Név: Doszpod Patrik
NK: IOS9WL

# NHF 4. – Programozói dokumentáció

## Bevezetés

A játékok (akár videojátékról, akár fizikai tárgyról beszélünk) világában – is – egyre nagyobb népszerűségnek örved a felhasználói igények, egész fejlesztés során figyelemmel követése. Ezen okból döntöttem úgy, hogy a Legyen Ön is Milliomos c. játék mintájára készített programom grafikus megjelenítést fog alkalmazni, ezáltal növelve a felhasználói élményt.

Kutatással nem tudom alátámasztani, hogy a grafikus megjelenítés alkalmazása videojátékoknál bizonyítottan növeli a felhasználói élményt, azonban saját, és a közvetlen környezetemben lévők véleménye alapján úgy határoztam, hogy szükség van rá.

Ehhez első körben szükségem volt a Python egy beépített, egyszerűen telepíthető könyvtárára, mellyel az általam elképzelt grafikák megvalósítását lehetségesnek tartottam a feladat kiválasztásakor. Ez a multimédiás könyvtár az ún. pygame, egy platformfüggetlen modul, mely képes átfedni a különböző operációs rendszerek közti különbségeket, és egységesen futtathatóak az ennek a segítségével készült programok Linuxon, Windows-on vagy akár Mac OS X-en is – melynek előfeltétele a Python, és a pygame modul megfelelő installálása.

A programot Windows operációs rendszerben írtam, ebben az esetben a pygame telepítése a „pip install pygame” paranccsal történt.

Miután a pygame rendelkezésemre állt, szükségem volt pár kép fájlra, amelyeket beépítettem a megfelelő helyre (ilyen pl. a híres „Legyen Ön is Milliomos” logo). Ezek mindegyikének forrását megjelöltem a dokumentáció végén.

## Program futtatása

A program futtatásához a bevezetésben megjelölt könyvtáron, és fájlokon kívül szükségesek még a különböző, adatszerkezetként funkcionáló „.txt” fájlok. Ezekről később, az adatszerkezet tárgyalásakor szót ejtek még.

Név: Doszpod Patrik
NK: IOS9WL

# A Program felépítése

A programot 6, részben, vagy egészben egymásra épülő modulból építettem fel. Ezek a következők:

1.) NHF_4_foprogram.py
2.) NHF_main_menu.py
3.) NHF_gameplay_v1.py
4.) NHF_dicsoseglista_v1.py
5.) NHF_adatszerk_v1.py
6.) NHF_UI_v1.py

A következőkben a modulokat, azok felépítését és működését tárgyalom.

## Adatszerkezet – NHF_adatszerk_v1.py

A modul logikája a következő:

- importálja a pygame beépített random modulját
- egy txt fájlt soronként olvas be - „loimszoveg.txt" –, majd a megfelelő formátumra hozott Kerdes objektumokat tömbben tárolja
- a random modul segítségével előállít adott nehézségi szinten 15 különböző kérdést
- a modul önmagában nem működik, az itt tárolt függvényeket meg kell hívnia más modulnak (NHF_4_foprogram, NHF_gameplay_v1, NHF_dicsoseglista_v1)

Az első lépés a logikai tervezet alapján tehát a random modul importálása, melyet a forráskód legelején „import random" kóddal tettem meg.

Ezután szükség volt az említett „Kerdes" osztály létrehozására, amelynek célja az, hogy egy adott kérdés adatait egy helyen lehessen tárolni:

```
class Kerdes:
    def __init__(self, nehezseg, kerdes, a, b, c, d, valasz, kategoria):
        self.nehezseg = nehezseg
        self.kerdes = kerdes
        self.a = a
        self.b = b
        self.c = c
        self.d = d
```

```
        self.valasz = valasz
        self.kategoria = kategoria
```

Ezeket az attribútumok szándékosan beszédesre terveztem, hogy későbbiekben könnyedén tudjak rá hivatkozni. (Az „a", „b", „c" és „d" attribútumok nem feltétlenül egyértelműek: ezekben tárolódik a az adott betűjelhez tartozó válaszlehetőség.).

