CS150 Fall 2023 – Midterm 1

Name: _____ **Section: A**

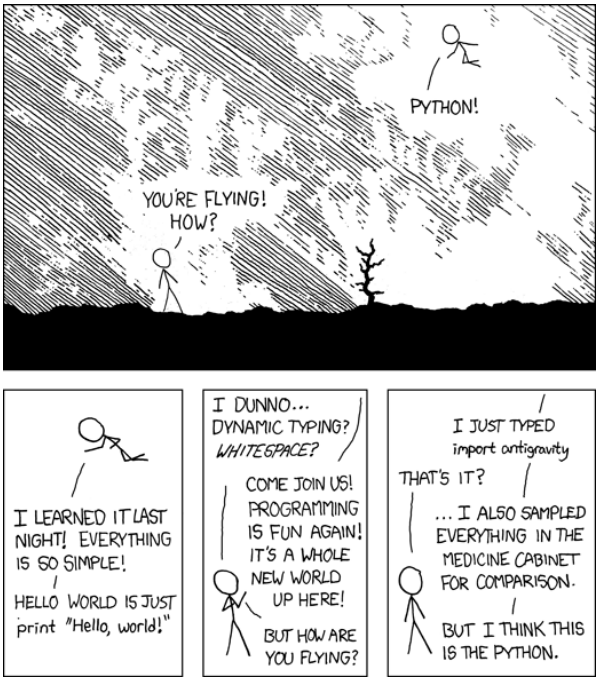
Date: _____ **Start time:** _____ **End time:** _____

Honor Code:

Signature: _____

This exam is closed book, closed notes, closed computer, closed calculator, etc. You may only use (1) the midterm 1 "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.



Question 1: Understanding the role of function scope

Consider the following Python code:

```
def baz(x, z):
    y = x + z
    return y * 2

def bar(x, y):
    z = x + y
    return z**2

def foo(y, z):
    x = y + z
    return x + 1

x = 1
y = 2
z = 3
a = foo(baz(1, 2), bar(3, 4))
```

(a) After the above is executed with the green arrow in Thonny, what are the values of `x`, `y`, `z` and `a`?

i. `x`: _____

ii. `y`: _____

iii. `z`: _____

iv. `a`: _____

(b) If the function definitions for `foo`, `bar` and `baz` were the same, but the code at the bottom was now

```
a = foo(baz(1, 2), bar(3, 4))
x = 1
y = 2
z = 3
```

what are the values of `x`, `y`, `z` and `a` if the code was executed with the green arrow in Thonny?

i. `x`: _____

ii. `y`: _____

iii. `z`: _____

iv. `a`: _____

Question 2: Writing functions with randomness

Write a function named `roll_dice`, that takes two parameters, `faces` and `num`, and prints the results of rolling `num` independent dice with `faces` number of faces. Dice with n faces are labeled with the integers 1 to n, inclusive. Your function should not return a value. Here are some sample calls and output.

```
>>> roll_dice(6, 2)
1
4
>>> roll_dice(20, 3)
1
17
20
```

Question 3: Writing functions with loops

Write two functions, one using a `for` loop and the other using a `while` loop, to print the numbers between 1 to 80 inclusive, in ascending order, one number per line, that are odd *and* and a multiple of 5. Your functions should not have any parameters and should not return a value. Your solution will be exclusively evaluated on correctness.

(a) Using a `for` loop

(b) Using a `while` loop

Question 4: Choosing appropriate loops

For each of the following tasks, indicate whether a `for` loop or a `while` loop would be more appropriate. Briefly explain your choice.

(a) Write a function that takes a list of integers as a parameter and returns the sum of the squares of the integers in the list.

◯ `for` loop ◯ `while` loop

(b) Decrypt a string that was encrypted by inserting differing numbers of random characters after each character of the original string (1 character after an "a", 2 after "b", 3 after "c", n after the n^{th} letter in the alphabet). For example, if the original string was `"abca"` the encrypted string could be `"akbzucwcdah"` (note the 1 character inserted after the `"a"`, 2 after the `"b"`, etc.)

◯ `for` loop ◯ `while` loop

(c) Write a function that prompts a user for a "yes" or "no" response, and keeps prompting them until they provide a valid response (either `yes` or `no`).

