Projekt Feladat

Beadási határidő:

2025.08.31. 23:59:59

Követelmények

A nyelv lehetőségeinek kihasználása: strukturált felépítés, több modulra bontás, fájlkezelés, osztályok és adatszerkezetek.

A program mellé el kell készüljön a programozói és a felhasználói dokumentáció.

Itt azt kell bizonyítani, hogy az adatszerkezetek, algoritmusok használatát sikeresen elsajátítottátok, nem pedig azt, hogy ügyesen és gyorsan oldotok meg problémákat külső könyvtárak és AI használatával.

Az *osztályok és adatszerkezetek* tekintetében: a programnak összetett adatszerkezetet kell építenie, pl. objektumok listáját vagy listák listáját. Ez nem helyettesíthető fájlműveletekkel, nem váltható ki egyéb nyelvi elemekkel.

Nem tárolhatóak egy entitás adatai listában (pl. **lista[0]** egy ember neve, **lista[1]** a születési dátuma, **lista[2]** a lakcíme, ...) saját osztályok definiálása helyett. Ezeket az adatszerkezeteket *a függvények közt paraméterátadással és visszatérési értékekkel* kell átadnia a programnak, nem használhat indokolatlanul globális változókat.

A programnak érdemben *adatfeldolgozást kell végeznie* az így felépített adatszerkezeten. Ez azt jelenti, hogy a program lényegi funkcióját saját Python kóddal kell megvalósítani.

(Nem alkalmas projekt feladatnak pl. egy képnézegető program, amely nem saját Python modullal jeleníti meg vagy dolgozza fel a képeket. Alkalmas viszont egy aknakereső, amely ugyan maga is megnyit képeket a pálya megjelenítéséhez, de ez csak mellékes része a programnak.)

A *fájlkezelésben* olyan adatokat kell tudni kezelni, amelyek száma változik, vagy változhatna. Nem teljesíti a követelményt pl. egy olyan program, amelyik az elindításainak számát (egyetlen egy egész számot) tárol fájlban.

Teljesíti viszont egy olyan, amelyik egy játék 10 elemű dicsőséglistáját tárolja. A 10 ugyan fix, de akár változhatna is.

A fájlkezelés saját programkódon kell alapuljon, pl. grafikus könyvtár által betöltött kép nem számít saját fájlkezelésnek.

A fentiek között *logikai és* kapcsolat értendő: ha bármelyik hiányzik, a projekt feladat nem elfogadható.

Projekt feladat részfeladatai

PF 1. – választás

Ez a részfeladat a feladat kiválasztását jelenti. A megadott feladatok közül is lehet választani, vagy saját, hasonló nehézségű feladatot is lehet hozni. A kiválasztott feladat utólag már nem módosítható.

PF 2. – specifikáció

Ez a kiadott feladat részletekbe menő pontosítása. A pontosítás célja az, hogy a program megrendelője, és a programot elkészítő programozó ugyanarra gondoljon, és ne a munka végén derüljön fény a félreértésekre. Ide tartozik a program feladatának leírása, a bemenetek és a várt kimenetek rögzítése, a program használatának leírása is. A specifikációban még nem kell a program belső felépítésével, működésével kapcsolatos részleteket megadni.

A pontosított specifikáció részletességre jellemző, hogy ha két külön programozó elkészíti a programot ugyanabból a specifikációból, de egymástól függetlenül dolgozva, akkor kívülről nézve nagyon hasonló programoknak kell keletkezniük. Amennyiben az eredeti, rövid specifikáció bármelyik része nem egyértelmű, akkor a pontosítás során egy lehetőség mellett dönteni kell.

PF 3. – félkész megoldás

A félkész megoldás lényege az, hogy a készülő program funkcionalitásának egy részét bemutassa: már látszania kell rajta akár felhasználói, akár programozói szemmel, hogy mi készül. Nem elegendő egy „hellóvilág” programot, vagy egy menüt feltölteni, a félkész változatnak valamennyire már működnie kell. Még nem kell hibamentes legyen, nem kell tartalmaznia az összes funkciót, sem felépítésében és használt adatszerkezeteiben nem kell a végleges programmal megegyezzen.

