Python Lab 6.2: OOP

> The purpose of this practice is to help you apply the concepts discussed up to **now**:
>
> - implement object classes and develop program to use them
> - test, debug and predict program behaviour

In `lab6_2.py` in the text editor at top-right, The exercise template contains a pre-defined BankAccount class, which can be used to model a BankAccount. Your task is to correct a logical error and create the following methods in the BankAccount Class:

1. A method to deposit an amount to a BankAccount
2. A method to withdraw an amount from a BankAccount
3. An **str** method which will return a string representation of the account

Bank Account - Specifications

1. Run the code, find and fix the logical error.

▶ HINT 1: The Error

```
Notice the order of the arguments in the main method, as compared to the order of
parameters in the constructor
```

1. The Deposit method should accept an amount and increment the account balance by that amount

▶ HINT 2: The Deposit Method

```
def deposit(self, amount):
        self.balance = self.balance + ______  # Which variable should be added?
```

1. The Withdraw method should accept an amount and decrement the account balance by that amount. If the balance is not enough the withdrawal is considered as successful and the method should return true, else it should return false.

▶ HINT 3: The WithDraw Method

```
 def withdraw(self, amount: float):
        if amount <= self.balance:
            self.balance -= amount
            return _____ # what value should be returned?

        return _____ # what value should be returned?
```

1. the **str** method should return a formatted output like the following:

```
BankAccount Owner: Nik, Balance: 1000.0
```

▶ HINT 4: The str Method

```
  def __str__(self) :
        return f"BankAccount Owner: {self.owner}, Balance: {self.balance}"
```

Main Program - Specifications

1. Delete or comment out all statements involving nik_account variable
2. Instantiate 1 Bank Account variable called john_account, which belongs to John and has an initial balance of 1000
3. Deposit 100 to this Bank Account
4. Try to withdraw 250
5. Try to withdraw 5000
6. Print the Bank Account balance

▶ HINT 5: Main Program

```
john_account =  BankAccount("John", 1000.0)
john_account.deposit(100)
john_account.withdraw(250)
john_account.withdraw(5000)
print(john_account.balance)
```

{% next %}

Execute and Test your program

Remember in order to execute your code you type in the terminal:

```
python lab6_2.py
```

Check that your code produces correct results, that is a balance of 850.0

```
850.0
```

Check Your Code

Execute the below to evaluate the correctness of your code using `check50`, but be sure to test it yourself also.

```
check50 mkotsovoulou/ods6001a/main/labs/lab6_2
```

Execute the below to evaluate the style of your code using `style50`.

```
style50 lab6_2.py
```

{% next %}

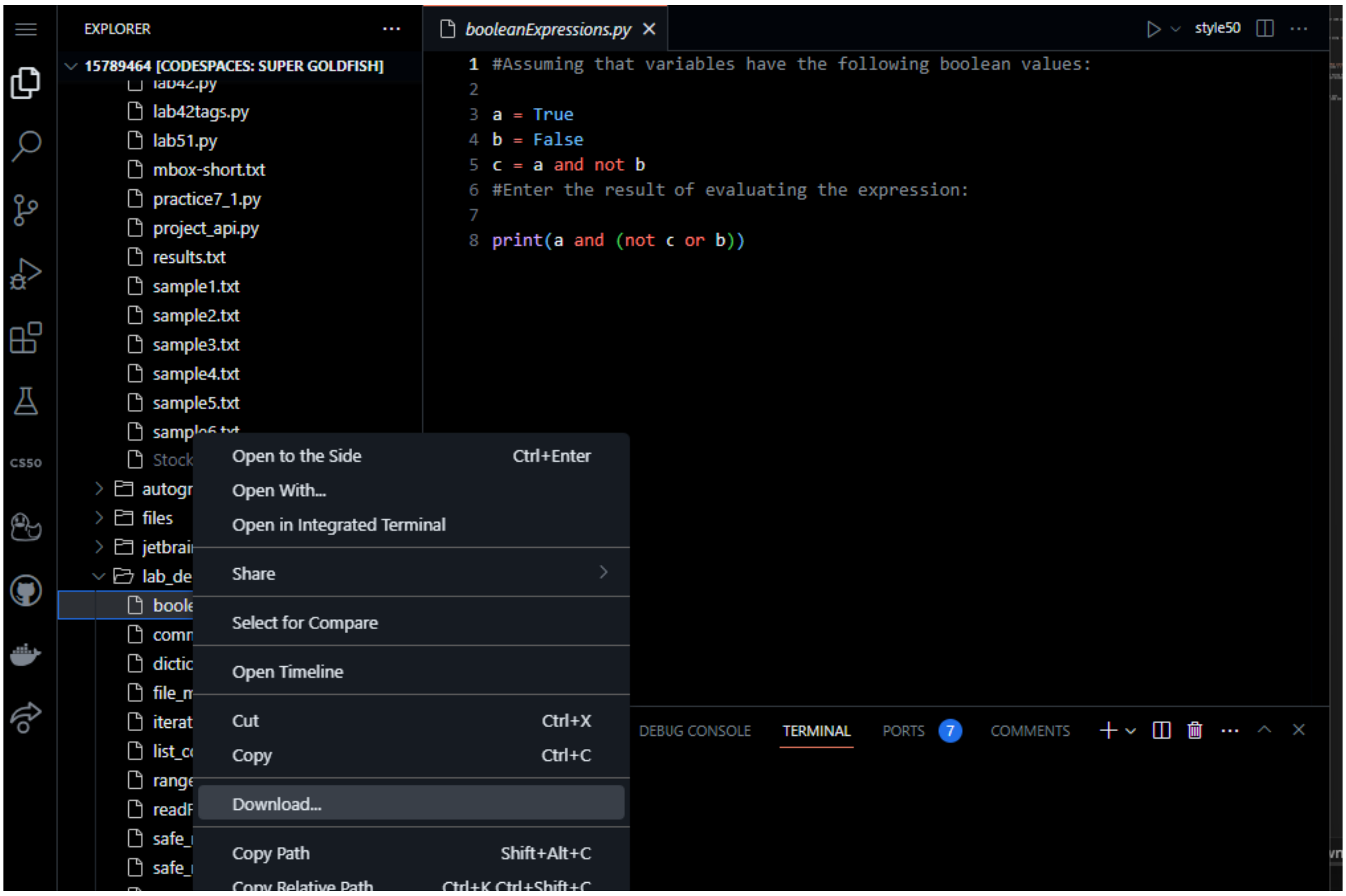
Submit your code

Execute the command below, logging in with your `GitHub username` and `Personal Access Token` when prompted. For security, you'll see asterisks (`*`) instead of the actual characters in your token.

If you do not have generated a Personal Access ToKen follow the instructions: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

```
submit50 mkotsovoulou/ods6001a/main/labs/lab6_2
```

You can re-submit your solution as many times as you want. When you are happy with your solution, download the code and upload it to Canvas.



```
#Assuming that variables have the following boolean values:

a = True
b = False
c = a and not b
#Enter the result of evaluating the expression:

print(a and (not c or b))
```

Done!

🎉