Klasy i obiekty — część 2

1. Przeładowywanie funkcji oraz operatorów

1 pkt za wykonanie zadań z tej sekcji

1. Wykonaj komendę `python3 -i DeaneriySystem/term.py`, a następnie (w konsoli Python) wprowadź poniższy kod:

```
term1 = Term(Day.MON, 8, 30)
term2 = Term(Day.TUE, 9, 45, 30)
term3 = Term(Day.TUE, 9, 45, 90)
print(term1)                                # Ma się wypisać: "Poniedziałek
print(term2)                                # Ma się wypisać: "Wtorek 9:45
print(term3)                                # Ma się wypisać: "Wtorek 9:45
print("term1 < term2:", term1 < term2)   # Ma się wypisać True
print("term1 <= term2:", term1 <= term2) # Ma się wypisać True
print("term1 > term2:", term1 > term2)   # Ma się wypisać False
print("term1 >= term2:", term1 >= term2) # Ma się wypisać False
print("term2 == term2:", term2 == term2) # Ma się wypisać True
print("term2 == term3:", term2 == term3) # Ma się wypisać False
term4 = term3 - term1                       # Tworzy termin, którego:
                                            # - termin rozpoczęcia jest tak
                                            # - czas trwania to różnica mir

print(term4)                                # Ma się wypisać "Poniedziałek
```

2. Przeczytaj artykuł "Overloading Functions and Operators in Python"
3. Rozbuduj klasę *Term* tak, aby pokazany wyżej kod nie kończył się błędem

2. Mechanizm agregacji oraz kompozycji

1 pkt za wykonanie zadań z tej sekcji

1. Utwórz klasę *Lesson*, która zawiera podane (publiczne) pola, metodę oraz konstruktor:
   - pole *term* typu `Term`
   - pole *name* typu `String`
   - pole *teacherName* typu `String`
   - pole *year* typu `int`
   - pole *fullTime* typu `bool`, którego wartość jest określana automatycznie, w konstruktorze (patrz informacje zawarte w pkt. 2). Wartość `true` oznacza, że podane zajęcia są zajęciami na studiach stacjonarnych, zaś `false` oznacza zajęcia na studiach niestacjonarnych
   - konstruktor `Lesson(Term term, String name, String teacherName, int year)` przypisuje polu *term*, *name*, *teacherName* oraz *year* podane wartości

- metody: `earlierDay()`, `laterDay()`, `earlierTime()` oraz `laterTime()`:
  - metody `earlierDay()` oraz `laterDay()` przesuwają zajęcia, odpowiednio, o jeden dzień do tyłu lub do przodu, pod warunkiem, że jest to możliwe, tzn. zmodyfikowany termin spełnia założenia wymienione w punkcie 2.2
  - metody `earlierTime()` oraz `laterTime()` przesuwają zajęcia, odpowiednio, o *duration* minut do tyłu lub do przodu, pod warunkiem, że jest to możliwe
  - Metody te zwracają `true` jeżeli operacja przesunięcia powiodła się, w przeciwnym przypadku zwracają `false`
- metoda `__str__()` zwraca napis z informacją o terminie zajęć / lekcji w następującej postaci:

```
nazwa_zajęć (dzień_tygodnia godzina_rozpoczęcia-godzina_zakończenia)
rok i rodzaj studiów
imię i nazwisko wykładowcy
```

  przykład użycia:

```
lesson = Lesson(Term(Day.TUE, 9, 40), "Programowanie skryptowe", "St
print(lesson); """Programowanie skryptowe (Wtorek 9:40-11:10)
                  Drugi rok studiów stacjonarnych
                  Prowadzący: Stanisław Polak
                """
```

2. W bloku głównym utwórz przykładową lekcję oraz dodaj wywołania, które sprawdzą poprawność implementacji powyższych metod

   Implementując powyższe metody należy przyjąć, że:
   - zajęcia na studiach stacjonarnych mogą się odbywać **tylko** od **poniedziałku** do **czwartku** w godzinach **8:00-20:00** oraz w **piątek** w godzinach **8:00 - 17:00**
   - zajęcia na studiach niestacjonarnych mogą się odbywać **tylko** w **weekendy** w godzinach **8:00-20:00** oraz w **piątek** w godzinach **17:00 - 20:00**

