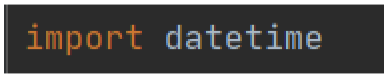
Task 6

Analyzing the code

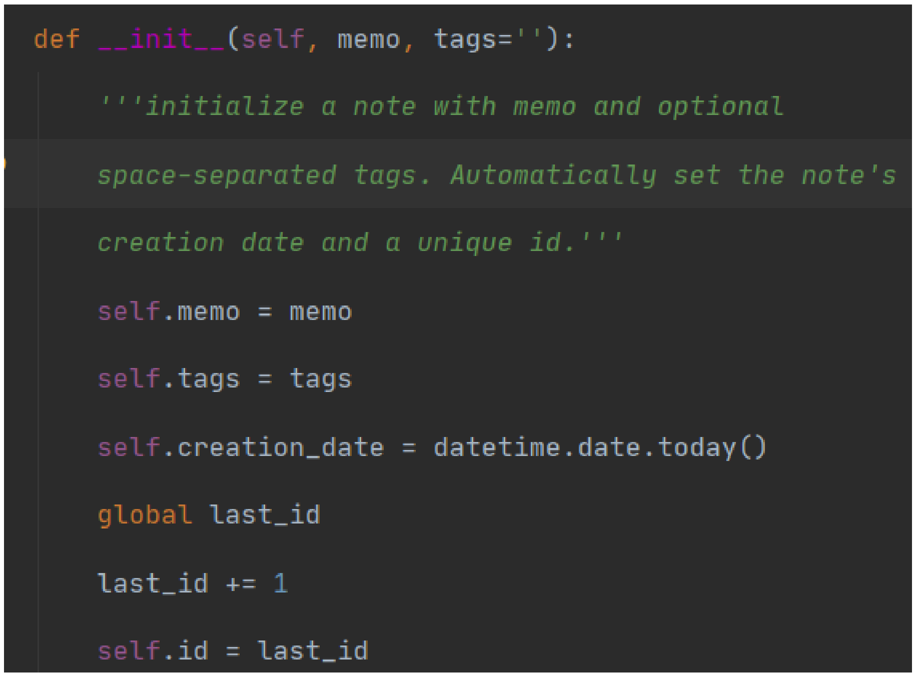
1.notebook.py



```python
import datetime
```

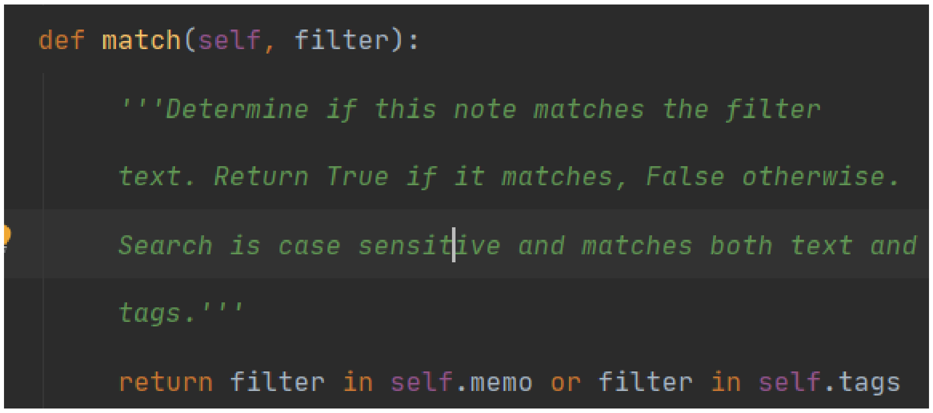
Imported this library to know when the memo was created

1.1 Class Note



```python
def __init__(self, memo, tags=''):
    '''initialize a note with memo and optional
    space-separated tags. Automatically set the note's
    creation date and a unique id.'''
    self.memo = memo
    self.tags = tags
    self.creation_date = datetime.date.today()
    global last_id
    last_id += 1
    self.id = last_id
```

