CS150 Fall 2022 – Midterm

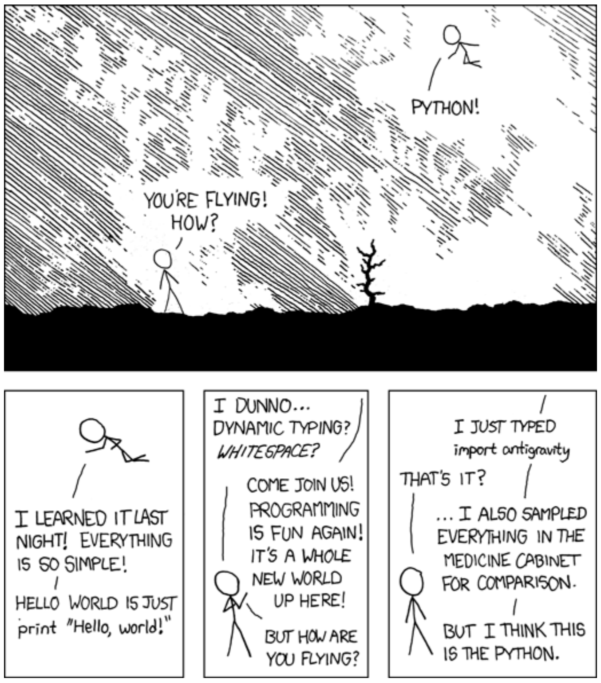
Name: ____ **Section: A / B**

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 of your own creation. **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.


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Question	Points	Score
Warming up	20	
Slice and dice	12	
Function calls	8	
T/F	8	
We've got problems	16	
Coding	16	
Turtle fun	10	
Total:	90	

Question 1: Warming up [20 points]

(a) (3 points) Semantics are similar across programming languages, and even non-programming contexts. The recipe instruction “simmer sauce until volume is reduced by half” is semantically most similar to which of the following Python constructs?

- ◯ An inline comment (e.g., a line starting with `#`)
- ◯ A list
- ◯ `for loop`
- ◯ `while` loop

Briefly explain your choice.

(b) (3 points) Which of the following best describes who benefits from good coding practices such as using informative function and parameter names?

- ◯ Someone using a function you wrote, but who does not inspect its implementation (i.e., the function body)
- ◯ Someone reading the function body
- ◯ Both of the above

Briefly explain your choice.

(c) (6 points) You just learned about Sicherman dice, a pair of 6-sided dice with non-standard numbers whose sums occur at the same frequencies as standard dice. One Sicherman dice has the sides 1, 2, 2, 3, 3, 4, and the other has sides 1, 3, 4, 5, 6, 8. Write a function named `sich` that simulates the sum of rolling both Sicherman dice. `sich` should have no parameters and return the sum of the simulated dice roll (an integer in the range 2-12 inclusive). Here are some sample calls and output.

```
>>> sich()
2
>>> sich()
7
```

(d) (8 points) Quick coding: Write two functions, one using a `for` loop and the other using a `while` loop, to print the numbers between 1 to 60 inclusive, in ascending order, one number per line, that are even *and* and a multiple of 3. Your functions should not have any parameters and should not return a value. For full credit your functions should be as concise and efficient as possible.

Question 2: Slice and dice [12 points]

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

```
s = "Returning To"
t = "Normal, Maybe?"
```

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

(a) `s[10:]+t[11:13]+t[1]+"k"+t[-1]`

(b) `s[::5] + s[::4]`

(c) `s[-12:-10] + t[:2].lower()`

(d) `(t[13:14] + t[6] + s[9])*2`

Question 3: Function calls [8 points]

Consider the following Python code:

```
def bar(x, i):
    i = 2*len(x)
    print(i)

def baz(x, i):
    return i-int(x[0])

def foo(s):
    r = 0
    for i in range(len(s)):
        y = bar(s[i:], i)
        r += baz(s[i:], i)
    return r

y = foo("4132")
```

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

(b) What if anything is printed during execution?

Question 4: T/F [8 points]

For each of the statements below state whether they are **T** (true) or **F** (false).

