Math 1MP3, final exam

23 April 2015

- Please write your name and student number on this test and on your answer sheet
- You have 120 minutes
- No external aids (calculator, textbook, notes)
- Please number your answers clearly on the test sheets

1. (15 points) Write a function `def letter_count(str):` that takes a alphabetic string `str` as input and returns a dictionary containing the counts of each letter (and punctuation, spaces, etc.), ignoring case. Remember that

   - `dict()` initializes an empty dictionary;
   - `d[z] = val` assigns a new value `val` to a key `z` or creates the key if it doesn't exist;
   - `z in d` tests whether the key `z` is defined in dictionary `d`;
   - `str.upper()` makes a string uppercase.

   For example, `letter_count("Squeamish ossifrage")` produces

```
## SOLUTION
def letter_count(str):
    d = dict()  ## or d = {}; empty dictionary
    for i in str:       ## loop over string
        w = i.upper()   ## convert to uppercase
        if not w in d:
            d[w] = 0    ## initialize new letters
        d[w] = d[w]+1   ## increment
    return(d)
import textwrap
print(textwrap.fill(format(letter_count("Squeamish ossifrage")),60))
```

```
## {'A': 2, ' ': 1, 'E': 2, 'G': 1, 'F': 1, 'I': 2, 'H': 1,
## 'M': 1, 'O': 1, 'Q': 1, 'S': 4, 'R': 1, 'U': 1}
```

2. (15 points) Write a function `def fizzbuzz(n):` that takes a positive integer `n` and returns a list containing the string form of these numbers from 1 to `n`, except:

- for multiples of 3, use `"Fizz"` instead of the number
- for multiples of 5 use `"Buzz"`
- for multiples of both 3 and 5 use `"FizzBuzz"`

   So `fizzbuzz(15)` would return:

```
## SOLUTION
def fizzbuzz(n):
    r = []    ## set up empty list
    for i in range(1,n+1):
        if i % 15==0:
            r.append("FizzBuzz")
        elif i % 5==0:
            r.append("Buzz")
        elif i % 3==0:
            r.append("Fizz")
        else:
            r.append(format(i))  ## add char representation of number
    return(r)

import textwrap
print(textwrap.fill(format(fizzbuzz(15)),60))

## ['1', '2', 'Fizz', '4', 'Buzz', 'Fizz', '7', '8', 'Fizz',
## 'Buzz', '11', 'Fizz', '13', '14', 'FizzBuzz']
```

Remember:

- `%` does modular division, e.g `14 % 7` is zero
- `format(x)` converts a number to a string
- you should probably check division by 3 *and* 5 ("fizzbuzz") before checking division by 3 ("fizz") or 5 ("buzz")

3. (15 points) With `sympy`, you can easily generate the first `D` digits of π by writing `pi.n(D)` (provided you have previously done `from sympy import pi`). Keeping in mind that you can use `format` to convert a number to a string, define a function `pi_digits(val,D)` to find the index of the beginning of first occurrence of a given digit string `val` (given as a string) in the first `D` digits of π. You can count the decimal point as a digit, so `pi_digits('141',1000)` is 2; `pi_digits('159',1000)` is 4; and `pi_digits('888',10)` is `None`.

```
## SOLUTION
from sympy import pi
def pi_digits(val,D):
    pd = format(pi.n(D))
    for i in range(0,D-len(val)):
        if pd[i:i+len(val)]==val:
            return(i)
    return(None)
## OR
## pd.index(val) is even easier!

print(pi_digits('141',1000))
print(pi_digits('159',1000))
print(pi_digits('888',10))
```

