A college decides to digitise its library's loan system. The college library currently has two types of items for loan: books and compact discs (CDs). The college intends to implement the college's library's loan system using object-oriented programming (OOP).

For each of the sub-tasks, add a comment statement, at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In [1]:    # Task 1.1
           Program code
           Output:
```

Task 1.1

Registered library users have a `user_id` which they use for all transactions. Library items can be loaned for three weeks.

The class `LibraryItem` will store the following data:

- `item_id` – stored as a string
- `title` – stored as a string
- `loaned_by` – the `user_id` of the user whom the item is loaned to, or an empty string if the item is not on loan
- `due_date` – stored as a date with the format YYYY-MM-DD

The class has four methods defined on it:

- `is_on_loan()` – returns False if `loaned_by` is empty, otherwise returns True
- `return_item()` – sets `loaned_by` to empty string
- `loan_to(user_id)` – returns a Boolean value to indicate whether the user has been loaned a library item successfully
  A library item that is already on loan cannot be loaned out again
- `print_details()`
  - display `title` and `due_date` of the `LibraryItem` in separate lines.
  - display `user_id` of the user(s) loaning the item, if any. E.g.

```
Title: Thinking with type
Loaned by: S1111111H
Due date: 2022-09-03
```

The `timedelta` library is built into Python and can be used to perform arithmetic operations on dates. Example code is shown in `Task2_timedelta.py`.

Write program code in Python to define the base class `LibraryItem`. [9]

Task 1.2

The `Book` class inherits from `LibraryItem`, such that:

- the following extra information is recorded:
  - `author` – stored as a string
  - `category` – stored as a string
  - `reserved_by` – the `user_id` of the user whom the item is reserved by, or an empty string if the item is not reserved
- they may be reserved, and reservation is handled through the following methods:
  - `is_reserved()` – returns False if `reserved_by` is empty, otherwise returns True
  - `release()` – sets `reserved_by` to empty string
  - `reserve_for(user_id)` – returns a Boolean value to indicate whether the user has reserved a library item successfully
    A library item that is already reserved cannot be reserved again.
- `print_details()` should display:
  - two additional attributes, `author` and `category`
  - `user_id` of the user(s) reserving the item, if any.

The `CompactDisc` class inherits from `LibraryItem`, such that:

- the following extra information is recorded
  - `artist` – stored as a string
  - `genre` – stored as a string
- `print_details()` should display two additional attributes, `genre` and `artist`.

Write program code to define the `Book` and `CD` subclasses. [6]

Task 1.3

The text file, `libraryitems.txt`, contains information for several books and CDs. Each row is a comma-separated list of information for a book or a CD. The item IDs for books begin with a B, while those for CDs begin with a C.

The information of a book is given in the following order:

```
item_id, title, author, category
```

The information of a CD is given in the following order:

```
item_id, title, artist, genre
```

Write program code to:

- read in the information from the text file, `libraryitems.txt`
- create an instance of the appropriate class for each library item
- display the information of each instance
- store the library items as a list of object instances. [5]

Task 1.4

User ID `S1111111H` borrows all the library items in the text file, `libraryitems.txt`.
User ID `S1222222D` reserves all the books that `S1111111H` has borrowed.

Write program code to carry out the above steps. Print out the details of all library items. [4]

Save your Jupyter notebook for Task 1.