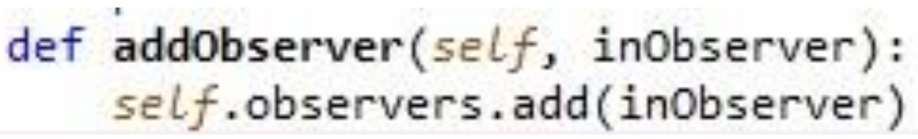
CSE 3342

Seung Ki Lee

HW7 : Python PUBSUB

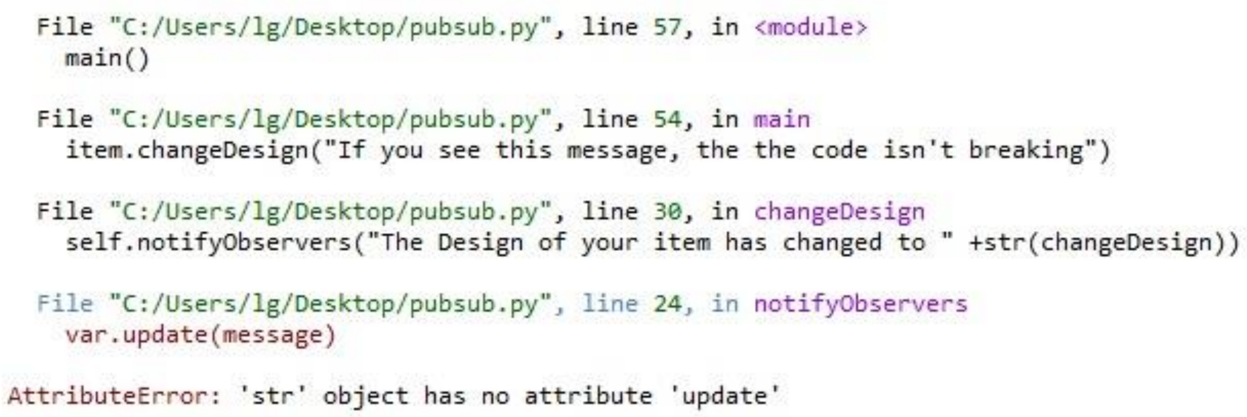
Part A) Breaking Code

Due to the fact that python is a dynamically typed language, it will accept anything into the set even if it's not truly and "Observer". That being said, the code will break if the input has no function 'update' as its attribute. So the addObserver code is:



```
def addObserver(self, inObserver):
    self.observers.add(inObserver)
```

The result is

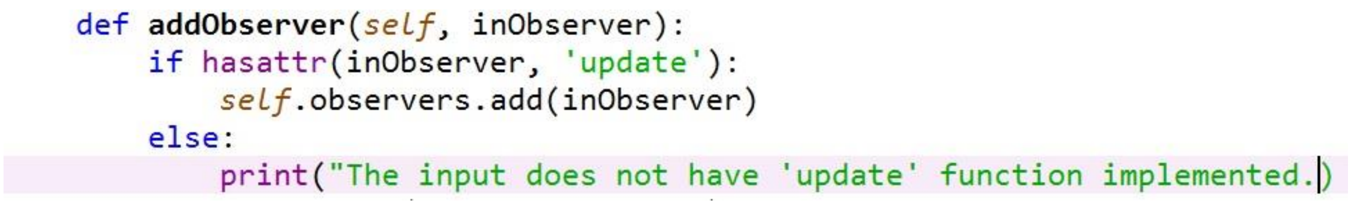


```
  File "C:/Users/lg/Desktop/pubsub.py", line 57, in <module>
    main()

  File "C:/Users/lg/Desktop/pubsub.py", line 54, in main
    item.changeDesign("If you see this message, the the code isn't breaking")

  File "C:/Users/lg/Desktop/pubsub.py", line 30, in changeDesign
    self.notifyObservers("The Design of your item has changed to " +str(changeDesign))

  File "C:/Users/lg/Desktop/pubsub.py", line 24, in notifyObservers
    var.update(message)

AttributeError: 'str' object has no attribute 'update'
```

It states that 'str' object has no attribute. To fix this problem, I've added a pseudo-type checking.

