Math 1MP3: midterm test 2, fall 2019
19 November 2019
Instructors: Dr. Bolker and Dr. Pang

MacID .....

You have one hour to complete the test. Please answer the questions on the same page as they are listed. No calculators or other test aids are allowed. There are 8 questions worth a total of 120 points. Good luck!

1. (16 points) **dictionaries and sets**: what are the results of the following Python commands?

```
D = {"A": 1, "B": 2, "C": 4}
S1 = {"a", 1, "b", 2, "b", 1, "c"}
S2 = {"a", "b"}

a. print("C" in D)
b. print(4 in D)
c. print("A" in D.items() )
d. print(len(D))
e. D["C"] = 56; print(D)
f. D = {"A": 1, "B": 2, "C": 4}; del D["A"]; print(D)
g. print(S1)
h. print(S1.issubset(S2))
```

solutions:

a. True
b. False
c. False
d. 3
e. {'A': 1, 'B': 2, 'C': 56}
f. {'B': 2, 'C': 4}
g. {1, 2, 'a', 'b', 'c'}
h. False

2. (12 points) **numpy**: what are the results of the following Python commands? (Assume we have already run `import numpy as np` in each case.)

   a.

```
a = np.eye(3, dtype = int)
a[a>0] = 5
print(a)
```

   b.

```
a = np.arange(6).reshape((3,2))
print(a.sum(axis = 1))
```

   c.

```
sqrt2 = np.sqrt(2)
sqrt4 = np.sqrt(4)
print(sqrt2**2 == 2.0)
print(np.isclose(sqrt4**2, 4.0))
```

solutions:

   a. [[5 0 0] [0 5 0] [0 0 5]]
   b. [1 5 9]
   c. False True

3. (16 points) Each of the following code chunks has a problem. Explain what problem/error each will produce (you only need to state the **first** error that will occur; assume that each chunk is run in a clean Python session, i.e. no variables have been defined and no modules have been loaded):

a.

```
D = dict( (("a",3), ("c", 3)) )
D.sort()
```

b.

```
L = list(range(8))
L += list(range(3))
S1 = set(L)
print(S1[3])
```

c.

```
import numpy as np
a = np.array([[1,2,3], [4,5,6]])
b = np.array([1,2])
diff = a - b
print(diff)
```

d.

```
import numpy as np
m = np.arange(9).reshape( (5,5) )
print(m[0,1])
```

solutions:

a. 'dict' objects can't be sorted
b. 'set' object is not subscriptable OR 'set' is not ordered OR 'set' cannot use index(slicing)

c. operands could not be broadcast together with shapes (2,3) (2,) OR the sizes (shapes) of these two array don't match
d. cannot reshape array of size 9 into shape (5,5) OR 9 items cannot be reshaped into a 5x5 array

4. (16 points) Write function `studentname(f)` that takes one file name `f` and returns a list. The file contains an unknown number of lines, each consisting of a student's name and an integer, separated by one or more spaces. Your function should read through the file and return a list of all of the names, **without any repeats**, as a `list`. For example, if the file was:

```
Mary 5
Alex 19
Mary 78
Ben 34
Mark 24
```

your function should return `["Mary", "Alex", "Ben", "Mark"]` (in any order).

solutions:

```
def studentname(f):
    f = open(f)
    studentname = set()
    for line in f:
        name, score = line.strip().split(" ")
        studentname.add(name)
    f.close()
    studentname = list(studentname)
    return studentname
```

or

```
def studentname(f):
    f = open(f)
    studentname = []
    for line in f:
        name = line.split()[0]
        if not name in studentname:
            studentname.append(name)
    return studentname
```

remarks:

1. sets do not have an addition operator `+`
2. Instead of using sets, one can use an `if` conditional to remove repeats
3. you can't use `{}` to initialize a set ({} is the way to initialize a dict)
4. without `strip()`, the answer is still correct 5. split doesn't really need `" "` (space-sep is the default)

