1. **[14 points]** Warming up

   (a) **[2 points]** List at least three kinds of information that should be in a function docstring.

   (b) **[4 points]** Quick coding: A "double fifteen" domino set has 0-15 dots inclusive (i.e. including 0 and 15) on each side of the dominos. Write a function named `domino` that returns a random domino as a string with a "|" separating the sides. For example:

```
>>> domino()
0|4
>>> domino()
12|12
>>> domino()
15|0
```

   (c) **[8 points]** Quick coding: Write two functions, one using a `for` loop and the other using a `while` loop, to print the even numbers from 1 to 10 inclusive, one number per line.

2. [**12 points**] Slice and dice

   Given the variables `s` and `t` with the following values:

```
>>> s
'simon says'
>>> t
'touch your nose'
```

   Evaluate the following expressions and provide the resulting value.

   (a) `s[:6] + "didn't " + s[6:9]`

   (b) `s[:7].replace(" ", "'") + t[-5:]`

   (c) `((t.split()[2][:2].capitalize() + "! ") * 5).strip()`

   (d) `s + " " + t[1:6] + s[2::6] + t[-5:]`

3. [**8 points**] Function calls

   Consider the following Python code:

```
def bar(s):
    print(s)
    r = int(s)

def foo(s):
    r = int(s)
    for i in range(1, len(s)):
        bar(s[::i])
    return r

y = foo("2583")
```

   After execution what is the value of `y` and what, if anything, is printed in the shell?

4. **[14 points]** T/F

For each of the statements below state whether they are true or false.

_____ If `lt` is a non-empty list, `lt[0]` and `lt[:1]` evaluate to same type

_____ `6%3 + 5//2` evaluates to `2`

_____ If `s = "pizza pie"` then `s[2] == s[-2]` would evaluates to `True`

_____ `list("my pie".split())` evaluates to `['m', 'y', 'p', 'i', 'e']`

_____ The following two functions do the same thing:

```
def f(a, b):
    return not a and b or False

def g(a, b):
    return not (a and b)
```

_____ The following function would return the first position of a letter in a string:

```
def find_first(letter, mystring):
    i = mystring.find(letter)
    return mystring.find(letter, i)
```

_____ Some, but not all, Python strings are immutable

5. **[12 points]** We've got problems

   (a) **[6 points]** The function below has two integer parameters, `a` and `b`. The function works correctly, however, it is too long and it uses bad coding style. Rewrite the function in good style to be as concise as possible.

```
def could_be_better(a, b):
    if a > 5 and b >= -5:
        return True
    elif a == 6 and a == b:
        return False
    elif b < -5 or a <= 5:
        return False
    else:
        return True
```

   (b) **[6 points]** The following function was designed to count the distinct characters in the string parameter `sentence`. There are several problems with this code that will lead to either Python errors or incorrect output. Describe two different problems (which are not variations of the same issue).

```
def distinct(sentence):
    sentence = sorted(sentence)
    distinct = 0
    previous = ""
    for letter in range(len(sentence)):
        if letter != previous:
            distinct = 1
            previous = letter
    return distinct
```

6. [**16 points**] Coding

You want to identify the US zip code with the largest income gap. You are provided with text files for every zip code, one file per zip code, e.g. `05753.txt`, which list the income of every resident (including cents); one income per line. Every file has at least one entry. Write a Python function named `income_gap` that returns the filename containing the largest difference between the highest and lowest income. Your function should have a single parameter, a list of file names as strings, and return one of those filenames as a string. Your code need not be a single function, you can write other functions to be invoked from `income_gap`.

```
>>> income_gap(["05753.txt", "05443.txt", "05456.txt"])
05753.txt
```

An example file, e.g. "05753.txt":

Spring 2020 midterm will not include questions about files.
A problem like this would be re-written using lists, not files.

```
35000.30
73250.00
47500
```

7. [**10 points**] Turtle fun

```
from turtle import *

def shape(side):
    forward(side)
    right(180)
    forward(side)
    right(180)

side = 100
while side <= 300:
    shape(side)
    right(45)
    side += 50

done()
```

Draw the image that would be created by the above code and label your drawing with relevant dimensions, e.g. lengths of lines. Assume that the turtle is initially at the origin, facing right.