NYU SHANGHAI 上海纽约大学

CSCI-SHU 210 Data Structures 100 Points

Homework Assignment 1 - Python Review and OOP

Problem 1 – String Generator - 20 Points

Write a Python generator that yields all possible strings formed by using the characters ‘c’, ‘a’, ‘t’, ‘d’ , ‘o’ , and ‘g’ exactly once. When called, it generates an iterator.

Requirements

- The method has to be non-recursive
- Your solution has to be a generator.
- You are not allowed to use any third-party libraries such as itertools.
- The order in which your generator yields strings does not matter.
- You are not allowed to store generated substrings.
- You can not use Python sets.

Example 1

```
cat_dog = string_generator()
print(next(cat_dog), next(cat_dog), next(cat_dog)) # should print: godtac, godtca, godatc, …
```

Problem 2 – My Iterable-Iterator of 20 - 20 Points

Write the class iterable-iterator that counts from 1 to 20 before staring over from 1.

Requirements

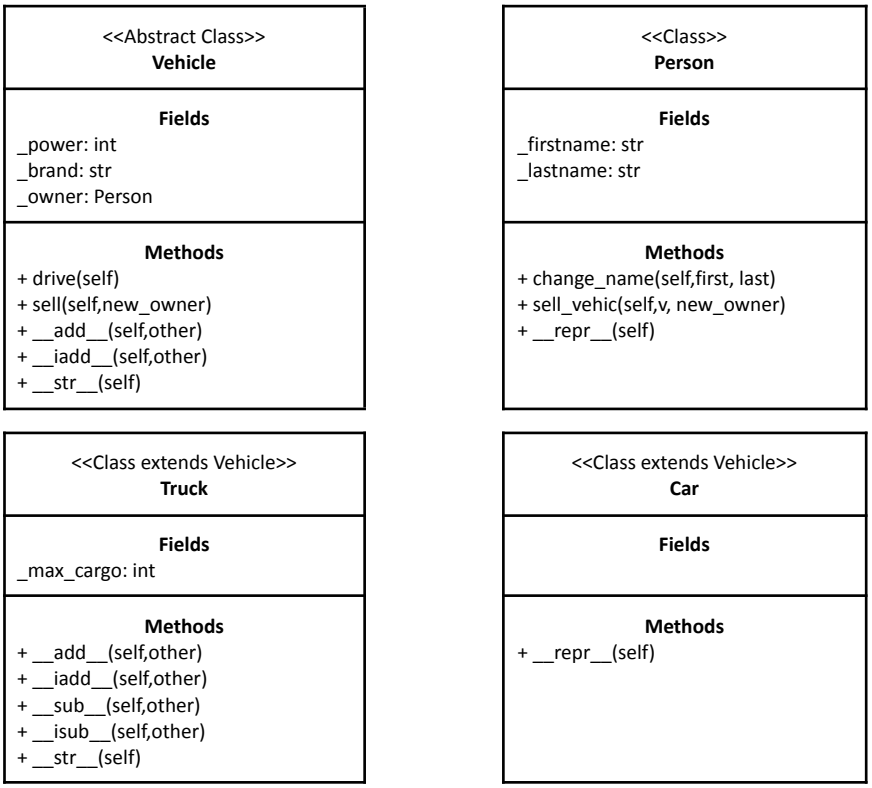
- Write a class IterableIterator that counts from 1 to 20 over and over.
- The class implements the iterable functions __len__ and __getitem__
- The class implements the iterator functions __iter__ and __next__
- You have to implement all functions correctly.

Example 1

```
my_obj = IterableIterator()
my_it = iter(my_obj)
print(my_obj[41]) # this prints 2
print(next(my_it), next(my_it), next(my_it), next(my_it)) # this prints 1 2 3 4
```

Problem 3 – Vehicle Class - 30 Points

Implement the following UML diagrams as Python classes as specified below.



Requirements

- Implement the classes as outlined above.
- Implement add, sub, iadd, and isub correctly. Return either new or edit existing instances.
- Implement and support the following operations:
    - Vehicles are initialized with a power value, a brand name, and an owner.
    - Vehicles can drive. This method returns the string “Driving”.
    - Vehicles can be sold (e.g., car1.sell(Person(“Peter”,”Moldan”)))
    - Adding vehicles adds powers to instances (e.g., car1._power + car1._power)
    - When a vehicle is printed, the following format is used:
        - “Brand: _brand, Power: _power, Owner: person.__repr__”
    - Persons can change their name (e.g., person1.change_name(“New”, “Name”))
    - A person can sell a vehicle, only if the vehicle belongs to the person.
    - Calling represents on a person returns the following “Person(firstna, lastna)”
    - Adding trucks adds max_cargos (e.g., truck1._max_cargo + truck2._max_cargo)
    - Subtracting trucks subs max_cargos (e.g., truck1._max_cargo - truck2._max_cargo)
    - When a truck is printed the following format is used:
        - “Brand: _brand, Power: _power, Owner: person.__repr__, c: _max_cargo”
    - Calling represents on a car returns the following “Car(Brand, power, firstna, lastna)”

Problem 4 – Binary Palindrome - 30 Points

Write the function *is_palindrome(num)*, which accepts a positive integer and checks whether this integer represents a palindrome in binary representation. You have to use bitwise operations. You can not use any Python function such as bin(), int(), str(), or others. You can not use lists, arrays, strings, or any other datatypes.

Requirements

- You can only use bitwise operations
- You are not allowed to use any Python functions

Example 1

```
print(is_palindrome(220395))  # should print True
```

Example 2

```
print(is_palindrome(1060))  # should print False
```

Example 3

```
print(is_palindrome(75817))  # should print True
```

Example 4

```
print(is_palindrome(820))  # should print False
```

Example 5

```
print(is_palindrome(5557))   # should print True
```