◯ `for` loop ◯ `while` loop

Question 5: Creating simpler equivalent conditionals

The function below has two integer parameters, `a` and `b`. The function works as desired, however, it is very verbose. 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.

```
def could_be_better(a, b):
    if a > b:
        if a >= 4 and b <= 6:
            return True
        else:
            return False
    else:
        if a >= 4:
            if b <= 6:
                return True
            else:
                return False
        else:
            return False
```

Question 6: Finding errors

The following function was designed to format strings with "American-style" phone numbers (XXX-XXX-XXXX) by inserting the appropriate dashes. If the number is 7 digits, it inserts one dash, if it is 10 digits it inserts two dashes, if it has a different length, it returns the number unmodified. The examples to the right show the intended behavior. There are 3 problems with this code. Report the line number of and briefly describe: 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. The implementation can assume all inputs are valid digits (i.e., that a user could provide an invalid phone number is not an error). You do not need to fix the errors.

```
1  def combine(part1, part2):
2      return part1 - part2
3
4  def format_phone(phone):
5      if len(phone) == 7:
6          return combine(phone[:3],phone[3:7])
7      elif len(phone) == 10:
8          return combine(phone[:3],combine(phone[3:6],phone[6:9]))
9      elif:
10         return phone
```

```
>>> format_phone("8675309")
'867-5309'
>>> format_phone("2126647665")
'212-664-7665'
>>> format_phone("5737")
'5737'
```

(a) Syntax error on line: _____

(b) Runtime error on line: _____

(c) Logical error on line: _____:

Question 7: Writing functions with sequences

Python strings have a very helpful `startswith` method that returns `True` if the string starts with some `prefix`. Unfortunately there isn't a similar method for lists. You would like to write one function that can be used with both sequence types (i.e., both strings and lists). Write a function named `startswith`, with two parameters, `seq` and `prefix`, which returns `True` if the sequence `seq` starts with the sequence `prefix` and `False` otherwise. Your function should work for any combination of string or list arguments, i.e., one of `seq` and `prefix` may be a string and the other a list. You can assume that `prefix` is the same length or shorter than `seq`. Some examples are shown below on the left and, for reference, some relevant string/list comparisons are shown on the right. Your solution will be exclusively evaluated on correctness.

```
>>> startswith("hello", "he")
True
>>> startswith(["h","e","l","l","o"], "he")
True
>>> startswith("hello", ["h","e"])
True
>>> startswith(["h","e","l","l","o"], ["h","e"])
True
>>> startswith(["h","e","l","l","o"], "hi")
False
>>> startswith([1, 2, 3], [1, 2])
True
```

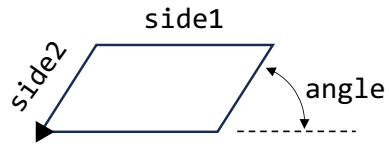
```
>>> "ab" == "ab"
True
>>> "ab" == ["a", "b"]
False
>>> ["a", "b"] == ["a", "b"]
True
```

Question 8: Utilizing turtle and other modules

Use the lines below to construct a function named `parallelogram` that takes three parameters, `side1`, `side2` and `angle`, and draws a parallelogram with the given side lengths and angle (like shown in the figure below). You may assume that the turtle (the pen) is initially at the triangle (lower left corner) and facing to the right. It should end at the triangle also pointing to the right (i.e., back at the initial starting position).

Write the line number and line of code in the rows below. The first line is done for you as an example. You may only use each of the scrambled lines once. Not all scrambled lines or rows may be needed. If there is an inconsistency between the line number and the code, the latter will be used as your answer.

```
1      t.forward(side1)
2      t.left(angle)
3      t.right(angle)
4  import turtle as t
5      t.left(angle)
6      t.forward(side2)
7      t.right(angle)
8      t.backward(side2)
9      t.left(180 - angle)
10     t.forward(side2)
11     t.forward(side1)
12 def parallelogram(side1, side2, angle):
13     t.left(180 - angle)
14     t.backward(side2)
```



Line number	Code
4	`import turtle as t`

Page intentionally left blank.