

Homework 6: OOP, Linked Lists

hw06.zip (hw06.zip)

Due by 11:59pm on Thursday, October 24

Instructions

Download [hw06.zip](#) ([hw06.zip](#)). Inside the archive, you will find a file called [hw06.py](#) ([hw06.py](#)), along with a copy of the `ok` autograder.

Submission: When you are done, submit the assignment by uploading all code files you've edited to Gradescope. 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 Gradescope. See [Lab 0 \(../lab/lab00#task-c-submitting-the-assignment\)](#) for more instructions on submitting assignments.

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

Grading: Homework is graded based on correctness. Each incorrect problem will decrease the total score by one point. **This homework is out of 2 points.**

Required Questions

Getting Started Videos

Midsemester Survey

Q1: Mid-Semester Feedback

As part of this assignment, fill out the Mid-Semester Feedback (<https://forms.gle/HgHnpLj1SC4jRAxE9>) form.

Once you finish the survey, you will be presented with a passphrase. Put this passphrase, as a string, on the line that says `passphrase = 'REPLACE_THIS_WITH_PASSPHRASE'` in the Python file for this assignment. E.g. if the passphrase is `abc`, then the line should be `passphrase = 'abc'`.

Use Ok to test your code:

```
python3 ok -q midsem_survey
```



OOP

Q2: Vending Machine

In this question you'll create a vending machine (https://en.wikipedia.org/wiki/Vending_machine) that sells a single product and provides change when needed.

Implement the `VendingMachine` class, which models a vending machine for one specific product. The methods of a `VendingMachine` object return strings to describe the machine's status and operations. Ensure that your output matches *exactly* with the strings provided in the doctests, including punctuation and spacing.

You may find Python's formatted string literals, or f-strings (<https://docs.python.org/3/tutorial/inputoutput.html#fancier-output-formatting>) useful. A quick example:

```
>>> feeling = 'love'
>>> course = '61A!'
>>> combined_string = f'I {feeling} {course}'
>>> combined_string
'I love 61A!'
```

class VendingMachine:

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

```
>>> v = VendingMachine('candy', 10)
```

```
>>> v.vend()
```

```
'Nothing left to vend. Please restock.'
```

```
>>> v.add_funds(15)
```

```
'Nothing left to vend. Please restock. Here is your $15.'
```

```
>>> v.restock(2)
```

```
'Current candy stock: 2'
```

```
>>> v.vend()
```

```
'Please add $10 more funds.'
```

```
>>> v.add_funds(7)
```

```
'Current balance: $7'
```

```
>>> v.vend()
```

```
'Please add $3 more funds.'
```

```
>>> v.add_funds(5)
```

```
'Current balance: $12'
```

```
>>> v.vend()
```

```
'Here is your candy and $2 change.'
```

```
>>> v.add_funds(10)
```

```
'Current balance: $10'
```

```
>>> v.vend()
```

```
'Here is your candy.'
```

```
>>> v.add_funds(15)
```

```
'Nothing left to vend. Please restock. Here is your $15.'
```

```
>>> w = VendingMachine('soda', 2)
```

```
>>> w.restock(3)
```

```
'Current soda stock: 3'
```

```
>>> w.restock(3)
```

```
'Current soda stock: 6'
```

```
>>> w.add_funds(2)
```

```
'Current balance: $2'
```

```
>>> w.vend()
```

```
'Here is your soda.'
```

```
"""
```

```
def __init__(self, product, price):
```

```
    """Set the product and its price, as well as other instance attributes."""
```

```
    """ YOUR CODE HERE """
```

```
def restock(self, n):
```

```
    """Add n to the stock and return a message about the updated stock level.
```

E.g., Current candy stock: 3

```

"""
*** YOUR CODE HERE ***

def add_funds(self, n):
    """If the machine is out of stock, return a message informing the user to restock
    (and return their n dollars).

    E.g., Nothing left to vend. Please restock. Here is your $4.

    Otherwise, add n to the balance and return a message about the updated balance.

    E.g., Current balance: $4
    """
    *** YOUR CODE HERE ***

def vend(self):
    """Dispense the product if there is sufficient stock and funds and
    return a message. Update the stock and balance accordingly.

    E.g., Here is your candy and $2 change.

    If not, return a message suggesting how to correct the problem.

    E.g., Nothing left to vend. Please restock.
           Please add $3 more funds.
    """
    *** YOUR CODE HERE ***

```

Use Ok to test your code:

```
python3 ok -q VendingMachine
```



Linked Lists

Q3: Store Digits

Write a function `store_digits` that takes in an integer `n` and returns a linked list containing the digits of `n` in the same order (from left to right).