A „Kerdes" osztály elkészítését követően a függvények megalkotására volt szükség, azzal a céllal, hogy a főprogram számára már egy olyan tömb típusú adatszerkezetet szolgáltasson, amelyben az adott körre vonatkozó „Kerdes" objektumok kerültek tárolásra.

Még mielőtt a függvényeket ismertetném fontos leszögeznem az adatszerkezet típusának választási okát. A döntés mögött az áll, hogy olyan adatszerkezetre volt szükség, amely könnyen és hatékonyan kereshető*, indexelhető*, és tartalma módosítható adott helyen. Ezekhez a kritériumokhoz igazodva döntöttem a tömb típus mellett.

*Az adatszerkezet függvényeinek ismertetése:*

1.) pontlista(): egysoros függvény, visszaadja a pontszámokat tartalmazó tömböt (5000-től 40 millióig, a játékszabály szerinti léptékben és darabszámban).
2.) beolvas(fajl): paraméterként egy txt fájlt kell kapnia, visszatérési értéke egy tömb, amely az átalakított sorokat tartalmazza.
3.) atalakit(adat): paraméterként egy sztring típusú adatot kell kapnia, amelyet átalakít Kerdes objektummá, és ez egyben a visszatérési értéke is.
4.) lehetseges_kerdesek(szint): paraméterként a nehézségi szintet kell megkapnia. Az összes kérdést az „*osszes_kerdes*" változóban tárolja, úgy, hogy meghívja rá referenciaként a beolvas(fajl) függvényt a „loimszoveg.txt"-vel. Miután az összes kérdés tárolva lett, a megfelelő szintű kérdések kiválasztása egy for ciklussal történik. A ciklus len(osszes_kerdes)-ig tart, és abban az esetben, ha a vizsgált indexű elem (amely egy Kerdes objektum) nehezseg attribútuma a szintnek megfelelő tartományon belül van, akkor hozzáadja a kerdesek nevű, kezdetben üres tömbhöz . Ez a tömb lesz a függvény visszatérési értéke.
5.) veletlenszeru_kerdesek(lista): paraméterként egy tömböt kap. Elsőként létrehoz egy üres, „jatek_kerdesei" nevű tömböt, majd egy for ciklus futása során hozzáadja a játék adott körben résztvevő Kerdes objektumait a tömbhöz. A visszatérési érték szintén a „jatek_kerdesei" referencia.
6.) kezdo(): Az előzőekben felsorolt függvények meghívásával adja vissza a játék kérdéseit. Ilyen például, ha a felhasználói interakció következtében kezdő szint kerül kiválasztásra, akkor a függvényhívás így néz ki: `veletlenszeru_kerdesek(lehetseges_kerdesek([1,2,3,4,5]))`

Név: Doszpod Patrik
NK: IOS9WL

# A felhasználói felület – NHF_UI_v1.py

A modul logikája a következő:

- importálja a sys, pygame és pygame.gfxdraw modulokat, a különböző grafikai objektumok megjelenítésének érdekében
- Az első és legfontosabb jellemző: a Rubrika osztály létrehozása
- különböző grafikai elemek megjelenítése képernyőn – ilyenek a négyzetek, logok és színek
- a modul önmagában nem működik, az itt tárolt függvényeket meg kell hívnia más modulnak (NHF_4_foprogram, NHF_gameplay_v1, NHF_dicsoseglista_v1)

A felhasználói felületért felelős modul (mostantól csak: UI) kulcsa a fent említett Rubrika objektum bevezetése:

```
class Rubrika:
    def __init__(self, x, y, a, b, szin, szoveg):
        self.x = x
        self.y = y
        self.a = a
        self.b = b
        self.szin = szin
        self.szoveg = szoveg

    def negyzet(self, kepernyo):
        return pygame.Rect(self.x, self.y, self.a, self.b)
```