(a) ________ `(8 % 3) == 8 - (8 // 3) * 3` evaluates to `True`

(b) ________ If a program *just* containing the following code was executed with the green arrow in Thonny, the message `Execute?` will be printed a random number of times.

```
import random
def mystery():
    for i in range(random.randint(1, 10)):
        print("Execute?")
```

(c) ________ All `for` loops must execute at least one iteration

(d) ________ The expression `a == b and a != b` will always evaluate to `True` where `a` and `b` are any integers

(e) ________ The following function will return the length of the shortest string in *any* list of strings provided as the argument

```
def min_len(sequence):
    cur_min = len(sequence)
    for val in sequence:
        if len(val) < cur_min:
            cur_min = len(val)
    return cur_min
```

(f) ________ For the following implementation of `mystery`, all of these function calls will execute without an error: `mystery("01234")`, `mystery([5, 0, 1])`, `mystery(["1", "2"])`.

```
def mystery(arg):
    for i in arg:
        print(int(i) * 2)
```

(g) ________ The following loop will eventually terminate for *any* input provided by the user

```
i = input("Enter your name: ")
while len(i) < 10:
    i += "!"
```

(h) ________ For the following value of `a`, `a[1][0] >= a[2][1]` evaluates to `True`

```
a = [[1, 2, 3], [4, 2, 1], [5, 3, 6]]
```

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 (i.e., for all possible values of `a` and `b` return the same value), but to be as concise as possible and implemented with good style.

```
def could_be_better(a, b):
    if a <= 5:
        if b > a:
            return True
        else:
            return False
    else:
        if a < 0:
            return True
        elif b <= a:
            return False
        else:
            return True
```

(b) The following function was designed to squash multiple contiguous spaces in a string to just one space, i.e., replace runs of 2+ spaces with 1 space. The boxed examples show the intended behavior. There are 3 problems with this code. Identify and briefly describe (referencing the line numbers) i) one syntax error, ii) one runtime error (syntactically valid Python that generates an error when actually executed) and iii) one logical error (the code would execute to completion if the other errors are fixed but produces incorrect results), for three errors total. The errors should not be variations of the same issue and should impact correctness, not just style. You do not need to fix the errors.

```
1 def squash(s):
2     new_s = ""
3     for i in range(s):
4         if i = 0 or (s[i] != " " and s[i-1] != " "):
5             new_s += s[i]
6     return new_s
```

```
>>> squash(" a   b")
' a b'
>>> squash("  a  b  ")
' a b '
>>> squash(" ")
' '
>>> squash("")
''
```

i. Syntax Error:

ii. Runtime Error:

iii. Logical Error:

Question 6: Coding [16 points]

As part of a forestry study you have collected data on the coordinates of diseased trees, recorded as fractional numbers of meters. Due to limitations in your satellite sensing system, you have separate files with the x and y coordinates that are linked by line number (i.e., the i^{th} line in each file corresponds the same tree). The example files below describe two trees located at $(25.6, 16)$ and $(20, 8.2)$.

x.txt

```
25.6
20
```

y.txt

```
16
8.2
```

Write a function named `count_rect` that takes 6 parameters, the filenames of the tree's x and y coordinates, and the x, y coordinates of the lower left and upper right corners of a rectangle (e.g., `min_x, min_y, max_x, max_y`) and returns the number of trees inside or on the boundary of that rectangle. Assuming the data above was in files names `x.txt` and `y.txt`,

```
count_rect("x.txt","y.txt",20,10,30,20)
```

would return 1 (as there is one tree in the rectangle defined by $(20, 10)$ and $(30, 20)$). Your implementation can include other functions if that is helpful to you but doing so is not required.

Question 7: Turtle fun [10 points]

```
from turtle import *

def shape(x):
    for i in range(1,4):
        forward(x // i)
        left(90)

side = 120
while side > 10:
    shape(side)
    right(90)
    side = side // 2
```

Using the grid below, draw the image that would be created by the above code. Label your drawing if the dimensions would not be clear from the grid (not all drawing may occur on grid lines). Assume that the turtle is initially at the lower left corner, facing right. Assume each grid square is 5 pixels by 5 pixels.

Page intentionally left blank.