# CS 1: Introduction to Computer Programming

## Recitation 2: Conditionals, Functions, and Reading Documentation Solutions

In this syntax recitation, we'll finish looking at `for` loops, writing our own user-defined functions, and practice reading documentation!

# Problem Solutions

## 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`.

```
TREASURE: final[str] = "gold!"
treasure_map1: list[str] = ["", "", "", TREASURE, ""]
treasure_map2: list[list[str]] = [
    [
        [],
        []
    ],
    [
        [],
        TREASURE
    ],
    []
]
```

Version 1

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

`treasure_map1`: Yes / No
`treasure_map2`: Yes / No

Version 2

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

`treasure_map1`: Yes / No
`treasure_map2`: Yes / No

### Solution:

In Version 1, the for loop would print `"Found it!"` for `treasure_map1`, but not `treasure_map2`.

In Version 2, the for loop would print `"Found it!"` for both treasure maps. Remember that `"gold!" in "gold!"` evaluates to `True` since we effectively look to see if "gold!" is a substring in "gold!" (which is True).

### Treasure Map Part 3

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

### Solution:

```
    for i in range(len(treasure_map)):
        ...
```

Then, replace all instances of `elem` with `treasure_map[i]`.

## 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)
```

### Solution:

`False`.

**Code**

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

### Solution:

`True`.

**Code**

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

### Solution:

`True`.

**Code**

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

### Solution:

`True`.

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

```
>> ["2025-04-07","13:45:22","User logged in, password denied."]
```

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

## 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 that integer.

```
def score_wavelength(hidden_spot: int, guess: int) -> int:
    """
    Scores a game of wavelength.

    Args:
    hidden_spot (int): Location of the hidden spot.
    guess (int): The player's guess.

    Returns:
    int: The number of points a player earns based on their guess and the hidden spot, according to
        the following rules below.

    Rules:
    - If the guess is not between 0 and 10 (inclusive), or is 3 or more away from the hidden spot: 0
        points
    - If the guess is 2 away from the hidden spot: 1 point
    - If the guess is 1 away from the hidden spot: 2 points
    - If the guess is exactly at the hidden spot: 3 points
    """
    delta: int = abs( player_guess - hidden_spot )
    if player_guess < 0 or player_guess > 10 or delta >= 3 :
        return 0
    elif delta == 2 :
        return 1
    elif delta == 1 :
        return 2
    else:
        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 that the player inputs a number.
    return score_wavelength(hidden_spot, int(player_guess))
```

### 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))
```

### Solution:

This would error, since `hidden_spot` is only defined in the scope of the `play_wavelength` function. If we try to access it in the top level, we'll see that it isn't defined.