- A. `[1,2,5,10]`
- B. `[[1,2], [5,10]]`
- C. `[6,12]`
- D. None, this code causes a `TypeError`.

Fill-in-the-blank: Writing code

For each of the blanks in question 20, fill in the statement needed to produce the indicated output. Each line is worth **3 points**.

20. `import random`

`with _____: # write to q20.txt` (3)

`for i in range(10):`

`# write a random int 1-100 on each line of the file`

`outfile.write(_____)` (3)

21. The following complete program contains **2** errors, which could be of the following types: (1) a `SyntaxError`, (2) a `NameError`, (3) a logic error, or (4) an `IndexError`. For full credit, **circle** each error **AND** indicate which type of error it is. (4)

Assume the file `input.txt` exists and contains five lines of text.

```
while len(line) != 0:
```

```
    f = open("input.txt", "r")
```

```
    line = f.readline()
```

```
    print line          # prints each line of the file
```

```
    f.close()
```

22. `class Ditto:` (5)

```
    """ Add a method to this class so that when Ditto() is called,
        a Ditto object with the following properties is returned:
```

```
        power = ["absorb"]
```

```
        count = 0
```

```
        level = [1]
```

```
    """
```


If you need extra space for question 22
(or for any other question),
please feel free to use this page.