Az x és y attribútumokban az objektum pozíciójának koordinátái, az a és b-ben a szélessége és magassága, a szin-ben a rubrika színe, a szoveg-ben pedig a rubrikában lévő szöveg kerül tárolásra. A Rubrika objektum szó szerint egy rubrikának a paramétereit hivatott tárolni, mint például a kérdésé, vagy a válaszlehetőségeké. Ezen objektumok aktív használata, és felhasználói inputra történő változásai viszik előre a játékmenetet.
Ezekről az NHF_gameplay_v1.py modul tárgyalásakor részletesebben kitérek.
Egy beépített függvényt is tartalmaz, a negyzet(kepernyo) függvény paraméterként az adott felhasználói felület képernyőjét kell kapnia, és visszatérési értéke egy pygame.Rect típusú tároló.
*A Rubrika objektumra későbbiekben csak rubrika-ként fogok hivatkozni.*
A UI-hoz tartozó függvények során sokszor előfordul paraméterként a „kepernyo”. Ez egy `pygame.display.set_mode((x, y))` pygame függvénnyel létrehozott képernyőt jelöl, és típusa pygame.Surface() beépített pygame típus.
A játékban ez adja a felhasználói felület helyét (pontosabban egy 1024*720 pixeles ablakot, amelyeken megjelennek az objektumok).

Név: Doszpod Patrik
NK: IOS9WL

*A későbbiekben a kepernyo paraméterre csak „képernyő”-ként fogok hivatkozni.*
A függvények bemutatása előtt fontos még megemlítenem, hogy a különböző méreteket és koordinátákat jelölő számok az egyes függvényekben nem véletlenül lettek így meghatározva, így nem kerek számok találhatók a forráskódban, azonban a megjelenítés szimmetrikus, könnyen befogadható látványt nyújt.
Fontos jellemző még a UI néhány függvényénél, hogy el lett látva egy „statusz” paraméterrel. Ennek a jelentősége az, hogy felhasználói interakciótól függően más értéket vesz fel, és ez alapján mást csinál a függvény (pl. más színben rajzol ki egy objektumot). Ezekre részletesebben kitérek az NHF_gameplay_v1.py tárgyalásakor.

*A UI függvényeinek ismertetése:*

1.) logo(kepernyo): paraméterként egy képernyőt kell kapnia, majd fájlból beolvasott képfájlt logo = (pygame.image.load(„mil_logo.gif”)), és megjeleníti a képernyőn: kepernyo.blit(logo, (435, 60)), amely paraméterei az előzőleg beolvasott képfájl referenciája, és az x, y koordináták.
2.) inditas_negyzet(), dicsoseg_negyzet(), menu_kilepes_negyzet(): az NHF_main_menu.py modul funkcióinak helyét adja vissza pygame.Rect() típusban
3.) fomenu(kepernyo): paraméterként egy képernyőt kell kapnia, kirajzolja a főmenü funkcióinak rubrikáit és a logót. Első lépésként az egész képernyőt sötétkék színűre változtatja: kepernyo.fill((0, 0, 60)), ezután fomenu_elemek változóban tárolja a 3 funkció („Játék indítás”, „Dicsőséglista”, „Kilépés”) rubrikáit. Ezeket a rubrikákat a negyzet_rajzol(kepernyo, rubrika, teli) és a szoveg_negyzetbe(kepernyo, forras) függvények meghívásával teszi. Miután ezekkel végez, a pygame.display.update() függvény meghívásával frissíti a képernyő tartalmát.
4.) segitseg_rubrika(): visszatérési értéke a segítségeket (felezés és közönség) jelölő rubrikák (2 elemű tömb)
5.) segitseg_negyzet(): visszatérési értéke a segítségek helyét megadó paraméterek, későbbiekben a kattintási interakciók szempontjából lesz fontos.
6.) segitseg_logo(kepernyo, hely, fajl): a logo(kepernyo)-vel azonos működésű, a hely paramétert a segitseg_negyzet() függvény visszatérési értéke adja.
7.) negyzet_rajzol(kepernyo, rubrika, teli): paraméterként egy képernyőt, egy rubrikát és egy teli státuszjelző számot kell kapnia. A teli a korábban említett statusz-hoz hasonló elven működik, amennyiben 1-es értéket kap, akkor egy teli, színnel kitöltött négyzetet rajzol ki:
pygame.gfxdraw.box( ((kepernyo, rubrika.negyzet(kepernyo), rubrika.szin)
egy körvonallal együtt, amely értelemszerűen átlátszó négyzet, fehér körvonallal: pygame.gfxdraw.rectangle(kepernyo, pygame.Rect(rubrika.x, rubrika.y, rubrika.a+2, rubrika.b+2), pygame.Color(255, 255, 255))