A félkész feladatnak kezdetleges dokumentációt már tartalmaznia kell: legalább egy rövid, fél-egy oldalas leírást arról, hogy mi az, ami már működik, és hogy az egyes feltöltött fájloknak, függvényeknek mi a szerepe.

Néhány példa. Kukacos játék esetén félkész feladat lehet egy olyan program, amelyikben a kukac feje már mozog a képernyőn, és az étkeket össze lehet gyűjteni, de a kukacnak még nincsen farka, és nem nő. Telefonkönyves feladat esetén félkész változat lehet az, ahol az adatokat már be tudja kérni a program, de nem tudja eltárolni, kereséseket még nem tud végezni, vagy képernyőre írja azt, amit amúgy fájlba mentene.

PF 4. – végleges program

Ez az elkészült, végleges program leadása, forráskódokkal és dokumentációkkal.

A projekt feladat beadása

PF 1. – választás

A listából választott feladatok esetén a feladatkiírás vagy a cím bemásolásával választható ki a feladat. Saját feladat esetén röviden el kell magyarázni a program lényegét.

PF 2-3-4. – programkódok és dokumentációk

A dokumentációkat és a forráskódot *elektronikusan kell leadni.* A megoldás forrásfájljait és dokumentációit kell feltölteni egy ZIP fájlban. A feltöltés formátumára az alábbi megkötések vonatkoznak:

- A csomag ZIP formátumú, maximális mérete 1 MB forráskóddal és dokumentációval együtt.
- A megoldás forrásfájljait *.py nevű szöveges fájlként kell leadni.
- A specifikációt (PF 2.) és a dokumentációt (PF 3-4.) PDF formátumban kell leadni.

Viták elkerülése végett

- A rossz formátumban vagy hiányosan feltöltött, feltölteni próbált megoldások szintén nem elfogadhatóak.
- A feltöltött csomag felesleges fájlokat nem tartalmazhat.
- Ha túl nagyok a PDF-ek, akkor javasolt:
  - nem használni rengetegféle betűtípust (a tartalmat pontozzuk, nem a díszítéseket)
  - egy PDF fájlban beadni a programozói és a felhasználói dokumentációt.
- Ha a feltöltött csomag hiányos, vagy a programban alapvető technikai hiányosságok vannak, akkor az óraadó/tanár/gyakorlatvezető utólag is megtagadhatja az elfogadást. A hibás feltöltés, követelményeknek nem megfelelés nem az óraadó/tanár/gyakorlatvezető hibája, akkor sem, ha nem jelezte azonnal.
- A projekt feladat fájljaiért, mint adatvagyonért mindenki saját maga felelős. Rendszeres biztonsági mentést kell készíteni róla. „Meghalt a vinyóm, vírusos lett a gépem”, és ezekhez hasonló indokokkal sem fogadunk el késést. Javasolt felhőszolgáltatások használata. Ha valaki tudja mi az a verziókezelő, használja bátran, de nagyon figyeljen arra, hogy a repository privát legyen!

Végleges megoldás értékelése

Általános követelmény a programmal szemben az, hogy a józan ész elvárásai szerint működjön. A programnak olyan magától értetődő képességekkel is kellhet rendelkeznie, amelyek a specifikációban külön nincsenek rögzítve. Például ha a specifikáció annyit mond, hogy a program egy nevet megjegyez, akkor elvárható az is, hogy a névben lehessen szóköz karakter. Vagy ha a specifikáció azt mondja, hogy a program bizonyos adatokat fájlba tud menteni, akkor elvárás az is, hogy vissza is tudja tölteni azokat.