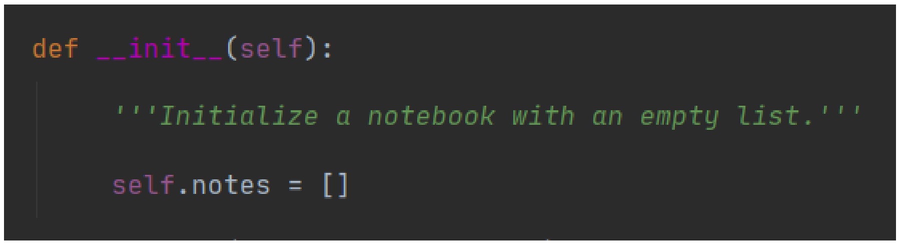
Initializing variables memo, tags, and creation date (that's where we use DateTime)



```
def match(self, filter):
    '''Determine if this note matches the filter
    text. Return True if it matches, False otherwise.
    Search is case sensitive and matches both text and
    tags.'''
    return filter in self.memo or filter in self.tags
```

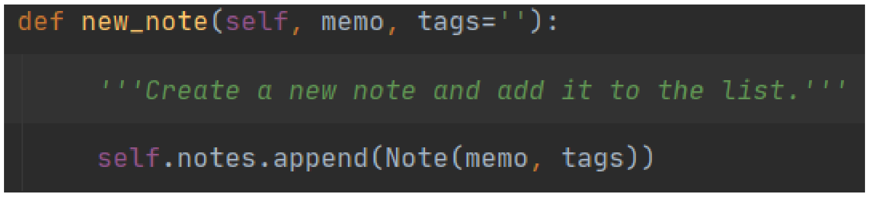
Match method checks if note matches the filter text, returns true if it matches, and False otherwise.

1.2 Class Notebook



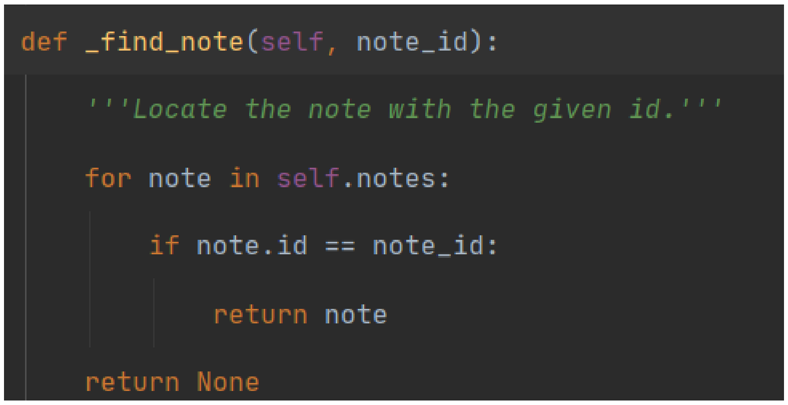
```
def __init__(self):
    '''Initialize a notebook with an empty list.'''
    self.notes = []
```

Creating a notebook as an empty list



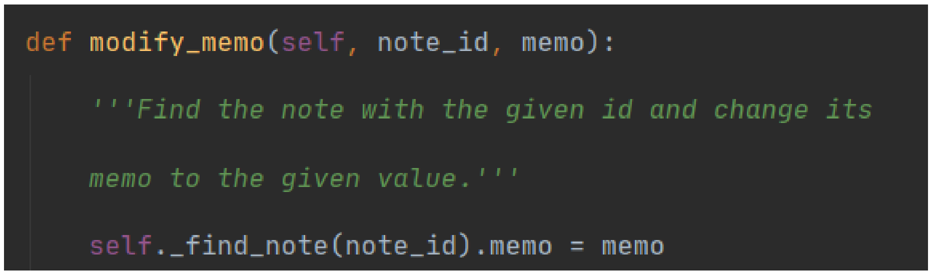
```
def new_note(self, memo, tags=''):
    '''Create a new note and add it to the list.'''
    self.notes.append(Note(memo, tags))
```