Ha a teli == 0, akkor egy átlátszó rubrikát rajzol ki a pygame.gfxdraw.rectangle függvénnyel.
Ezen két beépített pygame függvény működése azonos, az eltérés a kitöltésben van: paraméterként kell kapnia egy pygame.Surface() típust – azaz képernyőt – azért, hogy meglegyen, hogy hova kerül az objektum. Ezenkívül kapnia kell egy pygame.Rect() típust, amely az elhelyezkedés paramétereit tartalmazza (ezt ebben az esetben a teli négyzet körvonalát leszámítva a Rubrika objektum beépített negyzet(kepernyo) függvénye adja. Végül pedig a kirajzolandó négyzet színére is szükség van, ezt szintén a Rubrika objektum tárolja (szin attribútum, amely egy pygame.Color() beépített típust tárol – a zárójelen belül RGB kód szerint adtam meg a színeket minden esetben.

8.) szoveg_negyzetbe(kepernyo, forras): paraméterként egy képernyőt és egy forrást kap. A forrás egy Rubrika objektum, melyből közvetlenül a szoveg, x, y és b attribútumok kerülnek felhasználásra azzal a céllal, hogy a kívánt szöveg a megfelelő helyen megjelenjen. A betűstílust a pygame beépített, pygame.font.SysFont("Bahnschrift SemiLight", 24) függvényével adtam meg, paraméterei a kívánt betűstílus és betűméret. A képernyőn a már korábban említett kepernyo.blit() jeleníti meg a tartalmat, ám ezesetben az első paramétere a szoveg változóban tárolt betustilus.render(forras.szoveg, 1, (255, 255, 255)) függvény, amely a forrás szövegből alkot egy pygame.Surface() típusú objektumot, majd a már korábban bemutatott kepernyo.blit() függvénnyel jeleníti azt meg a képernyőn.

9.) valasz_negyzet(): visszatérési értéke egy tömb, amely a válaszlehetőségek pygame.Rect() típusú referenciáját tárolja

10.) kerdes_valasz_betoltes(kepernyo, forras, statusz): paraméterként egy képernyőt, forrásszöveget (forras) és statuszt (integer típus) kell kapnia. A kérdéshez és kategóriához külön, a válaszokhoz tarozó rubrikák egy tömbben tárolásra kerülnek. Miután ez megtörtént a statusz ellenőrzése történik: ha a statusz == 0, nem történik semmi (pass), ha a statusz == 1, akkor a helyes választ tartalmazó rubrika színe zöld lesz (pygame.Color(50, 100, 0)), ha a statusz == 2, akkor a jó választ tartalmazó rubrika piros színű lesz (pygame.Color(150, 0, 0)). A statusz attól függően változik, hogy milyen a felhasználói input (ezt a későbbiekben bővebben kifejtem).
Miután az elágazásban ellenőrizve lett a statusz, a rubrikák képernyőre kirajzolása és szöveg megjelenítése történik, a korábban bemutatott függvények meghívásával.

11.) hatter(kepernyo): azon képek, és rubrikák kirajzolását végzi, amelyek a játék során mindig meg kell, hogy jelenjenek. Ilyen pl. a logo.

12.) pontok_betoltes(kepernyo, forras): a forras paraméter az adatszerkezetből meghívott pontlista, az épp soronkövetkező betöltéséért a játékmenet modulja felelős. Az adott szinthez tartozó pontszámot jeleníti meg a képernyőn. A játék szabályainak megfelelően a „garantált pénznyereményt” jelentő szintek aranysárga színben kerülnek kirajzolásra.

