Homework 6: Object Oriented Programming

hw06.zip (hw06.zip)

Due by 11:59pm on Friday, 3/15

Instructions

Download hw06.zip (hw06.zip). Starter code for the problems can be found in `hw06.py`.

Submission: When you are done, submit with `python3 ok --submit`. You may submit more than once before the deadline; only the final submission will be scored. Check that you have successfully submitted your code on okpy.org (https://okpy.org/). See Lab 0 (/lab/lab00#submitting-the-assignment) for more instructions on submitting assignments.

Using Ok: If you have any questions about using Ok, please refer to this guide. (/articles/using-ok.html)

Readings: You might find the following references useful:

- Section 2.5 (http://composingprograms.com/pages/25-object-oriented-programming.html)

Grading: Homework is graded based on effort, not correctness. However, there is no partial credit; you must show substantial effort on every problem to receive any points.

Q0: Survey

Before you get started writing code, please fill out the midterm survey (https://goo.gl/forms/TtVWIGH5s1touFzm2).

Important Submission Note

> **You're not done yet!** Add the passphrase you receive at the end of the survey to passphrase at the top of the homework. For example, if the passphrase was `CS61A` (it isn't ☺), then the first line of your file should read:

```
passphrase = 'CS61A'
```

> Instead of:

```
passphrase = '*** PASSPHRASE HERE ***'
```

Use Ok to test your code:

```
python3 ok -q survey
```

Object Oriented Programming

Q1: Next Fibonacci Object

Implement the `next` method of the `Fib` class. For this class, the `value` attribute is a Fibonacci number. The `next` method returns a `Fib` instance whose `value` is the next Fibonacci number. The `next` method should take only constant time.

Note that in the doctests, nothing is being printed out. Rather, each call to `.next()` returns a `Fib` instance. The way each `Fib` instance is displayed is determined by the return value of its `__repr__` method.

> *Hint*: Keep track of the previous number by setting a new instance attribute inside `next`. You can create new instance attributes for objects at any point, even outside the `__init__` method.

```
class Fib():
    """A Fibonacci number.

    >>> start = Fib()
    >>> start
    Fib object, value 0
    >>> start.next()
    Fib object, value 1
    >>> start.next().next()
    Fib object, value 1
    >>> start.next().next().next()
    Fib object, value 2
    >>> start.next().next().next().next()
    Fib object, value 3
    >>> start.next().next().next().next().next()
    Fib object, value 5
    >>> start.next().next().next().next().next().next()
    Fib object, value 8
    >>> start.next().next().next().next().next().next() # Ensure start isn't changed
    Fib object, value 8
    """

    def __init__(self, value=0):
        self.value = value

    def next(self):
        "*** YOUR CODE HERE ***"

    def __repr__(self):
        return "Fib object, value " + str(self.value)
```

Use Ok to test your code:

```
python3 ok -q Fib
```

Q2: Vending Machine

Create a class called `VendingMachine` that represents a vending machine for some product. A `VendingMachine` object returns strings describing its interactions.
Fill in the `VendingMachine` class, adding attributes and methods as appropriate, such that its behavior matches the following doctests:

```
class VendingMachine:
    """A vending machine that vends some product for some price.

    >>> v = VendingMachine('candy', 10)
    >>> v.vend()
    'Machine is out of stock.'
    >>> v.deposit(15)
    'Machine is out of stock. Here is your $15.'
    >>> v.restock(2)
    'Current candy stock: 2'
    >>> v.vend()
    'You must deposit $10 more.'
    >>> v.deposit(7)
    'Current balance: $7'
    >>> v.vend()
    'You must deposit $3 more.'
    >>> v.deposit(5)
    'Current balance: $12'
    >>> v.vend()
    'Here is your candy and $2 change.'
    >>> v.deposit(10)
    'Current balance: $10'
    >>> v.vend()
    'Here is your candy.'
    >>> v.deposit(15)
    'Machine is out of stock. Here is your $15.'

    >>> w = VendingMachine('soda', 2)
    >>> w.restock(3)
    'Current soda stock: 3'
    >>> w.restock(3)
    'Current soda stock: 6'
    >>> w.deposit(2)
    'Current balance: $2'
    >>> w.vend()
    'Here is your soda.'
    """
    "*** YOUR CODE HERE ***"
```

> You may find Python string formatting syntax (https://docs.python.org/2/library/stdtypes.html#str.format) useful. A quick example:

```
>>> ten, twenty, thirty = 10, 'twenty', [30]
>>> '{0} plus {1} is {2}'.format(ten, twenty, thirty)
'10 plus twenty is [30]'
```

Use Ok to test your code:

```
python3 ok -q VendingMachine
```

CS 61A (/)

Weekly Schedule (/weekly.html)

Office Hours (/office-hours.html)

Staff (/staff.html)

Resources (/resources.html)

Studying Guide (/articles/studying.html)

Debugging Guide (/articles/debugging.html)

Composition Guide (/articles/composition.html)

Policies (/articles/about.html)

Assignments (/articles/about.html#assignments)

Exams (/articles/about.html#exams)

Grading (/articles/about.html#grading)