new_note method appends to previously created notes-list a memo with its tag



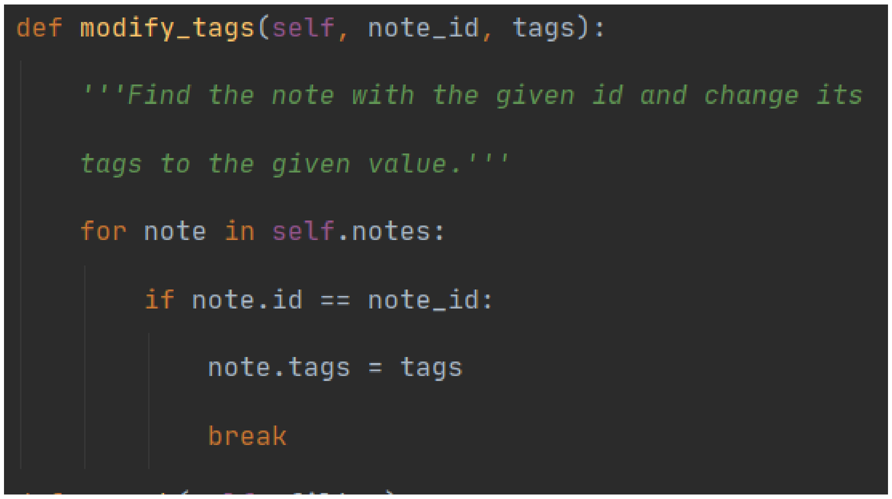
```
def _find_note(self, note_id):
    '''Locate the note with the given id.'''
    for note in self.notes:
        if note.id == note_id:
            return note
    return None
```

_find_note method finds notes by its given id. We iterate through the note list and check if our argument equals the existing id



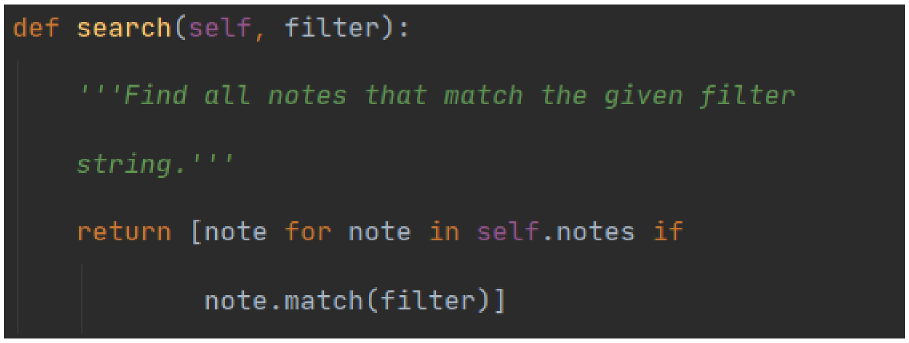
```python
def modify_memo(self, note_id, memo):
    '''Find the note with the given id and change its
    memo to the given value.'''
    self._find_note(note_id).memo = memo
```

modify_memo method is changing the memo to the new one



```python
def modify_tags(self, note_id, tags):
    '''Find the note with the given id and change its
    tags to the given value.'''
    for note in self.notes:
        if note.id == note_id:
            note.tags = tags
            break
```

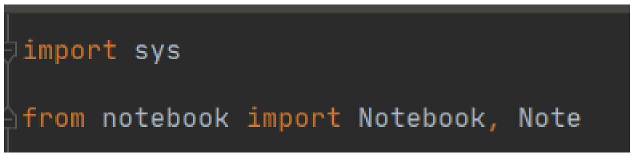
modify_tags method works similar to modify_memo



```python
def search(self, filter):
    '''Find all notes that match the given filter
    string.'''
    return [note for note in self.notes if
            note.match(filter)]
```

search method finds notes that match filter (list comprehension)

