CS 1: Introduction to Computer Programming

Recitation 3: Nested for loops, while loops, type annotations Solutions

In today's syntax recitation, we'll cover use cases for while loops, practice using type annotations, and do some more practice with for loops!

Problem Solutions

Password Suggester (topics: Looping over strings, while loops)

Let's build a password suggester together. Start by implementing `check_password` by following its docstring. Fill in the respective type annotations as well!

Valid Password (topic: looping over strings)

```
def check_password(password: [ str ]) -> [ bool ]:
    """
    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 [ password.upper() ]:
        if [ char in "AEIOU" ]:
            count += 1
    return [ count >= 3 ]
```

Solution:

Alternatively, you can do `char in "AEIOUaeiou"`, or replace "AEIOU" with ["A", "E", "I", "O", "U"] or set(["A", "E", "I", "O", "U"]).

Password Suggester (topic: while loops)

```
def prompt_for_passwords(num_passwords: int) -> None:
"""
Queries the user until we have num_passwords valid passwords.
A password is valid if it contains at least 3 vowels.

Args:
    num_passwords (int): The number of valid passwords to collect.
"""
valid_count: int = 0
while [ valid_count < num_passwords ]:
    pwd: str = input("Enter a password: ")
    if [ check_password(pwd) ]:
        print("Valid password:", pwd)
```

```
       valid_count += 1
    else:
       print("Invalid password. Try again.")
```

Loop-de-loop (topics: for loops, tuples, sets)

Fill in the relevant type annotations for each block of code. Note what each block of code outputs. If there is an error, write "Error".

Code

```
lst: list[str] = ["hi", "hello", "hey"]
for elem in lst:
   if len(elem) > 2:
      print(elem)
```

Solution:

```
hello
hey
```

Code

```
lst: list[str] = ["hi", "hello", "hey"]
for elem in lst:
   elem = "LOL"
print(lst[0])
```

Solution:

```
hi
```

Code

```
lst: list[str] = ["hi", "hello", "hey"]
for i in range(len(lst)):
   lst[i] = "LOL"
print(lst[0])
```

Solution:

```
LOL
```

Code

```
tup: tuple[str, str, str] = ("hi", "hello", "hey")
tup[0] = "LOL"
print(tup[0])
```

Solution:

`Error`. Cannot change values in a tuple.

It's like a maze...

Fill in the line of code necessary to print out all elements in the order specified.

Code

```
lst: list[list[int]] = [
                [1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]
                ]
#Should print out 1, 2, 3, 4, 5, 6, 7, 8, 9
for y in range(len(lst)):
   for x in range(len(lst)):
      print(lst [y][x] )
```

Code

```
lst: list[list[int]] = [
                [1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]
                ]
#Should print out 1, 4, 7, 2, 5, 8, 3, 6, 9
for y in range(len(lst)):
   for x in range(len(lst)):
      print(lst [x][y] )
```

Code

```
lst: list[list[int]] = [
                [1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]
                ]
#Should print out 1, 5, 9
for x in range(len(lst)):
   print(lst [x][x] )
```

Code

```
lst: list[list[int]] = [
                [1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]
                ]
#Should print out 3, 5, 7
for x in range(len(lst)):
   print(lst [x][len(lst[x]) - x - 1] )
```

Solution:

Since this is a square board, len(lst) would also work.