Maktab Sharif

Python | Main course

Quiz 1

- OOP
- Files
- Decorators
- Generators
- Scripting
- Datetimes

by Mohammad Amin H.B. Tehrani

www.maktabsharif.ir

Github (10 min)

Maktab Sharif

Create a repository in github (**maktab52_quiz1**) as public, in the next 10 minutes.

- **Submission** time is considered the **Commit time.**
- **Clone** the repo from github, make directories for each problem we'll mention later.
- Push the commits up to **5 minutes** after the quiz.

Problem 1 (15 min)

Maktab Sharif

Download **books.dill** (or books.pkl) file, load list data from it, then:

1. Sort the list **by ISBN**.
2. Sort the list **by book name**.
3. Sort the list **by publish date**, **reversed**.
4. Filter the list **by author = 'George Orwell'**
5. Filter the list **by publisher = 'Akbar pub' or 'Asqar pub'**
6. Filter the list **by publish date year >= 2001**

```
class Book:
    ISBN: int
    name: str
    author: str
    publisher: str
    publish_date: datetime.date

    def __init__(self, ISBN, name, author, publisher,
publish_date):
        self.ISBN = ISBN
        self.name = name
        self.author = author
        self.publisher = publisher
        self.publish_date = publish_date
```

Problem 2 (25 min)

Maktab Sharif

Use jdatetime & datetime modules and write codes below:

A. Write a context manager class **TimestampOpen** that appends the open jalali timestamp & close jalali timestamp of the file on closing that.

B. Write a **Iterator** class that gets 3 arguments **(start_date, end_date, week_day)** then, returns dates that correspond to the week_day between start_date & end_date.
Then write a argparser that gets:
   1. -s --start-date
   2. -e --end-date
   3. -w --week-date

   And prints the result.

Problem 3 (20 min)

Maktab Sharif

Write a program that reads a file, then write codes below:

1. **`duplicate_words_gen`**
   Gets argument **file_path**, read the file content, then Yields words that have duplicate letters.

2. **swapcase_decorator:** Gets a function (or generator) then returns a wrapper on the function(or generator) that **yield**s swapcased results.

3. **main:** Implement a arg parser, that gets only a positional argument **file_path** from the user, then prints swapcased duplicate_words of the target file.

```
def swapcase_decorator(gen):
    ...


@swapcase_decorator
def duplicate_words_gen(file_path):
    ...


if __name__ == '__main__':
    ...
```