2.menu.py

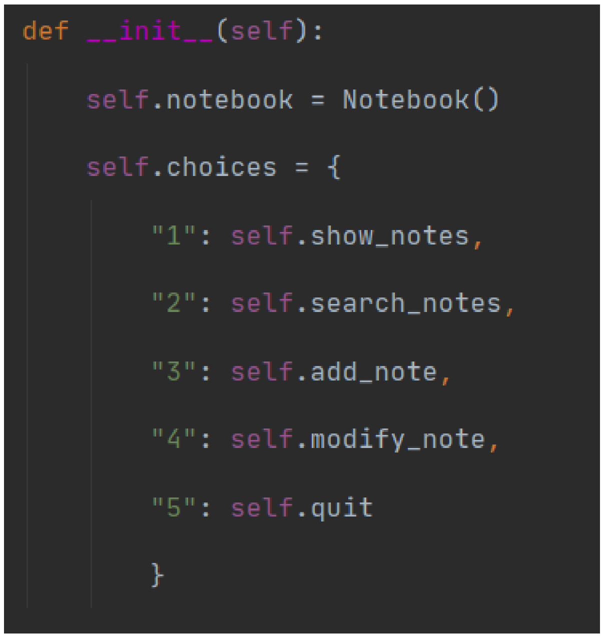


```python
import sys
from notebook import Notebook, Note
```

Importing sys to exit the program on a certain step

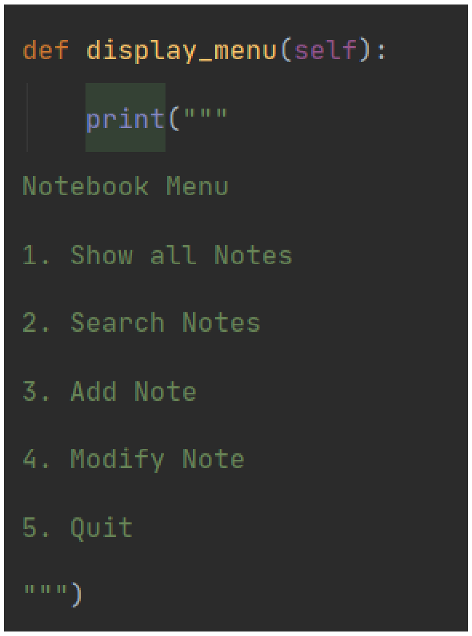
Importing Notebook and Note to work with these classes.

Class Menu



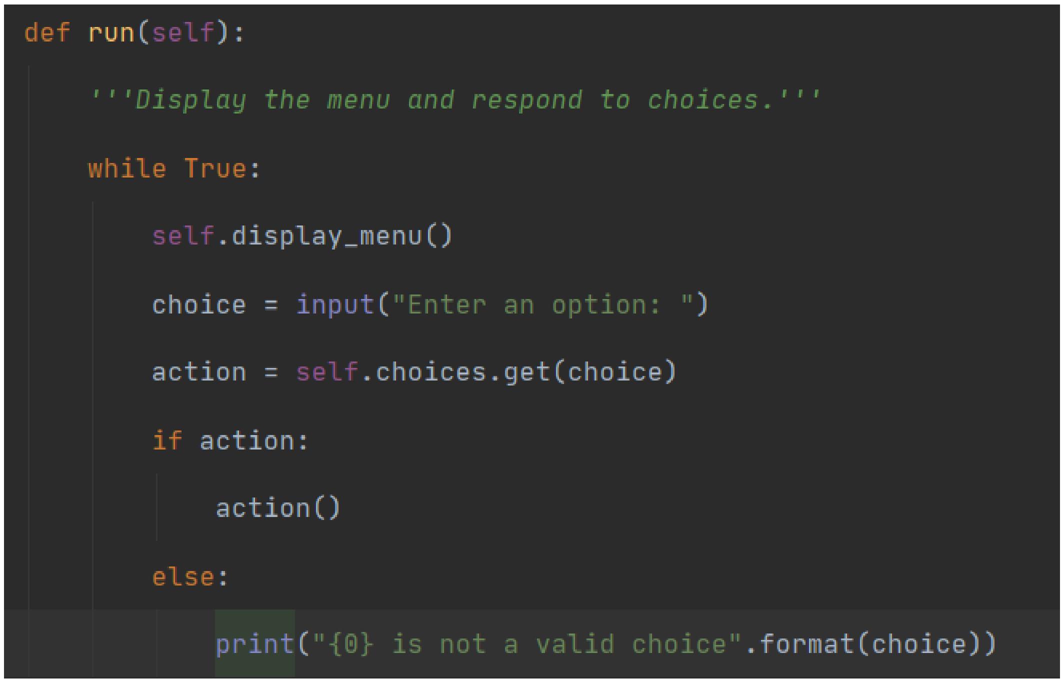
```python
def __init__(self):
    self.notebook = Notebook()
    self.choices = {
        "1": self.show_notes,
        "2": self.search_notes,
        "3": self.add_note,
        "4": self.modify_note,
        "5": self.quit
        }
```

