CS 1: Introduction to Computer Programming — Fall 2025

Recitation 02: Quiz Review

More Treasure (topics: Lists, for loops)

In this question, we will explore lists and for loops.

Treasure Map Part 1 and Part 2

Note whether the for loop would print "Found it!" for each respective `treasure_map`.

```
1  TREASURE = "gold!"
2  treasure_map1 = ["", "", "", TREASURE, ""]
3  treasure_map2 = [
4    [
5      [],
6      []
7    ],
8    [
9      [],
10     TREASURE
11   ],
12   []
13 ]
```

Version 1

```
14 for elem in treasure_map:
15    if elem == TREASURE:
16        print("Found it!")
```

treasure_map1: Yes / No
treasure_map2: Yes / No

Version 2

```
14 for elem in treasure_map:
15    if TREASURE in elem:
16        print("Found it!")
```

treasure_map1: Yes / No
treasure_map2: Yes / No

Treasure Map Part 3

Now, rewrite the for loop using `range` and `len` instead.

Bool-ied (topics: boolean operators)

Note whether each `print` statement outputs `True`, `False`, or makes an error.

Code

```
1  x: int = 3
2  print(x < 7 and x > 5)
```

Code

```
1  x: int = 3
2  print(x < 7 or x > 5)
```

Code

```
1  lst: list[int] = [1, 2, 3]
2  print(len(lst) > 0 and lst[0] > 0)
```

Code

```
1  lst: list[int] = []
2  print(len(lst) > 0 and lst[0] > 0)
```

Looking up things you don't know! (topics: looking up documentation, using the syntax handout)

Write a function `parse_log(log)` that assumes the format "DATE.TIME.MESSAGE" and returns a list of [date, time, message]. Below is an example:

Input/Output Example

```
1  print(parse_log("2025-04-07,13:45:22,User logged in, password denied."))
```

```
>> ["2025-04-7","13:45:22","User logged in, password denied."]
```

```
1  def parse_log(log: str) -> list[str]:
2      return
```

Wavelength (topics covered: functions, scope, reading docstrings)

Wavelength is a party game where players guess a hidden spot (from 0-10) based on a teammate's clue. The closer the guess, the more points they earn.

score_wavelength

Start by implementing `score_wavelength`.

Hint: You can use the built in function abs, which takes in an integer and returns the absolute value of the integer.

```
1  def score_wavelength(hidden_spot: int, guess: int) -> int:
2      """
3      Scores a game of wavelength
4
5      Args:
6      hidden_spot (int): Location of the hidden spot.
7      guess (int): The player's guess.
8
9      Returns:
10     int: The number of points a player earns based on their guess and the hidden spot,
11          according to the following rules below.
12
13     Rules:
14     - If the guess is not between 0 and 10 (inclusive), or is 3 or more away from the hidden
15       spot: 0 points
16     - If the guess is 2 away from the hidden spot: 1 point
17     - If the guess is 1 away from the hidden spot: 2 points
18     - If the guess is exactly at the hidden spot: 3 points
19     """
20
21         delta: int = abs(                          )
22         if                                                :
23             return 0
24         elif                :
25             return 1
26         elif                :
27             return 2
28     else:
29         return 3
```

play_wavelength

Now, implement play_wavelength, which takes in an integer hidden_spot representing the location of the hidden spot, solicits a guess from the player using input, and returns the score of the player.

```
def play_wavelength(hidden_spot: int) -> int:
    """
    Plays a game of wavelength by prompting the user for a guess.

    Args:
    hidden_spot (int): The location of the hidden spot.

    Returns:
    int: The score based on the user's input and the hidden spot.
    """
    player_guess: str = input("Make your guess!") # Assume the player inputs a number

    return [                                    ]
```

What happens?

Describe what would happen if we ran the following code (in the same file that score_wavelength and play_wavelength are implemented):

Code

```
score: int = play_wavelength(5)
print("With hidden spot " + str(hidden_spot) + ", the score was: " + str(score))
```

Valid Password (topic: looping over strings)

Let's build a password checker. Start by implementing check_password by following its docstring.

Fill in the respective type annotations as well!

```
def check_password(password: [    ]) -> [    ]:
    """
    Checks whether a password contains at least 3 vowels (A, E, I, O, U), case-insensitive.

    Args:
       password (???) The password string to validate.

    Returns:
      ???: True if the password has 3 or more vowels, False otherwise.
    """
    count: int = 0

    for char in [                ]:
        if [                ]:
            count += 1

    return [                ]
```