CS 1: Introduction to Computer Programming

Recitation 1: Introduction to Variables and Loops Solutions

Welcome to your first CS 1 "syntax recitation" of the term! These sessions aim to align with the material covered in lecture to help you identify common mistakes we've seen students make in the past (and programming) and lead you through some practice exercises with help!

Problem Solutions

WWPD?

Answer the questions about each code block.

Code

```
1  a = 2025
2  a = "Hello "
3  print(a)
```

(a) When the code is finished running, what is the type of `a`?

Solution:

`str`

(b) Does this code execute without any errors?

Solution:

Yes.

(c) If so, what does it output? If not, fix the error.

Solution:

`Hello`

Code

```
1  a = 2025
2  a = "Hello "
3  b = a
4  a = "there."
5  print(b)
```

(a) When the code is finished running, what is the type of `a`?

Solution:

`str`

(b) When the code is finished running, what is the type of `b`?

Solution:

`str`

(c) Does this code execute without any errors?

Solution:

Yes.

(d) If so, what does it output? If not, fix the error.

Solution:

```
Hello
```

Code

```
1  a = 2025
2  a = "Hello "
3  b = a
4  a = "there."
5  print(b + a)
```

(a) When the code is finished running, what is the type of `a`?

Solution:

`str`

(b) When the code is finished running, what is the type of `b`?

Solution:

`str`

(c) Does this code execute without any errors?

Solution:

Yes.

(d) If so, what does it output? If not, fix the error.

Solution:

`Hello there.`

Code

```
1  a = 2025
2  a = "Hello "
3  b = a
4  a = "there."
5  a = 5.0
6  print(a + b)
```

(a) When the code is finished running, what is the type of `a`?

Solution:

`float`

(b) When the code is finished running, what is the type of `b`?

Solution:

`str`

(c) Does this code execute without any errors?

Solution:

No

(d) If so, what does it output? If not, fix the error.

Solution:

Input/Output Example

```
1 a = 2025
2 a = "Hello "
3 b = a
4 a = "there."
5 a = 5.0
6 print(str(a) + b) #You can also put quotes around 5.0 in line 5
```

```
>> 5.0Hello
```

Code

```
1  a = 2025
2  a = "Hello "
3  b = a
4  a = "there."
5  a = 5.0
6  print(a - b)
```

(a) When the code is finished running, what is the type of `a`?

Solution:

`float`

(b) When the code is finished running, what is the type of `b`?

Solution:

`str`

(c) Does this code execute without any errors?

Solution:

No.

(d) If so, what does it output? If not, fix the error.

Solution:

Fairly difficult to fix without removing lines of code while keeping the `-` operator. Variables of type `str` cannot be operands to `-` in python.

Indexing strings, lists, and tuples

In this question, you'll work with a few different data types and common edge cases.

Finding Treasure

Gold!

Print the variable TREASURE by accessing it from the treasure_maps.

```
1    TREASURE = "gold!"
2    treasure_map1 = ["", "", "", TREASURE, ""]
3    print(treasure_map1[3])
4    treasure_map2 = [
5       [
6          [],
7          []
8       ],
9       [
10         [],
11         TREASURE
12      ],
13      []
14   ]
15   print(treasure_map2[1][1])
16   treasure_map3 = [
17      [
18         [],
19         []
20      ],
21      [
22         [],
23         [TREASURE]
24      ],
25      []
26   ]
27   print(treasure_map3[1][1][0])
28   treasure_map4 = (
29      (0,1),
30      (0,1,0),
31      (
32         1,
33         0,
34         (1,0),
35         (1,0,1,0,TREASURE)
36      )
37   )
38   print(treasure_map4[2][3][4])
```

More Treasure

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!")
```

Version 2

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

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.

```
1     for i in range(len(treasure_map)):
2        ...
```

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