Initializing variables notebook and choices(that variable was created for the user to interact with the program)



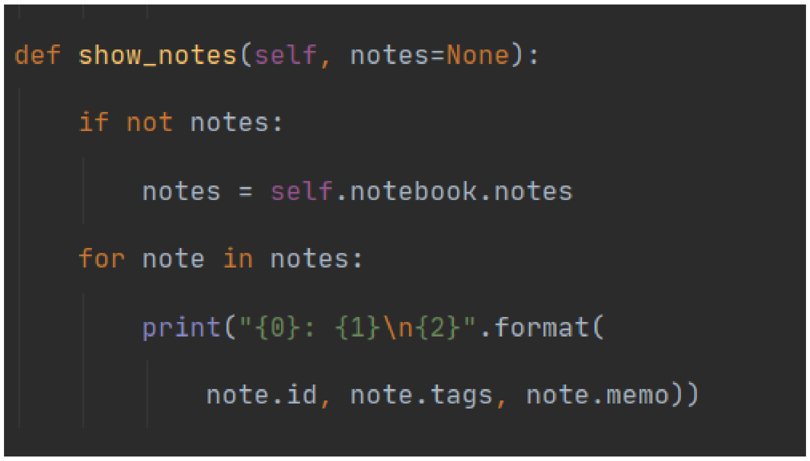
```python
def display_menu(self):
    print("""
Notebook Menu
1. Show all Notes
2. Search Notes
3. Add Note
4. Modify Note
5. Quit
""")
```

This method prints the menu



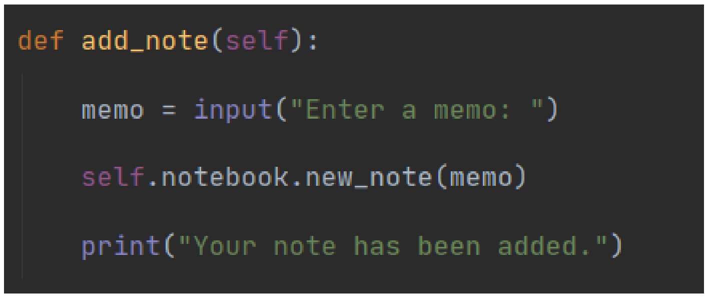
```python
def run(self):
    '''Display the menu and respond to choices.'''
    while True:
        self.display_menu()
        choice = input("Enter an option: ")
        action = self.choices.get(choice)
        if action:
            action()
        else:
            print("{0} is not a valid choice".format(choice))
```

The run method runs the introduction and allows users to choose an option of what he wants to do with the notebook



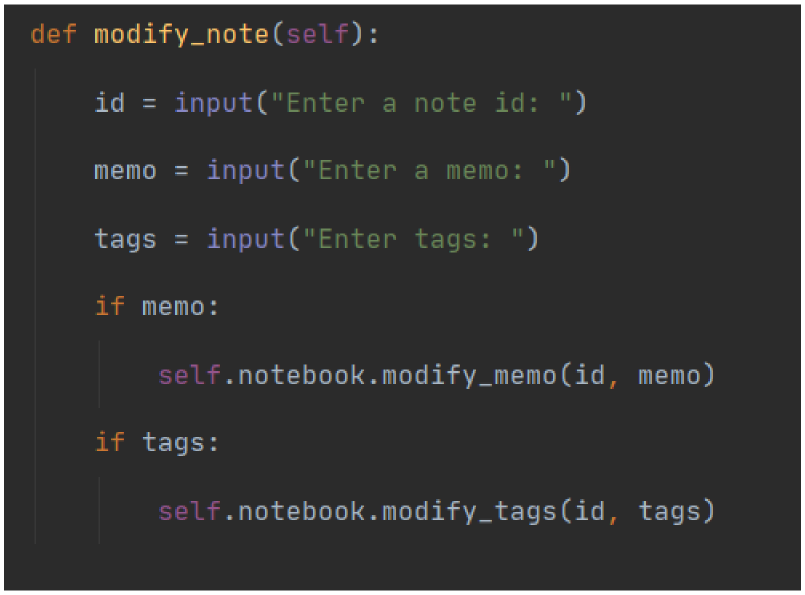
```python
def show_notes(self, notes=None):
    if not notes:
        notes = self.notebook.notes
    for note in notes:
        print("{0}: {1}\n{2}".format(
            note.id, note.tags, note.memo))
```

