

CS 301 - Spring 2017  
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 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 42 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.
2. **Multiple Choice** — 10 questions worth 2 points each. Choose the *best* answer.
3. **Fill-in-the-blank** — 2 questions worth 6 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
higher	( <expression> )	grouping with parentheses
	x[index:index]	slicing
	x[index]	indexing
	* / %	multiplicative
	+ -	additive
	< <= > >=	relational
	== !=	equality
lower	not	logical not
	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)` 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`.

### The math module:

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

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

**String methods:**

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

**Files:**

<code>open(p,m)</code>	Opens the file at path <code>p</code> in mode <code>m</code> , returning an object of file type.
<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.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 between 0 and 1.

**The matplotlib.pyplot module (as plt):**

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

## Dual Choice: Terminology

1. When creating a histogram using matplotlib, the data provided are \_\_\_\_\_. (1)  
A. a list of heights of each bar  
**B. a list of numbers to be divided up into bins**
2. Array elements must all be of \_\_\_\_\_ types. (1)  
A. numeric (int or float)  
**B. the same**
3. This is an example of a **relative path**: \_\_\_\_\_. (1)  
**A. ./Documents/file.txt**  
B. C:\Users\Hobbes\Documents\file.txt
4. The `__str__()` method must \_\_\_\_\_ an object's string representation. (1)  
**A. return**  
B. print
5. Code that may cause an error should be placed in a \_\_\_\_\_ block. (1)  
**A. try**  
B. except
6. To **display** a graph, you must use the \_\_\_\_\_ function. (1)  
A. `plt.plot()`  
**B. plt.show()**
7. In the call `random.shuffle(deck)`, `random` is a \_\_\_\_\_. (1)  
A. method  
**B. module**
8. If `update()` is a **method** of the class `Point` and `loc` is an object of that class, then \_\_\_\_\_ correctly calls the method. (1)  
A. `Point.update()`  
**B. `loc.update()`**

9. To add a new row to an existing CSV file without erasing its contents, you must first open it in \_\_\_\_\_ mode. (1)
- A. "a"
  - B. "w"
10. If `os.path.exists("file.txt")` returns `False`, opening `file.txt` in \_\_\_\_\_ mode causes an `IOError`. (1)
- A. write
  - B. read

### Multiple Choice: Reading code

11. A program you are running displays the following output in the console: (2)
- ```
<__main__.Animal instance at 0x02855328>
```
- Which of the following statements about this program is necessarily **true**?
- A. The `Animal` class' `instance()` method expects no arguments.
  - B. The program has crashed.
  - C. The program stores `Animal` instances in a dictionary.
  - D. The `Animal` class does not contain a `__str__()` method.
12. What is the *value* in `x` after the following code executes? (2)
- 
- ```
import numpy as np

a = np.arange(1, 3, .5)
b = np.arange(4)
x = a + b
```
- 
- A. `[1.5 2.5 0.0 1.0 2.0 3.0]`
  - B. `[1.0 2.5 4.0 5.5]`
  - C. `a` and `b` contain different **types**, which will cause this program to crash.
  - D. `a` and `b` have different **lengths**, which will cause this program to crash.

13. What is the *type* of the variable `x` after the following code has run? Assume numpy has been imported as `np`. (2)

```
y = np.array(["1.0", "2.0", "3.0"], float)
x = y[1]
```

- A. `str`
- B. `float`**
- C. `array`
- D. This code will produce an error when casting "1.0" to a float.