13.) idozito_keret(kepernyo, szam): a visszaszámláló rubrikát rajzolja ki, a benne lévő szöveget a szam paramétertől kapja, amely 60-0-ig csökkenő sorrendben jelenik meg a képernyőn – a játékmenet függvényében.
14.) kezdokepernyo(kepernyo): Miután a felhasználó a játék indítását választotta a főmenüben, ezt tölti be képként, a funkciók rubrikáit rajzolja ki.
15.) szint_negyzetek(), inditogomb_negyzet(), kilepes_negyzet(), felhasznalo_rubrika(): A különböző funkciók pygame.Rect(), illetve a felhasznalo_rubrika() függvény esetében Rubrika objektumát adja visszatérési értékként.
16.) szintvalaszto_negyzetek(kepernyo, statusz): Szintén egy statusz paraméterrel ellátott függvény, amely az adott szint kiválasztását jelöli: ha a statusz == 0, nem csinál semmit, ha a statusz == 1, a kezdő, ha a statusz == 2, a normal, illetve, ha a statusz == 3, akkor az extrém fokozatot jelölő rubrikát szinezi át. A statusz ellenőrzése után kirajzolja az objektumokat, és megjeleníti a képernyőn.
17.) szint_hibauzenet(kepernyo): A hibahely változóban felveszi a hibaüzenetet tartalmazó rubrikát, és kiírja a képernyőre.
18.) segitseg_megjelenes(obj, kepernyo, statusz): A statusztól függően jeleníti meg adott színben az obj-t, amely a segítséget jelölő rubrika körvonala.
19.) kozonsegszavazat(kepernyo, statusz, *forras): Abban az esetben, ha a forras paraméterben van tárolva adat, akkor ezt megfelelő formátumban írja ki a kozonseg_rubrika helyére. Ha a statusz == 0, akkor üres szöveggel jeleníti meg a négyzetet.
20.) jatek_vege(kepernyo, forras, statusz): a játék végének felületét tölti be, szintén statusz változó értéke alapján: ha a statusz értéke True, akkor megnyerte a játékot a felhasználó, így a rubrika szövege gratuláció, ha False, akkor inkább vigasztaló hangulatú. A szöveg tartalmát ezenkívül a forras adja, amely az elért pontszámot jelöli. A függvény egy teljesen új képernyőt tölt be a pygame.Surface((1024, 720)) függvénnyel. Erre az új képernyőre rajzolja ki a kilépés funkció rubrikáját, és a korábban említett szöveget.
21.) megallas(kepernyo): A megállási funkciót jelölő rubrikát rajzolja a képernyőre.
22.) megallas_negyzet(): visszatérési értéke a megállás gombjának helyzetét adó pygame.Rect() típusú érték.
23.) teljesites_ido_UI(ido): az NHF_dicsoseglista_v1.py modul kirajzolásában közreműködik: a játékmenetből mikroszekundumban érkező játékidőt átalakítja „perc:mp” formátumra, és ezt sztringként visszaadja a visszatérési értékben.
24.) dicsoseglista_UI(kepernyo, forras): grafikusan megjeleníti a toplistát: a 20 legjobb játékos eredményét írja ki a képernyőre. A forras paraméter egy tömb referenciája, amelyben Jatekos objektumok vannak. Elsőként a 4 soros, 20 oszlopos táblázatnak a címsorát veszi fel egy tömbben, amely az oszlopok neveit tartalmazza. Ezt követően a listaadatok változóba építi az

NHF_dicsoseglista_v1.py modultól kapott, Jatekos objektumokat tartalmazó tömböt. Ezt a tömböt a megfelelő oszlopokban rajzolja ki. Amennyiben segítség nélkül csinálta meg az adott kört a játékos, a nevét aranysárga színnel jelzi a képernyőn.

## A játékmenet – NHF_gameplay_v1.py

A moduk logikája a következő:

- importálja a sys, pygame, pygame.gfxdraw, NHF_UI_v1.py, NHF_adatszerk_v1.py, NHF_dicsoseglista_v1.py és random modulokat
- a játékmenet futásához és különböző felhasználói inputok miatt keletkező változások követéséért a Jatek objektum felelős
- A segítségek felhasználását és elérhetőségét a Segitoeszkoz modul tárolja
- A Valasz objektumra a felezés logikája miatt van szükség

