Pythonの変数(名札)にはアドレスは書かれていない

同じ値を指していても新しい値を指定されると別の値を指すようになる

インプレースでの変更とは

Pythonの変数 は、C・Java・C# などとは全く違う仕組み

このお題を念頭に以下のコードを考えてみてください。

print関数で何が表示されるでしょうか。

ヒントだけ付けました。

```
a = [1, 2, 3]
# 変数b はアドレスをもらったのではなく、同じ値に名札を付けただけである
b = a
print(b)
# 変数a は「新しい値」に名札を付けた
# アドレスとは無縁である
a = [4, 5, 6]
print(b)

x = ['a', 'b', 'c']
y = x
print(y)
# インプレースの変更になる
y.append('d')
# インプレースの変更では、同じ値が変更される
# インプレース(in place)：[副詞句]所定の位置に
# append()メソッドは、インプレースに変更させることを知っていることが重要！
# これは理屈で導くべきではなく、仕様を知っていることが大事です。
# つまり、変数x は、依然として同じものを指している
print(x)
```

上記のコードの答

コードで説明した方が分かりやすいと思ったのでそうしました。 コメントに答を書きました。

```
a = [1, 2, 3]
# 変数b はアドレスをもらったのではなく、同じ値に名札を付けただけである
b = a
print(b)  # [1, 2, 3]
# 変数a は「新しい値」に名札を付けた
# アドレスとは無縁である
a = [4, 5, 6]
print(b)  # [1, 2, 3]（変数にはアドレスが書かれていない証拠である）

x = ['a', 'b', 'c']
y = x
print(y)  # ['a', 'b', 'c']
# インプレースの変更になる
y.append('d')
# インプレースの変更では、同じ値が変更される
# インプレース(in place)：[副詞句]所定の位置に
# append()メソッドは、インプレースに変更させることを知っていることが重要！
# これは理屈で導くべきではなく、仕様を知っていることが大事です。
# つまり、変数x は、依然として、変数yと、同じものを指している
print(x)  # ['a', 'b', 'c', 'd']
```

END