# Lab 06: Object Oriented Programming 2

AUTHOR
Christian Osendorfer

## Implement in Python

Unless otherwise stated, don't use any modules that implement a solution to the questions asked. Come up with your doctests and compare these with those from other students.

## Account

Let's say we'd like to model a bank account that can handle interactions such as depositing funds or gaining interest on current funds:

```
class Account:
    """An account has a balance and a holder.
    >>> a = Account('John')
    >>> a.deposit(10)
    10
    >>> a.balance
    10
    >>> a.interest
    0.02
    >>> a.time_to_retire(10.25) # 10 → 10.2 → 10.404
    2
    >>> a.balance               # balance should not change
    10
    >>> a.time_to_retire(11)    # 10 → 10.2 → ... → 11.040808032
    5
    >>> a.time_to_retire(100)
    117
    """
    max_withdrawal = 10
    interest = 0.02

    def __init__(self, account_holder):
        self.balance = 0
        self.holder = account_holder

    def deposit(self, amount):
        self.balance = self.balance + amount
        return self.balance

    def withdraw(self, amount):
        if amount > self.balance:
            return "Insufficient funds"
        if amount > self.max_withdrawal:
            return "Can't withdraw that amount"
        self.balance = self.balance - amount
        return self.balance
```

Add a `time_to_retire` method to `Account`. This method takes in an amount and returns how many years the holder would need to wait in order for the current balance to grow to at least amount, assuming that the bank adds balance times the interest rate to the total balance at the end of every year.

```
    def time_to_retire(self, amount):
        """Return the number of years until balance would grow to amount."""
        assert self.balance > 0 and amount > 0 and self.interest > 0
        # Your Code Here
```

## FreeChecking

Implement the `FreeChecking` class, which is like `Account` except that it charges a withdraw fee after 2 withdrawals. If a withdrawal is unsuccessful, it still counts towards the number of free withdrawals remaining, but no fee for the withdrawal will be charged.

```
    class FreeChecking(Account):
        """A bank account that charges for withdrawals, but the first two are fr
        >>> ch = FreeChecking('Jack')
        >>> ch.balance = 20
        >>> ch.withdraw(100)  # First one's free
        'Insufficient funds'
        >>> ch.withdraw(3)    # And the second
        17
        >>> ch.balance
        17
        >>> ch.withdraw(3)    # Ok, two free withdrawals is enough
        13
        >>> ch.withdraw(3)
        9
        >>> ch2 = FreeChecking('John')
        >>> ch2.balance = 10
        >>> ch2.withdraw(3) # No fee
        7
        >>> ch.withdraw(3)  # ch still charges a fee
        5
        >>> ch.withdraw(5)  # Not enough to cover fee + withdraw
        'Insufficient funds'
        """
        withdraw_fee = 1
        free_withdrawals = 2
        # Your Code Here
```