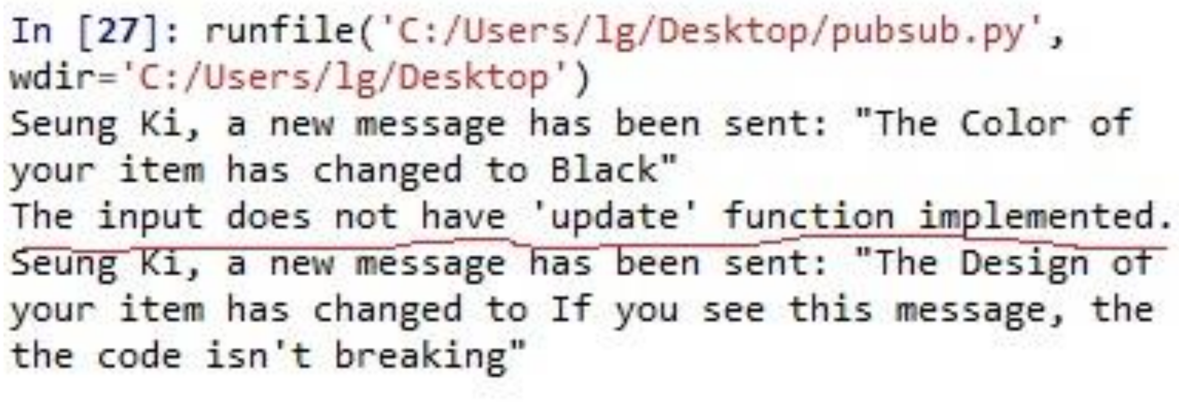
Part B) Refactored Code

The point is, the "Observer" should be anything that had function 'update' implemented. So the new code has one conditional:



```
def addObserver(self, inObserver):
    if hasattr(inObserver, 'update'):
        self.observers.add(inObserver)
    else:
        print("The input does not have 'update' function implemented.)
```