The show_notes method allows the user to see all of his notes



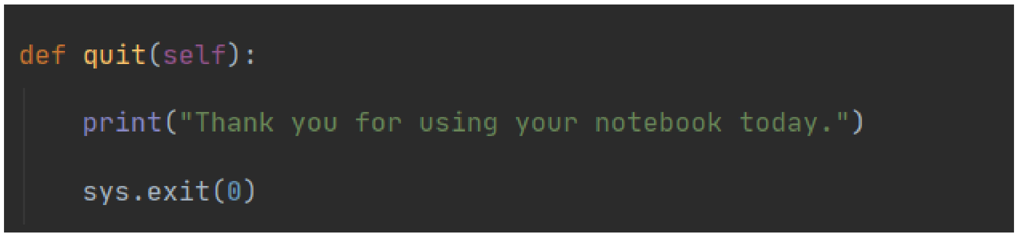
```python
def add_note(self):
    memo = input("Enter a memo: ")
    self.notebook.new_note(memo)
    print("Your note has been added.")
```

Adding notes to the notebook.



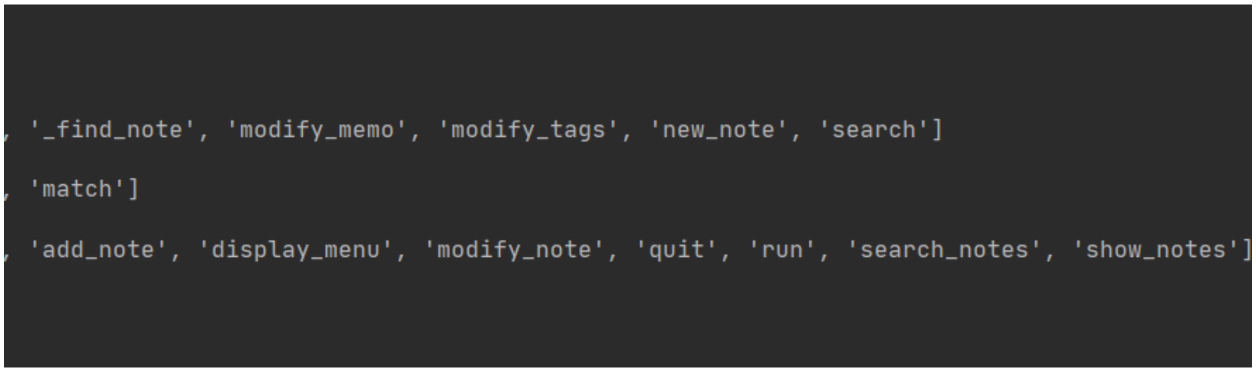
```python
def modify_note(self):
    id = input("Enter a note id: ")
    memo = input("Enter a memo: ")
    tags = input("Enter tags: ")
    if memo:
        self.notebook.modify_memo(id, memo)
    if tags:
        self.notebook.modify_tags(id, tags)
```

Modyfing the note



```
def quit(self):
    print("Thank you for using your notebook today.")
    sys.exit(0)
```

Quitting the program



```
, '_find_note', 'modify_memo', 'modify_tags', 'new_note', 'search']
, 'match']
, 'add_note', 'display_menu', 'modify_note', 'quit', 'run', 'search_notes', 'show_notes']
```

When running dir(Notebook), dir(Menu), we can see that the methods were added.