CMPT 120-D400
Sample Final Exam Questions
Fall 2022

Question 1

a) (10 marks) Write **a complete turtle graphics program** that draws a **straight line** of length 300 pixels. It's one continuous line with no gaps. The first 10 pixels of the line are **red**, the next 10 pixels are **green**, and the next 10 pixels are **blue**. This pattern repeats in the same way for each group of 10 pixels.

When done, the line will have this color pattern, where R=red, G=green, and B=blue:

R	G	B	R	G	B	R	G	B	…

Each box represents a line segment 10 pixels long, and the letter is the color (don't print the letters!).

Use `turtle.color(name)` to set the pen color, e.g. `turtle.color('red')`.

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Sample Solution

```
import turtle

def triple_rgb():
    turtle.color('red')
    turtle.forward(10)
    turtle.color('green')
    turtle.forward(10)
    turtle.color('blue')
    turtle.forward(10)

def long_rgb_line():
    for i in range(100):
        triple_rgb()

long_rgb_line()
```

b) (10 marks) Write a function called `num_check(a, b, c)` that works like this:

- If `a`, `b`, and `c` are all **different**, then it returns the string "all different"
- If `a`, `b`, and `c` are all the **same**, then it returns the string "all same"
- If neither of the above are true, then it returns the string "2 the same"

For example:

```
>>> num_check(4, 3, 10)
'all different'
>>> num_check(7, 7, 7)
'all same'
>>> num_check(4, 7, 4)
'2 the same'
```

Assume that the values passed into `num_check` are all numbers of the same type.

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Sample Solution

```
def num_check(a, b, c):
    if a == b and b == c:
        return 'all same'
    elif a != b and a != c and b != c:
        return 'all different'
    else:
        return '2 the same'
```

`

Question 2

a) (7 marks) Write a function called `is_triangle(a, b, c)` that returns `True` if `a`, `b`, and `c` can be the lengths of the sides of a valid triangle, and `False` otherwise. You can assume that `a`, `b`, and `c` are numbers of the same type.

a, b, and c form a valid triangle if they are each greater than 0, and if the expression $s(s - a)(s - b)(s - c)$ is also greater than 0. Here, s is the sum of a, b, and c, all divided by 2.

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Sample Solution

```
def is_triangle(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        return False
    else:
        s = (a + b + c) / 2
        m = s*(s-a)*(s-b)*(s-c)
        return m > 0
```

b) (13 marks) Write a **complete program** (it doesn't need to be in a function) that asks the user to enter the lengths of the three sides of a triangle, and then prints the area of that triangle. Assume the user always types in validly formatted Python floats.

If the numbers **are not** the lengths of the sides of a valid triangle, then print "not a valid triangle". You can use `is_triangle` in your answer.

If the numbers **are** the lengths of the sides of a valid triangle, then print "they form a triangle of area <*area*>", where <*area*> is replaced by the area of the triangle. Calculate the area using this formula:

$$Area = \sqrt{s(s-a)(s-b)(s-c)}$$

Here, *s* is the sum of *a*, *b*, and *c*, all divided by 2.

Hint Python's square root function is in the `math` module.

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Here are four sample runs:

```
What is the 1st number? 8
What is the 2nd number? 4
What is the 3rd number? 5
the triangle has an area of 8.181534085976786
```

```
What is the 1st number? 8
What is the 2nd number? 8
What is the 3rd number? 8
the triangle has an area of 27.712812921102035
it's equilateral!
```

```
What is the 1st number? 0
What is the 2nd number? 9
What is the 3rd number? 5
not a valid triangle
```

```
What is the 1st number? 3
What is the 2nd number? 1
What is the 3rd number? 2
not a valid triangle
```

`

Sample Solution

```
import math

a = float(input('What is the 1st number? '))
b = float(input('What is the 2nd number? '))
c = float(input('What is the 3rd number? '))

if not is_triangle(a, b, c):
    print('not a valid triangle')
else:
    s = (a + b + c) / 2
    area = math.sqrt(s*(s-a)*(s-b)*(s-c))
    print(f'the triangle has an area of {area}')
    if a == b and b == c:
        print("it's equilateral!")
```

`

Question 3

A **vote dictionary** stores the names of candidates in an election, and how many votes they received. For example:

```
{'Rick': 15, 'Morty': 3, 'Summer': 10, 'spoiled': 10}
```

This says Rick got 15 votes, Morty got 3, Summer got 10, and 10 other votes were spoiled (they were not filled in, or couldn't be read, etc.).

The key `'spoiled'` may, or may not, appear in a vote dictionary. For example, these are both *valid* vote dictionaries:

```
{'Alan': 10, 'Bo': 12, 'Cass': 13, 'spoiled': 200}
{'Alan': 10, 'Bo': 12, 'Cass': 13}
```

In the following questions assume that the passed-in variable `vote` is always a valid vote dictionary with at least 3 key/value pairs.

a) (5 marks) Write a function called `get_num_votes_cast(votes)` that returns the total number of votes for all candidates. If `'spoiled'` is a key in votes, then its value is **not** included in the total.

For example:

```
>>> get_num_votes_cast({'Rick': 15, 'Morty': 3, 'Summer': 10,
                        'spoiled': 10})
28

>>> get_num_votes_cast({'Rick': 15, 'Morty': 3, 'Summer': 10})
28
```

Make sure `votes` is **not** modified: it should have exactly the same contents after calling the function as before.

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Sample solution

```
def get_num_votes_cast(votes):
    total = sum(votes.values())
    if 'spoiled' in votes:
        return total - votes['spoiled']
    else:
        return total
```

`

b) (10 marks) Write a function called `get_winner_name(votes)` that returns the name of the candidate who received the most votes. For simplicity, assume that all candidates received a *different* number of votes.

If the special key `'spoiled'` is in `votes`, then ignore it and do **never** return it as the winner.

Make sure `votes` is **not** modified: it should have exactly the same contents after calling the function as before.

For example:

```
>>> get_winner_name({'Rick': 15, 'Morty': 3, 'Summer': 10})
'Rick'

>>> get_winner_name({'Alan': 10, 'Bo': 12, 'Cass': 13, 'spoiled': 200})
'Cass'
```

Use correct syntax, correct and consistent indentation, and general good Python style in your answer. Your code should **not** do any unnecessary work.

Sample solutions

```
def get_winner_name(votes):
    result = []
    for name in votes:
        if name != 'spoiled':
            result.append([votes[name], name])
    result.sort()
    return result[-1][1]
```

```
def get_winner_name(votes):
    result = []
    for name in votes:
        if name != 'spoiled':
            result.append([votes[name], name])
    result.sort()
    result.reverse()
    return result[0][1]
```

```
def get_winner_name(votes):
    result = []
    for name in votes:
        if name != 'spoiled':
            result.append([votes[name], name])
    winner = max(result)
    return winner[1]
```