5. (16 points) A dictionary called `register` uses student numbers as keys: the value for each student number is a list of the course numbers for all the courses that the student is enrolled in. Write Python code (*not* a function) that creates a new dictionary called `course` that has the course numbers as keys and a list of student numbers as values. For example, if the dictionary `register` is

```
{12: ["1MP3", "1F03"], 23: ["1F03", "1ZA3"], 21:["1MP3", "1ZA3"],
 15:["1ZA3"]}
```

then `course` should be

```
{"1MP3": [12, 21], "1F03": [12,23], "1ZA3": [23,21,15]}
```

solutions:

```
course = {}
for k in register:
    for c in register[k]:
        if c in course:
            course[c].append(k)
        else:
            course[c] = [k]
```

or

```
for k in register:
    for c in register[k]:
        if not c in course:
            course[c] = []
    course[c].append(k)
```

remarks:

1. Cannot assume the original course number list has a fixed length
2. Need a "if" condition to check the existing keys of dict "course"

6. (16 points) Write a function `blackjack()` that simulates a game. The player starts each game with a counter value equal to 0. In each turn, use the `random.randrange()` function to pick a single random integer between 1 and 10 and add this number to the counter. Keep taking turns as long as the counter is less than or equal to 16.

   Once the value of the counter is greater than 16, choose a value to return. If the counter is greater than 21, return the string "bust"; if the counter is exactly equal to 21, return the string "blackjack"; otherwise, return the numeric value of the counter.

   Assume your code is being run in a clean Python session (no modules loaded).

solutions:

```
import random
def blackjack():
    counter = 0
    while counter <= 16:
        rnum = random.randrange(1,11)
        counter += rnum
    if counter > 21:
        return "bust"
    if counter == 21:
        return "blackjack"
    return counter
```

Or:

```
import random
def blackjack():
    nums = list(range(1,11))
    counter = 0
    while counter <= 16:
        rnum = random.choice(nums)
        counter += rnum
```

```
    if counter > 21:
        return "bust"
    if counter == 21:
        return "blackjack"
    return counter
```

remarks:

1. need to remember to `import random`

2. use `while` loop with correct exit condition
3. use correct conditions to return final result

7. (16 points) Write a function `col_means()` that takes a filename `fn` as an argument. Assuming that the file contains an unknown number of rows and columns of numeric values separated by spaces, your function should return a `numpy` array containing the mean values of each column. For example, if the file is

```
0 1 2
3 3 3
4 2 -1
1 2 2
```

your function should return the array `[2 2 1.5]`.
solutions:

```
def col_means(fn):
    f = open(fn,"r")
    L1 = readline(f)
    res = np.array(L1.split(),dtype="float")
    n = 1
    for L in f:
        n +=1
        res += np.array(L.split(),dtype="float")
    return res/n
```

Or:

```
def col_means(fn):
    f = open(fn,"r")
    my_list = []
    for line in f:
        my_list.append(line.split())
    a = np.array(my_list, dtype="float")
    return a.sum(axis=0)
```

Or (I don't expect anyone to use this one!):

```
def col_means(fn):
    f = open(fn,"r")
    return np.fromtext(f).sum(axis=0)
```

8. (12 points) What are the results of the following computations? (Assuming we have loaded the `numpy` library as `np`.) Some helpful numbers: $2^7 = 128$; $2^8 = 256$; $2^{2^{10}} \approx 10^{308}$; $2^{-53} \approx 10^{-16}$.

   a.

```
print(np.array(128, dtype="int8"))
```

   b.

```
print(np.array(128, dtype="uint8"))
```

   c.

```
print(1e350)
```

   d.

```
print(1e-400)
```

   e.

```
print(1e350 - 1e310)
```

   f.

```
print(1.0 + 1e-20)
```

solutions:

a. -1 (integer overflow)
b. 128 (no overflow)
c. `inf` (overflow)
d. 0 (underflow)
e. `nan` (can't subtract infinities)
f. 1.0 (catastrophic loss of precision)

No questions on this page.