Python Variables

Each exercise presents a sequence of Python commands. Think carefully about each, predict what will be printed, and answer any questions.

You can later check your predictions with a Python REPL at https://pyodide.org/en/stable/console-v2.html

1. Basics

Variables hold values, and we can use them in place of those values.

```
name = "Bregman"
age = 31
print(name)
print(age + 5)
```

What will be printed?

What will the value of `age` be after all the commands have been executed?

2. Reassignment

Re-assigning a variable overwrites the old value. Python doesn't "remember" the previous value once it's gone.

```
score = 100
score = 150
score = 50
print(score)
```

What will be printed?

3. Accessing Uninitialized Variables

Until you create a variable by assigning it a value, it does not exist.

```
print(room)
room = 204
print(room)
```

What will be printed?

4. Relative Updates

Variables can appear on both sides of the equals sign (the **assignment operator**). Python evalutes the right side *first*, then updates the value of the variable on the left.

```
count = 10
count = count + 5
count = count * 2
print(count)
```

What will be printed?

What would be printed if you swapped the order of the second and third commands?

5. Multiple Variables

Each variable keeps track of its own value.

```
apples = 5
bananas = 12
total_fruits = apples + bananas
apples = 10
print(total_fruit)
```

What will be printed?

6. Linked Variables?

Does assigning one variable to another *link* them?

```
f = 100
g = f
f = 42
print(g)
```

What will be printed?

Does `g = f` **link** `g` and `f`? What evidence supports your answer?

7. Chain Link?

Here's an example of "chained assignment". Can multiple variables point to the exact same value at the same time?

```
j = k = 2
j = j + 1
print(j)
print(k)
```

What will be printed?

Explain why.

8. "Seat Swap" (aka Tuple Unpacking)

Here's an example of "parallel assignment". This is a very "Pythonic" way to assign multiple values to multiple variables.

```
p, q = "pint", "quart"
p, q = q, p
print(p)
```

What will be printed?

Explain why.