3. Zaimplementuj testy sprawdzające poprawność działania metod przesuwających zajęcia

3. Mechanizm hermetyzacji

1 pkt za wykonanie zadań z tej sekcji

1. Utwórz typ wyliczeniowy *Action* o wartościach: `DAY_EARLIER`, `DAY_LATER`,`TIME_EARLIER` oraz `TIME_LATER`
2. Uczyń prywatnymi wszystkie pola w dotychczas utworzonych klasach — dokonaj pełnej hermetyzacji własności, a następnie zdefiniuj, dla każdego prywatnego pola, metodę ustawiającą oraz pobierającą w oparciu o dekoratory `@property` oraz `@nazwa_metody.setter`
3. Przeczytaj artykuł "Type hints", a następnie przeanalizuj poniższy kod źródłowy

```
from typing import List


class TimetableWithoutBreaks(object):
    """ Class containing a set of operations to manage the timetable """

############################################################
    def can_be_transferred_to(self, term: Term, fullTime: bool) -> bool:
        """
Informs whether a lesson can be transferred to the given term

Parameters
----------
term : Term
    The term checked for the transferability
fullTime : bool
    Full-time or part-time studies

Returns
-------
bool
    **True** if the lesson can be transferred to this term
"""

        pass

############################################################

    def busy(self, term: Term) -> bool:
        """
Informs whether the given term is busy.  Should not be confused with ``c
since there might be free term where the lesson cannot be transferred.

Parameters
----------
term : Term
    Checked term

Returns
-------
bool
    **True** if the term is busy
        """

        pass

 ############################################################
```

```
    def put(self, lesson: Lesson) -> bool:
        """
Add the given lesson to the timetable.

Parameters
----------
lesson : Lesson
    The added  lesson

Returns
-------
bool
    **True**  if the lesson was added.  The lesson cannot be placed if t
        """

        pass

############################################################

    def parse(self, actions: List[str]) -> List[Action]:
        """
Converts an array of strings to an array of 'Action' objects.

Parameters
----------
actions:  List[str]
    A list containing the strings: "d-", "d+", "t-" or "t+"

Returns
-------
    List[Action]
        A list containing the values:  DAY_EARLIER, DAY_LATER, TIME_EARL
        """

        pass

############################################################

    def perform(self, actions: List[Action]):
        """
Transfer the lessons included in the timetable as described in the list

Parameters
----------
actions : List[Action]
    Actions to be performed
        """
```

```
        pass
##########################################################

    def get(self, term: Term) -> Lesson:
        """
Get object (lesson) indicated by the given term.

Parameters
----------
term: Term
    Lesson date

Returns
-------
lesson: Lesson
    The lesson object or None if the term is free
        """

        pass
```

4. Uzupełnij treść metod klasy *TimetableWithoutBreaks* — klasa ma realizować następujące funkcjonalności:
   - Ma przechowywać informacje o wielu zajęciach / lekcjach w oparciu o listę
   - Ma pozwalać na przenoszenie zajęć

   - Metoda `parse(tablica)`:
     - wczytuje tablicę napisów
     - zamienia napisy "d-", "d+", "t-" oraz "t+" na wartości typu `Action`, odpowiednio: `DAY_EARLIER`, `DAY_LATER`, `TIME_EARLIER` oraz `TIME_LATER`
     - nieznane napisy (opcje) ignoruje
     - zwraca tablicę opcji `Action`
   - Metoda `perform()` naprzemiennie kieruje procesem przenoszenia zajęć, tzn. jeżeli przykładowo założymy, że rozkład zajęć zawiera dwa przedmioty (lekcje), to dla ciągu akcji `d+ d- t- t+`, akcje o numerach nieparzystych (`d+ t-`) dotyczy pierwszych zajęć, a akcje o numerach parzystych (`d- t+`), drugich. Przetwarzanie ciągu wyjściowego ma się odbywać sekwencyjnie (po kolei)
   - Metoda `__str__()` ma rysować rozkład zajęć w postaci tabeli, analogicznej do pokazanej poniżej

```
            *Poniedziałek*Wtorek     *Środa       *Czwartek    *
            ***************************************************
8:00-9:30   *            *Angielski  *            *            *
            ***************************************************
9:30-11:00  *    JTP     *           *            *            *
```

```
                ***************************************************
...             *            *           *            *           *
                ***************************************************
18:30-20:00     *            *           *            *           *
                ***************************************************
```

5. Utwórz konstruktor `Lesson(TimetableWithoutBreaks timetable, Term term, String name, string teacherName, int year)`, który przypisuje swoje argumenty do odpowiednich pól
6. Zmodyfikuj metody `earlier*()` oraz `later*()` klasy *Lesson* tak, aby korzystały z `can_be_transferred_to()` klasy *TimetableWithoutBreaks* (argument 'timetable' konstruktora) — termin zajęć może być przesunięty pod warunkiem, że:
   1. W tym samym czasie nie odbywają się inne zajęcia
   2. Nie wychodzimy poza przyjęte ramy czasowe — patrz pkt 2.2

   Tak więc zakładamy, że badaniem możliwości przesunięcia lekcji zajmuje się klasa *TimetableWithoutBreaks* (metoda `can_be_transferred_to()`), a nie klasa *Lesson* (jak to było poprzednio)
7. Sprawdź czy zajęcia są poprawnie przenoszone dla ciągu: `t- d- t+ d-`
8. **Napisz testy** sprawdzające poprawność działania zaimplementowanych metod
9. (**2 pkt.**) Rozbuduj skrypt z poprzednich ćwiczeń o nowe klasy oraz metody zgodnie z podaną, na początku zajęć, specyfikacją