Important: Do not use any string manipulation functions, such as `str` or `reversed`.

```
def store_digits(n):
    """Stores the digits of a positive number n in a linked list.

    >>> s = store_digits(1)
    >>> s
    Link(1)
    >>> store_digits(2345)
    Link(2, Link(3, Link(4, Link(5))))
    >>> store_digits(876)
    Link(8, Link(7, Link(6)))
    >>> store_digits(2450)
    Link(2, Link(4, Link(5, Link(0))))
    >>> store_digits(20105)
    Link(2, Link(0, Link(1, Link(0, Link(5)))))
    >>> # a check for restricted functions
    >>> import inspect, re
    >>> cleaned = re.sub(r"#.*\\n", '', re.sub(r'"{3}[\s\S]*?"{3}', '', inspect.getsourcel
    >>> print("Do not use str or reversed!") if any([r in cleaned for r in ["str", "revers
    """

    """*** YOUR CODE HERE ***"""
```

Use Ok to test your code:

```
python3 ok -q store_digits
```



Q4: Mutable Mapping

Implement `deep_map_mut(func, s)`, which applies the function `func` to each element in the linked list `s`. If an element is itself a linked list, recursively apply `func` to its elements as well.

Your implementation should *mutate* the original linked list. **Do not create any new linked lists.** The function returns `None`.

Hint: You can use the built-in `isinstance` function to determine if an element is a linked list.

```
>>> s = Link(1, Link(2, Link(3, Link(4))))
>>> isinstance(s, Link)
True
>>> isinstance(s, int)
False
```

Construct Check: The final test case for this problem checks that your function does not create any new linked lists. If you are failing this doctest, make sure that you are not creating link lists by calling the constructor, i.e.

```
s = Link(1)
```

```
def deep_map_mut(func, s):
```

```
    """Mutates a deep link s by replacing each item found with the
    result of calling func on the item. Does NOT create new Links (so
    no use of Link's constructor).
```

```
    Does not return the modified Link object.
```

```
>>> link1 = Link(3, Link(Link(4), Link(5, Link(6))))
>>> print(link1)
<3 <4> 5 6>
>>> # Disallow the use of making new Links before calling deep_map_mut
>>> Link.__init__, hold = lambda *args: print("Do not create any new Links."), Link.__init__
>>> try:
...     deep_map_mut(lambda x: x * x, link1)
... finally:
...     Link.__init__ = hold
>>> print(link1)
<9 <16> 25 36>
"""
"""*** YOUR CODE HERE ***"
```

Use Ok to test your code:

```
python3 ok -q deep_map_mut
```



Check Your Score Locally

You can locally check your score on each question of this assignment by running

```
python3 ok --score
```

This does NOT submit the assignment! When you are satisfied with your score, submit the assignment to Gradescope to receive credit for it.

Submit Assignment

Submit this assignment by uploading any files you've edited **to the appropriate Gradescope assignment**. [Lab 00 \(../lab/lab00/#submit-with-gradescope\)](#) has detailed instructions.

Optional Questions

Q5: Two List

Implement a function `two_list` that takes in two lists and returns a linked list. The first list contains the values that we want to put in the linked list, and the second list contains the number of each corresponding value. Assume both lists are the same size and have a length of 1 or greater. Assume all elements in the second list are greater than 0.

```
def two_list(vals, counts):  
    """  
    Returns a linked list according to the two lists that were passed in. Assume  
    vals and counts are the same size. Elements in vals represent the value, and the  
    corresponding element in counts represents the number of this value desired in the  
    final linked list. Assume all elements in counts are greater than 0. Assume both  
    lists have at least one element.  
    >>> a = [1, 3]  
    >>> b = [1, 1]  
    >>> c = two_list(a, b)  
    >>> c  
    Link(1, Link(3))  
    >>> a = [1, 3, 2]  
    >>> b = [2, 2, 1]  
    >>> c = two_list(a, b)  
    >>> c  
    Link(1, Link(1, Link(3, Link(3, Link(2)))))  
    """  
    "*** YOUR CODE HERE ***"
```

Use Ok to test your code:

```
python3 ok -q two_list
```



Exam Practice

Homework assignments will also contain prior exam questions for you to try. These questions have no submission component; feel free to attempt them if you'd like some practice!

Object-Oriented Programming

1. Spring 2022 MT2 Q8: CS61A Presents The Game of Hoop.
(<https://cs61a.org/exam/sp22/mt2/61a-sp22-mt2.pdf#page=17>).
2. Fall 2020 MT2 Q3: Sparse Lists (<https://cs61a.org/exam/fa20/mt2/61a-fa20-mt2.pdf#page=9>).
3. Fall 2019 MT2 Q7: Version 2.0 (<https://cs61a.org/exam/fa19/mt2/61a-fa19-mt2.pdf#page=8>).

Linked Lists

1. Fall 2020 Final Q3: College Party (<https://cs61a.org/exam/fa20/final/61a-fa20-final.pdf#page=9>).
2. Fall 2018 MT2 Q6: Dr. Frankenlink (<https://cs61a.org/exam/fa18/mt2/61a-fa18-mt2.pdf#page=6>).
3. Spring 2017 MT1 Q5: Insert (<https://cs61a.org/exam/sp17/mt1/61a-sp17-mt1.pdf#page=7>).