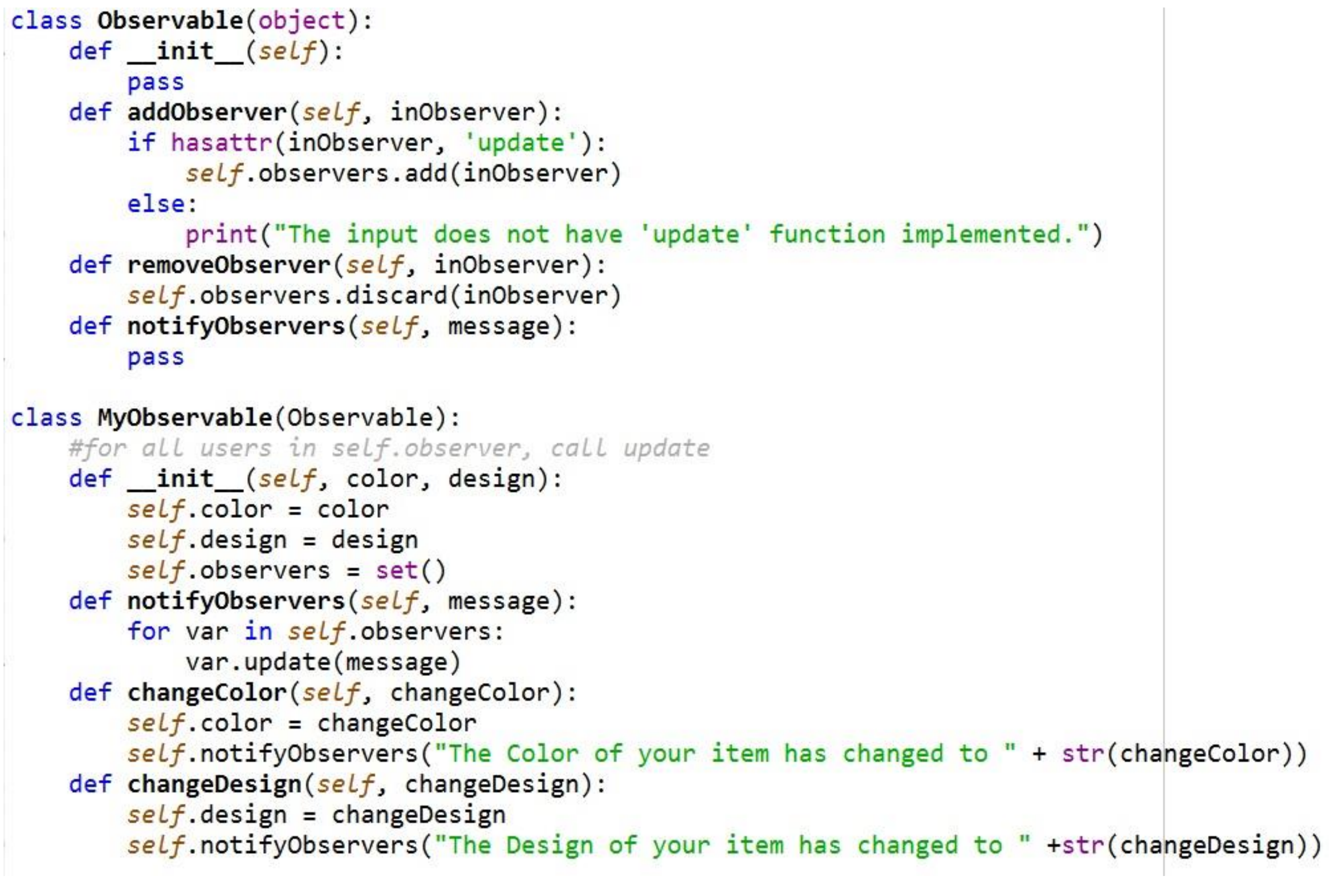
Then, the output comes out



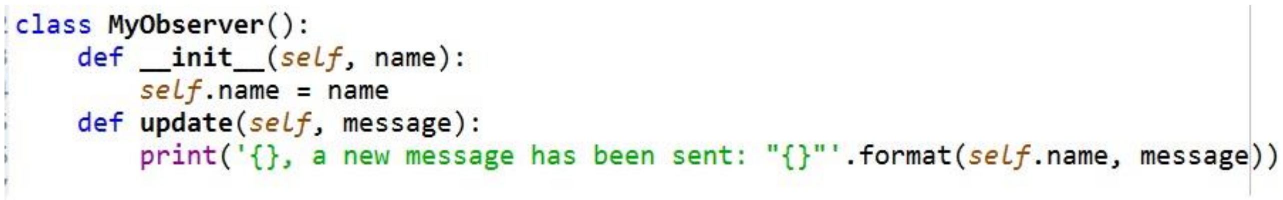
```
In [27]: runfile('C:/Users/lg/Desktop/pubsub.py',
wdir='C:/Users/lg/Desktop')
Seung Ki, a new message has been sent: "The Color of
your item has changed to Black"
The input does not have 'update' function implemented.
Seung Ki, a new message has been sent: "The Design of
your item has changed to If you see this message, the
the code isn't breaking"
```

With proper error message.

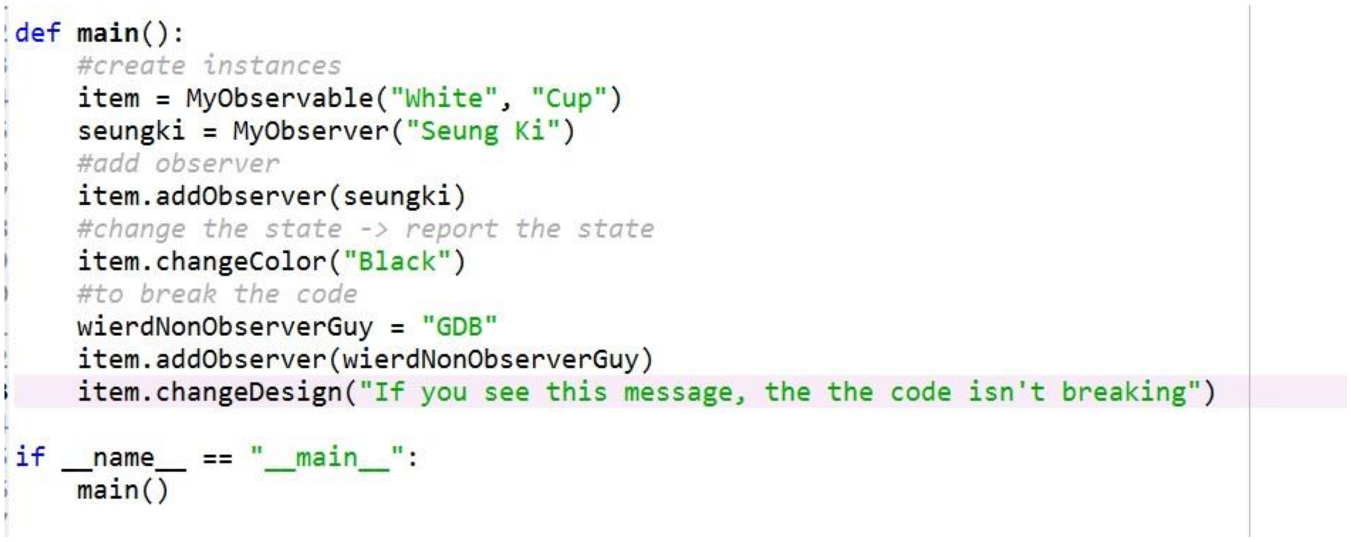
All of the code screenshot



```python
class Observable(object):
    def __init__(self):
        pass
    def addObserver(self, inObserver):
        if hasattr(inObserver, 'update'):
            self.observers.add(inObserver)
        else:
            print("The input does not have 'update' function implemented.")
    def removeObserver(self, inObserver):
        self.observers.discard(inObserver)
    def notifyObservers(self, message):
        pass

class MyObservable(Observable):
    #for all users in self.observer, call update
    def __init__(self, color, design):
        self.color = color
        self.design = design
        self.observers = set()
    def notifyObservers(self, message):
        for var in self.observers:
            var.update(message)
    def changeColor(self, changeColor):
        self.color = changeColor
        self.notifyObservers("The Color of your item has changed to " + str(changeColor))
    def changeDesign(self, changeDesign):
        self.design = changeDesign
        self.notifyObservers("The Design of your item has changed to " +str(changeDesign))
```



```python
class MyObserver():
    def __init__(self, name):
        self.name = name
    def update(self, message):
        print('{}, a new message has been sent: "{}"'.format(self.name, message))
```



```python
def main():
    #create instances
    item = MyObservable("White", "Cup")
    seungki = MyObserver("Seung Ki")
    #add observer
    item.addObserver(seungki)
    #change the state -> report the state
    item.changeColor("Black")
    #to break the code
    wierdNonObserverGuy = "GDB"
    item.addObserver(wierdNonObserverGuy)
    item.changeDesign("If you see this message, the the code isn't breaking")

if __name__ == "__main__":
    main()
```