CS150 Fall 2019 – Midterm Solution

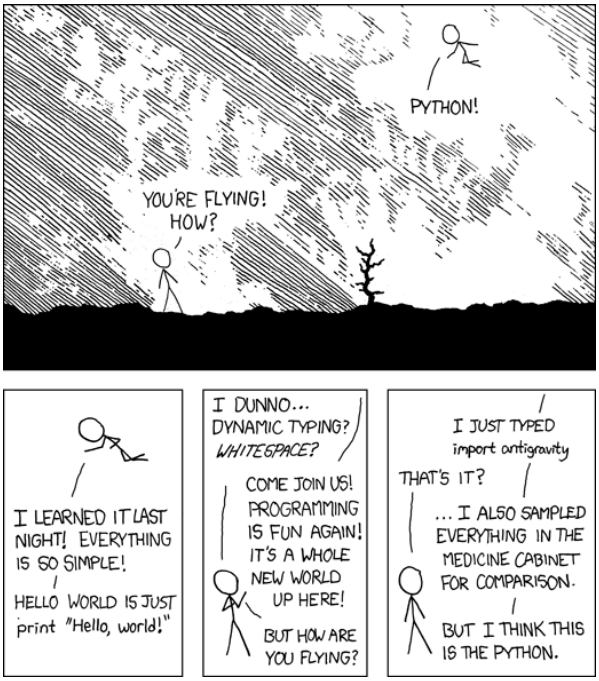
Name: _____ **Section: A / B**

Date: _____ **Start time:** _____ **End time:** _____

Honor Code:

Signature: _____

This exam is closed book, closed notes, closed computer, closed calculator, etc. You may only use (1) the midterm "cheat sheet" from the course page, and (2) a single double-sided letter sheet of notes. **You have 2.5 hours.** Read the problem descriptions carefully and write your answers clearly and *legibly* in the space provided. Circle or otherwise indicate your answer if it might not be easily identified. You may use extra sheets of paper, stapled to your exam, if you need more room, as long as the problem number is clearly labeled and your name is on the paper. If you attached extra sheets indicate on your main exam paper to look for the extra sheets for that problem.
You do need to include module imports (if relevant for your code), but do not need to include comments, docstrings or constants in your code.



Question	Points	Score
Warming up	16	
Slice and dice	12	
Function calls	8	
T/F	8	
We've got problems	16	
Coding	16	
Turtle fun	10	
Total:	86	

Question 1: Warming up [16 points]

(a) (3 points) Write an appropriate docstring for the following function, which computes the volume of a sphere.

```
def vol(radius):
    return 4/3*3.1415*radius**3
```

Solution:

```
"""
Compute the volume of a sphere

Args:
    radius: Radius of the sphere

Returns: Volume as a float
"""
```

(b) (5 points) A bingo ball set contains 75 balls. Each ball is labeled with a letter (one of "B", "I", "N", "G" or "O") and a number. The balls with the letter "B" are numbered 1-15 inclusive, "I" 16-30, "N" 31-45, "G" 46-60 and "O" 61-75. Write a function named `bingo` that returns a string with the label for a random bingo ball. For example:

```
>>> bingo()
'B3'
>>> bingo()
'N31'
>>> bingo()
'G52'
```

Solution:

```
from random import randint

def bingo():
    letter=randint(0,4)
    return "BINGO"[letter] + str(letter*15 + randint(1,15))
```

(c) (8 points) Quick coding: Write two functions, one using a `for` loop and the other using a `while` loop, to print the numbers from 1 to 100 inclusive, in ascending order, one number per line, which are multiples of either 3 or 5.

Solution:

There was some ambiguity around how to interpret "either", i.e. would that exclude numbers that are multiples of both 3 and 5. The intent was to interpret "or" in the logical sense, i.e. 3, 5 and 15 would all be printed. Both interpretations were accepted. To exclude 15, you could change the conditional expression to `(i % 3 == 0) != (1 % 5 == 0)`.

```
def forloop():
    for i in range(1, 101):
        if i % 3 == 0 or i % 5 == 0:
            print(i)
```

```
def whileloop():
    i = 1
    while i <= 100:
        if i % 3 == 0 or i % 5 == 0:
            print(i)
        i += 1
```

Question 2: Slice and dice [12 points]

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

```
>>> s
'many string'
>>> t
'copies in cs150'
```

Evaluate the following expressions and provide the resulting value in the boxes, one character per box. Fill in any unused boxes at the end of the string. Make sure upper case letters can be clearly distinguished from lower case letters.

(a) `t[2] + s[3] + s[4:] + s[5]`

Solution: `py strings`

(b) `t[::5] * 3`

Solution: `csccsccsc`

(c) `(s[0] + s[3:5]).upper() + t[10:]`

Solution: `MY cs150`

(d) `s[-1] + t[1] + " " + t[:8:5]`

Solution: `go cs`

Question 3: Function calls [8 points]

Consider the following Python code:

```
def bar(x, y):
    print(x + y)
    return x * 2

def foo(x):
    y = x
    for i in range(len(x)):
        y = y + bar(x[:i], y)
    return int(y)

y = foo("23")
```

(a) After execution the value of `y` is:

(a) **2322**

(b) What if anything is printed during execution?

