Határidőnapló dok.

Fájlok:

- adatok.txt
  - Azokat az adatokat tartalmazza, amiket majd a program kezelni fog:
    - listázás
    - törlés
    - módosítás
    - hozzáadás
    - rendezés
- class_file.py
  - A beolvasáshoz szükséges class-t tartalmazza, amit később a *reading.py* fog használni
- myapp.py
  - Itt van a program lényegesebb pontja, függvények, importált modulok
  - Itt hívódnak meg a többi fájlok is, mint modulok:
    - reading.py
    - searching.py
    - sorter.py
    - sorted_to_index.py
    - months_to_index.py
    - Illetve a grafikus részhez használt beépített modul: *tkinter*
- Naplo(főprog).py
  - Itt hívódik meg a myapp.py-ben használt osztály, amibe szintén egy *tkinter* modul épül be.
  - Itt vannak felsorolva az esetleges segítséghez használt linkek, plágium elkerülése végett
- reading.py
  - Meghívja a *class_file.py*-t modulként
  - Itt történik az *adatok.txt* beolvasása, egy *Reading()* függvényben, majd egy listaként tér vissza
- searching.py
  - A *reading.py* hívódik meg modulként, majd az események neveit egy ismétlődés mentes listába rakja, hogy később, a *myapp.py*-ben könnyebben lehessen keresni.
- sorter.py
  - A *myapp.py*-hez a könnyebb rendezés érdekében 4 függvényt tartalmaz:
    - Név szerinti rendezéshez
    - Dátum szerinti rendezéshez
    - Idő szerinti rendezéshez
    - ID szerinti rendezés
  - Valamint, mivel a *class_file.py* adatait rendezi, így meg van hívva modulként
- sorted_to_index.py
  - A későbbi havi listázáson belül ezek szerint rendezi az adott hónaphoz felsorolt adatokat, egy szótárat tartalmaz
- months_to_index.py
  - A havi adatok listázásához, amit később a *myapp.py*-ben használunk tartalmaz egy szótárat.

Myapp.py

- class Myapp
  - A programban használt grafikus dolgok vannak definiálva:
    - gombok
    - label-ök
    - beviteli mezők
    - lista, amiben a kijelölt napra listázott elemek lesznek
    - egy táblázat, amiben a beolvasott adatok lesznek
    - legördülő menü, ami az adott hónap szerinti listázásért felelnek
    - legördülő menü, ami az adott hónap név, dátum, idő szerinti listázásáért felel
  - show()
    - A fentebb definiált „Kezdés” gombhoz van hozzárendelve, amely azt csinálja, hogy amint a user megnyomja, megjeleníti a fentebb definiált dolgok többségét, valamint lefuttatja a *fill_trv()* függvényt
  - sorted_name()
    - A táblázat „Név” oszlopához van rendelve a függvény, amely azt csinálja, hogy először kitöröl mindent a táblázatból, majd az adott oszlop szerint rendezve tölti vissza, itt hívódik meg a *sorter.py sorted_by_name()* függvénye
  - sorted_date()
    - A táblázat „Dátum” oszlopához van rendelve a függvény, amely azt csinálja, hogy először kitöröl mindent a táblázatból, majd az adott oszlop szerint rendezve tölti vissza, itt hívódik meg a *sorter.py sorted_by_date()* függvénye
  - sorted_date()
    - A táblázat „Idő” oszlopához van rendelve a függvény, amely azt csinálja, hogy először kitöröl mindent a táblázatból, majd az adott oszlop szerint rendezve tölti vissza, itt hívódik meg a *sorter.py sorted_by_date()* függvénye
  - sorted_list_by()
    - Itt hívódik meg a *months_to_index.py* illetve a *sorted_to_index.py*
    - A fentebb definiált *self.sorterer* legördülő menühöz van hozzárendelve, hogy a már kiválasztott hónapot lehessen rendezni 1:Név,2:Dátum,3:Idő szerint.
    - Minden egyes alkalommal kitörli először a táblázatot, majd visszaírja kiválasztott rendezés szerint
  - selected()
    - Kitörli a táblázat összes elemét, majd a *self.drop* menüből kiválasztott hónap szerint listázza az adatokat, itt hívódik meg a *months_to_index.py* is
  - update_list()
    - Minden egyes alkalommal, amikor új adat kerül, módosul, esetleg törlődik, lefut ez a függvény, hogy az új/módosított adatokat is tartalmazza, illetve a törölt adatokat ne tartalmazza már a táblázat
  - seleceted_to_list()
    - A táblázat adott sorára kattintva lefut az a függvény, majd ahhoz a naphoz, amit a kiválasztott sor tartalmaz, egy listába kiírja a nap többi időpontjához társított eseményeket, mind ezt persze időrendben
  - searching()
    - A fentebb definiált self.*search_entrty*-be írt adatot lekérdezi egy változóba, majd a *searching.py*-ben lévő adatokkal összehasonlítja, és a megegyezőket belerakja egy listába, amivel majd visszatér