Automatikusan elutasításra kerül

Vannak olyan elvi hibák és alapvető hiányosságok, amelyek esetén a beadás egyáltalán nem elfogadható. Érdemes a következő ellenőrzési listán végigmenni beadás előtt:

- Nincs feltöltve a forráskód megfelelő formátumban. Esetleg csak link, megjegyzés van a forráskód helyett, vagy csak a fejlesztőkörnyezet projektfájlja.
- Hiányzik a dokumentáció. Ha a felhasználói dokumentáció lényegében megegyezik a specifikációval, akkor is át kell szerkeszteni és fel kell tölteni.
- A program leglényegesebb funkciói nem működnek, nem valósítja meg a specifikációját.
- Az egész program egy forrásfájlban van, nincsen több modulra bontva.
- A program nem használ fájlkezelést.
- A program nem definiál saját osztályokat a tárolandó adatok modellezéséhez.
- A program nem épít adatszerkezeteket, esetleg fájlműveletekkel próbálja kiváltani azokat.
- A program indokolatlanul, túlzóan használ globális változókat a függvényparaméterek és visszatérési értékek helyes használata helyett.
- A program a függvényhívást ciklusként használja: pl. a menu() meghívja az adatbevitel()-t, az pedig visszatérés nélkül meghívja a menu()-t, hogy az meghívhassa a keres()-t, ami megint meghívja a menu()-t stb. (Gyanús, ha túl sok helyen szerepel sys.exit() hívás a programban, másképp nem oldható meg a leállítása!)

Programkód minősége

- modulokra bontás minősége (tipikusan a main.py+fuggvenyek.py nem elég; pl. adatszerkezeteket kezelő modul, grafika modul, különféle menüket megjelenítő modul stb.)
- funkcionális dekompozíció minősége (ne legyenek túl nagy függvények, és ne hívják egymást következetlen módon, pl. beolvasás a menüt)
- adatszerkezetek, típusok használata (tipikusan: osztály bekerültek, amik összetartoznak; nem szamlalo, nevezo, hanem class Tort; nem rajzol(palya, szelesseg, magassag),

hanem rajzol(palya), ahol az egy osztály; általában a túl sok paraméterű függvényekből látszik, hogy baj van)
- helyes erőforráskezelés (fájlok bezárása, kivételek kezelésével vagy with blokkokkal)
- nyelvi elemek helyes használata (ne legyen sok mágikus szám vagy sztring, helyes vezérlési szerkezetek, nincs from module import *)
- a tárgyban elvárt kódolási stílust követi (van main() függvény, osztályok adattagjai már a konstruktorban létre vannak hozva)
- nincsen telis-tele indokolatlanul újraimplementált szabványos függvénnyel
- szerep szerinti változónevek, függvénynevek (nem a betűk száma a lényeg, lehetnek 1 betűsek; vmi, logikai – ezek nem)
    - feladatfüggő pont, a feladat jellege alapján (tipikusan egyéb hibáért, hatékonytalanságért levonható, főleg ha az adott problémának van tanult megoldása; laborvezető indokolja)
    - feladatfüggő pont, mint a fenti
    - feladatfüggő pont, mint a fenti

Dokumentáció általában

- a dokumentációk bekezdésekre és fejezetekre vannak tagolva, elfogadható a helyesírásuk, nincsenek bennük képként forráskódok, nincsenek felfújva (16-os betűméret, 4 cm margó, dupla sorköz, egyéb terjedelemnövelő technikák).

Programozói dokumentáció

- a projekt felépítése (melyik fájl mit tartalmaz) és a szükséges környezet, külső könyvtárak leírása, telepítéshez szükséges lépések leírása (pl. grafikus könyvtár; ha nem igényel szabványos könyvtárakon kívül semmit, akkor ez a tény)
- adatszerkezetek dokumentációja, tervezési megfontolások leírása (melyik típus és adatszerkezet mire való, mit tárol, miért arra esett a választás)
- a függvények dokumentációja (feladat, paraméterek, visszatérés, esetleg körülmények, pl. xy nem lehet None)

Felhasználói dokumentáció

- program feladata, célja; milyen bemenetek, kimenetek vannak, játéknál hogyan kell irányítani

Határidők betartása

- Projekt időben kiválasztva *2025.08.10.* (PF1 elfogadva és PF4 elfogadva)
- Specifikáció időben elkészült *2025.08.17.* (PF2 elfogadva és PF4 elfogadva)
- Félkész projekt időben elkészült *2025.08.24.* (PF3 elfogadva és PF4 elfogadva)
- Végleges projekt időben elkészült *2025.08.31.* (PF4 elfogadva)