Solution:

```
23
223
```

Question 4: T/F [8 points]

For each of the statements below state whether they are **T**rue or **F**alse.

(a) ___**F**___ `sorted(["alpha", "charlie", "Bravo", "Delta"])` would evaluate to `["alpha","Bravo","charlie","Delta"]`.

(b) ___**T**___ `8 % 3 + 3 * (8 // 3)` evaluates to 8

(c) ___**F**___ A while loop requires that we know the number of iterations to execute before the loop starts

(d) ___**T**___ The following two functions return identical results for all possible integer inputs

```
def f(a, b):
    return 2 * a + b + 1

def g(a, b):
    return (2 * a) + (b + 1)
```

(e) ___**T**___ After this code executes `x` has the value 12

```
x = 0
for i in "ab":
    for j in range(3):
        x += 1 + j
```

(f) ___**F**___ The following code will raise an error when the argument is a string, but execute successfully when the argument is a list of integers.

```
def mystery(arg):
    for i in range(len(arg)):
        print(arg[i] * i)
```

(g) ___**F**___ The following code will have an infinite loop for *all* possible user inputs

```
i = input("Enter your name: ")
while len(i) > 0:
    i = i[0:]
```

(h) ___**F**___ After this code executes `len(a)` will be 3

```
a = [1, 2, 3]
b = a.pop(1)
```

Question 5: We've got problems [16 points]

(a) The function below has two integer parameters, `a` and `b`. The function works as desired, however, it uses bad coding style. Rewrite the function to have identical behavior, but to be as concise as possible and implemented with good style.

```
def could_be_better(a, b):
    if b >= 8:
        return False
    elif b <= 5 and False:
        return True
    elif a == -10 or b > 5:
        return False
    else:
        return True
```

Solution:

```
def better(a, b):
    # Also accepted:
    # return not (a == -10 or b > 5)
    return a != -10 and b <= 5
```

(b) The following function was designed to return True if the elements of a list are sorted in ascending order (i.e. $x_i \leq x_{i+1}$) and False otherwise. An empty list is considered sorted. There are several problems with this code including at least one syntax error, one runtime error and one logical error (the code executes but produces incorrect results). Identify and describe one syntax error, one runtime error and one logical error (for three errors total), which are not variations of the same issue. Your problems should impact correctness, not just style. You do not need to fix the errors.

```
def issorted("seq"):
    for i in range(len(seq)):
        if seq[i] > seq[i+1]:
            return False
    return False
```

Solution:

1. Syntax Error: Function parameter `seq` specified as a string literal not an identifier.
2. Runtime Error: In the last iteration, when `i` is `len(seq)-1`, there will be an out of range of error because `seq[i+1]` is beyond the end of the list.
3. Logical Error: The final return statement should be `return True` because at that point we know the list is in sorted order (if not, the function would have already returned False).

Question 6: Coding [16 points]

Write a function named `closest_to` that takes two parameters, a filename as a string and an integer `target`, and returns the integer in the file closest to `target`. The file will have one integer per line, and is guaranteed to contain at least one integer. If multiple values are equally close to `target` you can return any of those values. For example if the file contained 10, 6, -3 and 20, and the target is 0, your function should return the integer -3. Your implementation can include other functions if that is helpful to you but doing so is not required.

Solution:

```
def closest_to(filename, target):
    with open(filename, "r") as file:
        values = []
        for line in file:
            values.append(int(line))

        closest = values[0]
        closest_dist = abs(closest - target)
        for value in values:
            dist = abs(value - target)
            if dist <= closest_dist:
                closest_dist = dist
                closest = value
        return closest
```

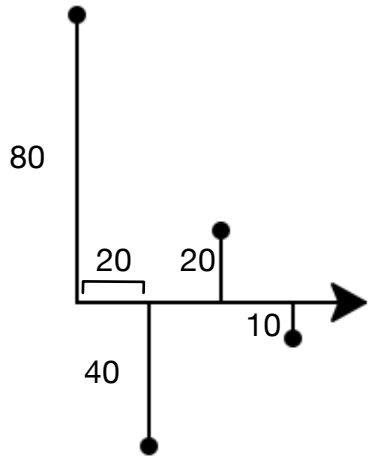
Question 7: Turtle fun [10 points]

```
from turtle import *

def shape(x):
    forward(x)
    dot(5)
    backward(x)

value=80
while abs(value) >= 10:
    if value > 0:
        left(90)
        shape(value)
        right(90)
    else:
        right(90)
        shape(-value)
        left(90)
    forward(20)
    value = value / -2
```

Using the grid below, draw the image that would be created by the above code. Label your grid so all relevant dimensions and angles are clear. Assume that the turtle is initially at the origin, facing right.



Page intentionally left blank.