Avoiding Surprises with Immutable Variables

Gerald Britton
IT SOLUTIONS DESIGNER

@GeraldBritton www.linkedin.com/in/geraldbritton

Overview

Immutable variables cannot change

Two rules:

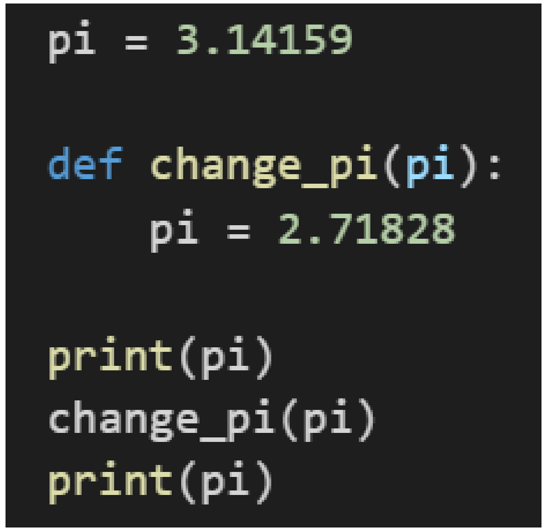
- Local variables do not change
  - `locals()`
- Global variables can only change references
  - `globals()`

Mutable variables can be troublemakers

$$\pi = 3.14159$$

$$\cancel{\pi = 3.14159}$$

Trouble in Mutable Town

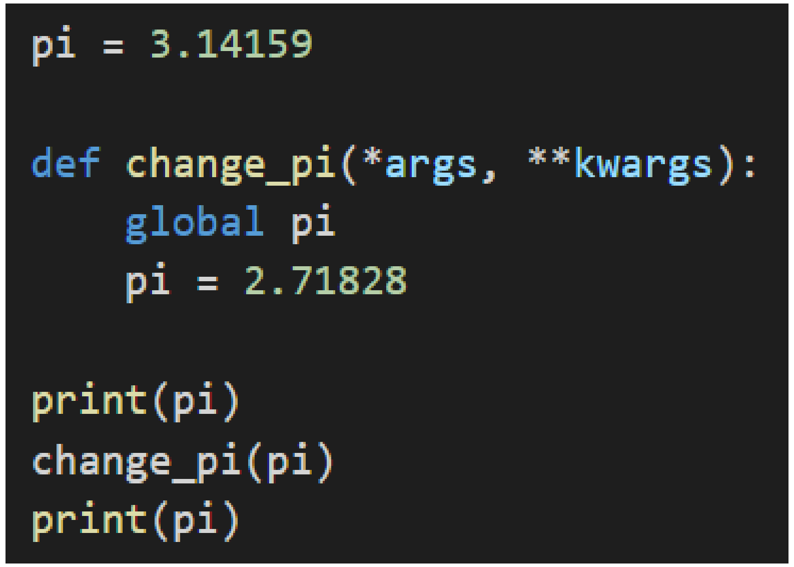


```
pi = 3.14159

def change_pi(pi):
    pi = 2.71828

print(pi)
change_pi(pi)
print(pi)
```

Try to change pi



```
pi = 3.14159

def change_pi(*args, **kwargs):
    global pi
    pi = 2.71828

print(pi)
change_pi(pi)
print(pi)
```

Really change pi!

https://en.wikipedia.org/wiki/Indiana_Pi_Bill

Demo

An order has a list of order items

Notify customer if item is backordered

Mark an order item as backordered

Demo

Notify customer if item is backordered

Mark an order item as backordered

Find an order item and set it to backordered

Immutably!

Summary

Immutability in Python

Local variables never change

Global variables can change references

Added backordered items notification

Added ability to mark items backordered

Preserved immutability

- Immutable meta class

Switched from lists to tuples

Created a new, pure, higher-order function