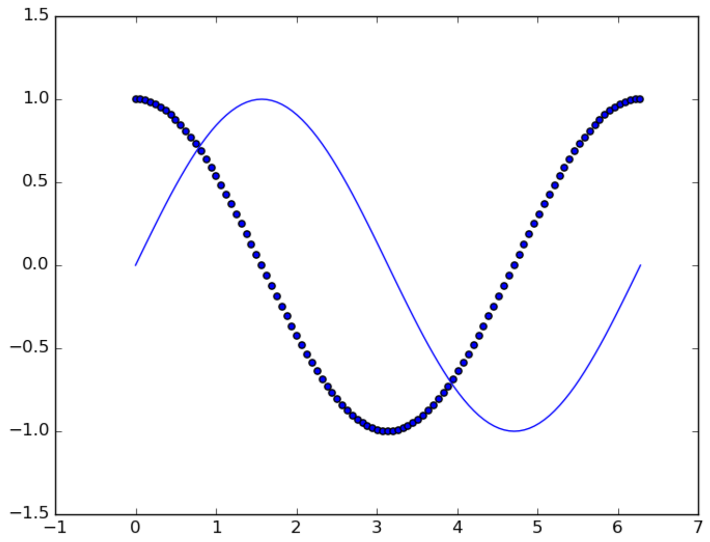
4. (15 points) Pretending temporarily that pennies still exist and are legal tender, and given an amount between 0 and 100 cents (inclusive), write a function `make_change(amount, coin_values)` that takes a *tuple* of coin denominations (always including at least 1, and specified in decreasing order) and returns a list giving the number of coins of each type required to make change. (Remember that `//` does integer division.) For example, `make_change(40,(10,1))` should return `[4,0]`; `make_change(73,(25,10,5,1))` should return `[2,2,0,3]`.

```
## SOLUTION
def make_change(amount, coin_values):
    res = []      ## initialize empty list
    for v in coin_values:
        res.append(amount // v)  ## add number of coins
        amount = amount % v      ## find remaining amount
    return res

print(make_change(40,(10,1)))
print(make_change(73,(25,10,5,1)))
```

4. (10 points) Draw an approximation of the picture that the following code produces. Include x- and y-axis limits.

```
## SOLUTION
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,2*np.pi,num=101)
y1 = np.sin(x)
y2 = np.cos(x)
f = plt.figure()
a = f.add_subplot(1,1,1)
a.plot(x,y1)
a.scatter(x,y2)
## f.show()
f.savefig("sincos.png")
```



6. (3 points for each item) Given a two-dimensional `numpy` array `a`, write a single line of code **using slicing or ranges** that will extract
* the element in the first row, second column * the 5th row * the last column * the last three elements in the last column

```
7. (3 points for each item) Consider the following code.

##   File "<string>", line 2
##     7. (3 points for each item) Consider the following code.
##                   ^
## SyntaxError: invalid syntax

from sympy import *
z1 = integrate(x**2,x)
solve(z1)
```

a. What sort of error does the following code produce, and why?
   A: *an error saying that the symbol is undefined (`NameError: name 'x' is not defined`), because `sympy` can't operate with an arbitrary symbol until it has been defined*
b. What line of code do you need to add to make it work? **A**: *`x = symbols('x')` or `x = Symbol('x')`*
c. What will it return once you have fixed it? **A**: *a list containing 0 (the solution of $x^3/3 == 0$); `[0]`*

8. (4 points for each item) Suppose the file `weather.csv` looks like this:

```
year,month,day,time,temp,wind,wind_dir,precip
2014,01,01,0800,-3,1,NW,0
2014,01,01,0900,-2,0,NA,0
...
2014,12,31,1100,-18,0,NA,1
```

(assume there is a row for every hour in 2014). Now we run the following `pandas` code:

```
import pandas as pd
import numpy as np
dd = pd.read_csv("weather.csv")
ddm = dd.groupby(dd.month)
ddm_avg = ddm.aggregate(np.mean)
## result is indexed by month value ... January=1
ff = ddm_avg.loc[2,"temp"]
res = dd[(dd.temp>ff) & (dd.wind>0)]
```

a. Describe `ff`: what data type is it (`int`, `float`, `str`, `tuple`, `list`, `Series`, `DataFrame`, ...) and what does it represent? **A**: *a `float` giving the average temperature in February*
b. Describe `res`: what data type and what does it represent? **A**: *a `DataFrame` containing the rows of the original data set corresponding to times when the temperature was above the February average and there was any wind*

9. (15 points) Given a tuple of values, write a function `lucky_sum(vals)` that returns the sum of all the values *before* the first value of 13. For example, `lucky_sum(())`, `lucky_sum((13,))`, or `lucky_sum((0,13,1))` would all return 0; `lucky_sum((1,2,3))` and `lucky_sum((1,2,3,13,2,5))` would both return 6.

```
## SOLUTION
def lucky_sum(vals):
    r = 0               ## initialize total
    for i in vals:
        if i==13:
            return(r)   ## return total immediately
        r += i          ## add current value to total
    return(r)           ## completed list; return total
```

```
print(lucky_sum(()))
print(lucky_sum((13,)))
print(lucky_sum((1,2,3)))
print(lucky_sum((1,2,3,13,5)))
```

```
## 0
## 0
## 6
## 6
```

10. (2 points, extra credit) what is the result of

```
from sympy import *
print(E*I*E*I*oo)
```

? **A**: `-Inf` *(mathematically this is* $e^2 \cdot i^2 \cdot \infty$*; the* i^2 *term makes the result negative rather than positive)*