Kolokwium 1

Informacje wstępne

1. Klonujemy repozytorium o nazwie PO-nr_indeksu-zaliczenia do wybranego katalogu.
2. W PyCharm w sklonowanym katalogu z repozytorium tworzymy projekt o nazwie K1-nr_indeksu, np. K1-12345.
3. W projekcie tworzymy 3 pliki: vehicle.py, car.py oraz main.py
4. W trakcie kolokwium można korzystać z narzędzi wbudowanych w PyCharm lub zainstalowanych pakietów takich jak: flake8 lub mypy w celu sprawdzenia typowania i PEP8.
5. W przypadku braku internetu student zapisuje projekt jako archiwum zip. Kolokwia będą zebrane np. za pomocą pendrive.
6. Rozwiązania po terminie: z uwagi na rozbieżność czasu na stanowiskach w pracowni bieżący czas będzie wyświetlony za pomocą rzutnika. Studenci zostaną uprzedzeni ustnie o zbliżającym się końcu czasu w okresie między 10 a 5 minut przed końcem kolokwium. Brak przesłania repozytorium w wyznaczonym czasie powoduje dodanie dodatkowych 5 minut, ale wtedy maksymalna ocena to 3,5. Powyżej 5 minut spóźnienia - kolokwium pozostaje bez sprawdzenia z oceną niedostateczną. Czas liczony jest od momentu udostępnienia poleceń do poprawnego wypchnięcia na zdalne repo na Githubie lub utworzenia pliku zip w wypadku braku internetu.

Zadanie 1. Klasa Vehicle (pol. Pojazd) (10pt max.)

1. Napisz klasę Vehicle. Klasa powinna posiadać prywatne atrybuty instancyjne:
    1. reg (pol. rejestracja), typ string.
    2. model, typ int,
    3. prod_year (pol. rok produkcji), typ int.
2. Zaimplementuj:
    1. **(3pt)** Inicjalizator z trzema argumentami, gdzie wartościami domyślnymi dla reg będzie None, a dla model 0, a dla prod_year 2022. Zadbaj również o to aby inicjalizator w razie podania modelu mniejszego od 0 lub podania roku mniejszego niż 1900 lub większego niż 2022 ustawiał wartość domyślną danego atrybutu. (W inicjalizatorze zachowaj kolejność argumentów jak w pt.1!).
    2. **(2pt)** Zaimplementuj właściwości (propercje) setter i getter dla każdego atrybutu. Jeśli dla setterów podane wartości argumentów nie spełniają założeń, wartość atrybutu nie powinna się zmieniać i powinien zostać wypisany odpowiedni komunikat.

3. **(1pt)** Zaimplementuj metody:
    1. brake, zwracającą napis „Zatrzymuję się”, oraz
    2. drive zwracającą napis „Jadę świetnym pojazdem z roku {tu wartosc pora rok produkcji}!”.
4. Nadpisz:
    1. **(2pt)** Odpowiednią metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy:

       „Pojazd wyprodukowany w roku: 2019.

       Model: 1.

       Rejestracja: XYZ12345.”

       Gdy wartość danego atrybutu wynosi None, ta część nie powinna się wyświetlać np.:

       „Pojazd wyprodukowany w roku: 2019.

       Model: 1.”
    2. **(2pt)** Metody magiczne __eq__, __ne__ porównujące obiekty po modelu, a jeśli te są sobie równe to po roku produkcji.

Zadanie 2. Klasa Car (pol. Samochód) (10pt max.)

1. Zaimplementuj klasę Car dziedziczącą po klasie Vehicle. Klasa ta powinna posiadać trzy atrybuty instancyjne prywatne:
    1. price (pol. cena), typu float,
    2. colour (pol. kolor), typu string,
    3. extra_seats (pol. dodatkowe siedzenia), typu int, reprezentujący liczbę ile dodatkowych siedzeń można włożyć do auta.
2. **(3pt)** Nadpisz inicjalizator. Domyślne wartości to: dla price 0, dla colour None, dla extra_seats 0. Jeśli cena samochodu jest ujemna powinna być ustawiona domyślna wartość. Jeśli liczba dodatkowych siedzeń jest mniejsza od 0 powinna być ustawiona domyślna wartość.
3. **(2pt)** Zaimplementuj właściwości (propercje) setter i getter dla każdego atrybutu. Jeśli dla setterów podane wartości argumentów nie spełniają założeń, wartość atrybutu nie powinna się zmieniać i powinien zostać wypisany odpowiedni komunikat.
4. **(1pt) Nadpisz** metodę drive aby zwracała napis postaci: „Jadę świetnym pojazdem z roku {tu wartość pora rok produkcji}! Ma kolor {tu kolor}.”.
5. **(2pt) Nadpisz** metody magiczne __eq__, __ne__ porównujące obiekty biorąc pod uwagę jedynie model oraz cenę.

6. **(2pt) Nadpisz** metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy:

   a) „Pojazd wyprodukowany w roku: 2019.

   Model: 3.

   Rejestracja: XYZ12345.

   Cena: 20000.

   Kolor: czerwony.

   Dodatkowe siedzenia: 3.”

   Gdy wartość danego atrybutu wynosi None, ta część nie powinna się wyświetlać np.:

   „Pojazd wyprodukowany w roku: 2019.
   Model: 6.

   Cena: 20000.

   Kolor: czerwony.

   Dodatkowe siedzenia: 3.”

Polecenia pomocnicze do testowania klas. Plik main.py z funkcją main() (Polecenia mają na celu pomoc w testowaniu napisanych klas)

Część do klasy Vehicle:

1. Stwórz obiekt o nazwie v_1 i zainicjalizuj go domyślnymi wartościami.
2. Wypisz na ekran reprezentację obiektu v_1.
3. Stwórz obiekt o nazwie v_2 i zainicjalizuj go wybranymi danymi.
4. Wypisz na ekran reprezentację obiektu v_2.
5. Wypisz na ekran wartość atrybutu model obiektu v_1.
6. Zmień wartość atrybutu prod_year obiektu v_1 na 1990 i wypisz na ekran reprezentację obiektu v_1 po zmianie stanu.

Część do klasy Car:

1. Stwórz obiekt o nazwie c_1 i zainicjalizuj go domyślnymi wartościami.
2. Wypisz na ekran reprezentację obiektu c_1.
3. Stwórz obiekt o nazwie c_2 i zainicjalizuj go danymi jak w punkcie 2a.
4. Wypisz na ekran reprezentację obiektu c_2.
5. Sprawdź czy obiekty c_1 i c_2 są sobie równe.