- list_searched()
    - Kitörli az összes adatot a táblázatból, majd a *searching()*-ből visszaadott lista elemeit összeveti a beolvasott adatok neveivel, és az egyezők minden adatával feltölti ismét a táblázatot, mindezt dátum szerint rendezve
- clear_entry()
    - A definiált beviteli mezők tartalmát kitörli használat után, a nem megjelenítettekét is
- get_selected_information()
    - Amint duplán kattintuk a táblázat kijelölt elemére, kitölti az adott kijelölés adataival a megjelenített és nem megjelenített beviteli mezőket.
- add_item_command()
    - Lekéri megjelenített beviteli mezőkből a hozzáadni kívánt adatot
    - Az adatok ID-ját beleírja egy lista, majd annak a listának a maximum eleméhez ad hozzá egyet, hogy 2 adatnak ne legyen ugyan az az ID-ja
    - A lekért adatokat egy hasonló sorrá fűzi, mint amikből az *adatok.txt* áll, a már megnövelt ID-val (a későbbi használat miatt kell, hogy hasonló sor legyen)
    - Megnyitja az *adatok.txt*-t hozzáfűzésre, majd a végére írja a létrehozott sort, majd bezárja
    - Végül lefuttatja a *clear_entry()* és az *update_list()* függvényeket
    - Majd ezt a függvényt a *self.add_item* gombhoz rendelem hozzá
- delete_item_command()
    - Lekéri a megjelenített beviteli mezőkből a törölni kívánt adatot (mivel pontos egyezés alapján töröl, így a duplakattintás a mezők kitöltéséhez ajánlott)
    - A lekért elemeket a beolvasott adatokhoz hasonló sorba fűzi
    - Majd kitörli a kiválasztott sort a táblázatból
    - Beolvassa az *adatok.txt* és a sorait egy listába teszi, majd a lista elemeit összehasonlítja a fentebb összefűzött változóval, és ha egyezőt talál, akkor azon kívül minden mást beleír az *adatok.txt*-be
    - Végül lefuttatja a *clear_entry()* és az *update_list()* függvényt
    - Majd ezt a függvényt rendelem hozzá a fentebb definiált *self.delete_item* gombhoz
- update_item_command()
    - Igazság szerint ezt a függvényt 2 felé lehetne bontani:
        - 1. fele: hasonlóképpen működik, mint a *delete_item_command()* függvény, annyi különbséggel, hogy a nem megjelenített bevitali mezők adatait kérdezi le
        - 2. fele: hasonlóképpen működik, mint az *add_item_command()* függvény, annyi különbséggel, hogy nem növeli az adatok ID-ját, mivel csak a meglévőt cseréltük le más adatokra, de az ID maradt
    - Ezt pedig a fentebb definiált self.*update_item* gombhoz rendelem hozzá
- Több rejtett elemet is tartalmaz a UI, amihez alapból a felhasználó nem fér hozzá, főleg a módosítás és törlés miatt.
    - beviteli mezők
    - újralistázó gomb, ami a keresés utáni vagy hónaponkénti listázás utáni összes elem kilistázását teszi lehetővé, későbbi használat érdekében

- A hibakezelések/hibák jelzésére egy error labelt is tartalmaz a program amellett a gomb mellett, amely egy külön csv-be történő exportálást végez, hogy később akár más célra is lehessen használni akár a txt-ben lévő adatokat.
- export_to_csv():
    - Egy csv-be történő exportálást végzi *pandas* modul segítségével, így a program használatához szükséges letölteni ezt a modult (pip install pandas)
    - az error label-t módosítja, hogy tudja a felhasználó, hogy sikeres volt az export