14. Which of the following is a valid header for a function which will give a starting set of properties to an instance of the `Animal` class? (2)

- A. `def instance(Animal):`
- B. `def __str__():`
- C. `def __init__(self):`**
- D. `def Animal():`

15. Given the following list of imported modules from the beginning of a file, which function call is *not* legal? (2)

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

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

- A. `matplotlib.pyplot.plot([0,1], [100,100])`**
- B. `randint(1,20)`
- C. `plt.pie([40, 20, 20, 20])`
- D. `numpy.array([1,2,3])`

16. Why does the following code cause an error? **Pay attention to details!**

(2)

---

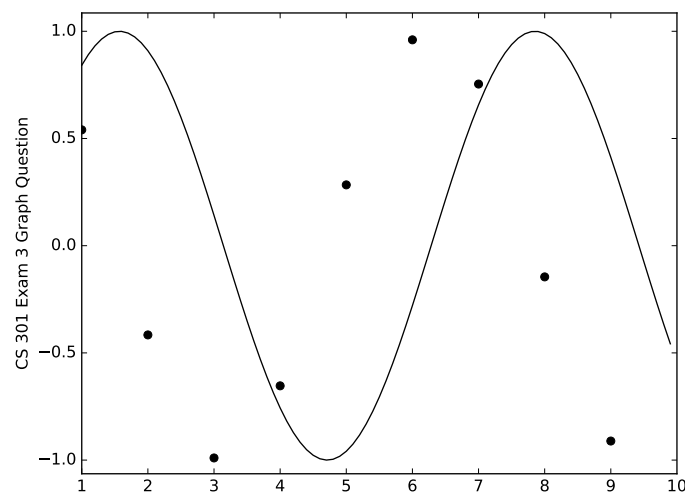
```
with open("data.txt", 'r') as f:
    for i in range(10):
        f.write(str(i)+"/n")
```

---

- A. The variable `i` must be cast to a float before it can be written to the file.
- B. The newline character is `"\n"`, not `"/n"`.
- C. The file is opened in "r" mode but the `write()` method is called.**
- D. The file is never closed.

17. Which of the following commands was **not** use to create the graph shown here? Assume `x` is an array of integers 1-9.

(2)



- A. `plt.plot(x, np.cos(x), "ko")`
- B. `plt.ylabel("CS 301 Exam 3 Graph Question")`
- C. `plt.legend()`
- D. `plt.show()`

18. Given that a program containing the following code executes **without error**: (2)

```
x = int(sys.argv[1])
```

Which of the following did the user type *on the command line* to run the program?

- A. `from functions_file import *`
- B. `import sys.argv`
- C. `python functions_file.py data.txt 1 2 3`
- D. `python functions_file.py 50`

19. Which of the following lines **will run** if a `TypeError` occurs on `ERROR LINE`? (2)

---

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

---

- A. `LINE A`
- B. `LINE B`
- C. `LINE C`
- D. The program will crash without running any of these lines.

20. Which of the following `if` statements could be used instead of `try` to **prevent** `ValueErrors` from occurring? (2)

---

```
num = # some string
try:
    x = int(num)
except ValueError:
    print "An error occurred."
```

---

- A. `if num[0] == "-" or num[0].isdigit():`
- B. `if len(num) > 0:`
- C. `if num.isdigit():`
- D. `if not num.isalpha():`



**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**.

```
21.  import numpy as np          # if you want to use it later
      def check_input( row ):
          """ The check_input() function expects one string of user input.
              It returns True if and only if the string contains 3 integers,
              separated by spaces.  FOR EXAMPLE (should work in general):

                  >>> print check_input("1 2 3")
                  True
                  >>> print check_input("a b c")
                  False
                  >>> print check_input("1 2")
                  False
          """
          values = row.split()

          if _____:      # there may not be 3 numbers      (3)
              return False

          try:

              _____      # cast elements to ints      (3)
              return True

          except ValueError:
              return False
```

Please turn to the next page - there's one more question.

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

22. Write a function called `question_22` which takes a file path AND a dictionary of vending machine contents as arguments. If the file exists, it will contain a list of items to add to the dictionary, with one item on each line: (6)

```
cola
cola
m&ms
snickers
m&ms
cola
sun chips
```

If the file exists, update the dictionary in place with a count of items of each type. If the file does not exist, end the function without crashing.

For example:

```
>>> contents = {}
>>> question_22("input.txt", contents)
>>> print contents
{'sun chips': 1, 'm&ms': 2, 'snickers': 1, 'cola': 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.

Nitwit, blubber, oddment, tweak.