A pygame-ben készült játékok futása gyakorlatilag egy végtelen ciklus, amely logikája az, hogy különböző felhasználói inputok hatására (pl. egér billentyű lenyomása, más billentyűk lenyomása) változik a felhasználói felület.
A játék futásáért Jatek objektum futas attribútuma felelős: amennyiben ez False értéket vesz fel, véget ér a játék.
A felhasználói inputok szempontjából egy kifejezett típusról beszélhetünk a program esetén, ez pedig az egér periféria „1”-es, azaz bal klikk gombja. Minden input az alapján változik, ahol a kurzor épp tartózkodik, és ez a gomb lenyomásra került.
Már szót ejtettem a működéshez kapcsolódó osztályokról, azonban a következőkben részletesebben bemutatom ezeket:

```
class Jatek:
    def __init__(self, futas, jovalasz, megallas):
        self.futas = futas
        self.jovalasz = jovalasz
        self.megallas = megallas
```

A játék akkor fut tovább (futas = True), és tölt be új kérdést a program, ha a jovalasz attribútum True, más esetben azt jelenti, hogy a felhasználó rosszul válaszolt. A megallas attribútum ha True értéket vesz fel, akkor a játék szintén véget ér, viszont az addigi megszerzett nyereményt kapja pontszámul a felhasználó.

Név: Doszpod Patrik
NK: IOS9WL

```
class Segitoeszkoz:
    def __init__(self, kozonseg, felezes, koz_felhasznal, fel_felhasznal):
        self.kozonseg = kozonseg
        self.felezes = felezes
        self.koz_felhasznal = koz_felhasznal
        self.fel_felhasznal = fel_felhasznal
```

A Segitoeszkoz attribútumai szintén boole típusúak, alapesetben a kozonseg és felezes True értékűek, a hozzájuk kapcsolódó felhasznalas-ok pedig False-ok. Ha egy segítség felhasználásra kerül, akkor a kozonseg/felezes False-ra vált, ezáltal többször nem lehet majd felhasználni (a másik két attribútum csak egyszer vesz fel True értéket, abban a körben amikor felhasználásra kerültek a segítségek).

```
class Valasz:
    def __init__(self, valasz, betujel):
        self.valasz = valasz
        self.betujel = betujel
```

A Valasz osztályban az adott kérdéshez tartozó válaszok, és azok betűjele kerül tárolásra. Ennek létrehozására a felezés logikája miatt volt szükség, abból a célból, hogy eltűnjenek a kiszűrt választási lehetőségek, és a felhasználó ezekre ne tudjon már kattintani.

A függvények bemutatása előtt fontos egy gyakran használt, beépített pygame függvény bemutatása: ez a kattintás vizsgálata a kurzor pozíciója alapján, azaz a „.collidepoint(pygame.mouse.get_pos())”. Egy adott pygame.Rect(), vagy Rubrika objektum negyzet(kepernyo) függvénye által meghatározott négyzet paramétereit hasonlítja össze az egér pillanatnyi pozíciójával. Ez a függvény rendre, kattintás hatására kerül meghívásra.

A játékmenet logikájának, és osztályainak ismeretében a következőkben bemutatom az ezen logikai kapcsolatokat létrehozó és működtető függvényeket.

*A játékmenet függvényeinek ismertetése:*

1.) kilepes_jelzo(): ha a felhasználó a kilépés négyzetére kattint True értéket ad vissza. Ellenkező esetben False-t.
2.) megallas_jelzo(): ha a felhasználó a megállás funkciót tartalmazó négyzetre kattint, True értéket ad vissza, ellenkező esetben False-t.
3.) megjelenites(kv_forras, p_forras, statusz): A UI függvényeiben bemutatott hatter(), kerdes_valasz_betoltes(), pontok_betoltes() és megallas() függvényeket hívja meg. A statusz felhasználói interakció hatására változik.
4.) valasz_hely(v): a válaszlehetőségek rubrikái közül visszaadja a helyes választ tartalmazó rubrika helyét (pygame.Rect() típusú értékként). A kiválasztás az

alapján történik, hogy a paraméterként megadott v-ben, milyen sztring van (pl. ha v="A", akkor a valasz_negyzet() függvényben kapott tömb 0.indexű tagját adja vissza, azaz az A válasz pygame.Rect()-jét).

5.) tippeles(lehetosegek, valaszlehetosegek, helyesvalasz): A paraméterei sorrendben a következők: egy Kerdes objektum, egy, a válaszlehetőségek négyzetét tároló tömb, és a helyes választ tartalmazó négyzet. Az egér pozícióját hasonlítja a jó válasz pozíciójával - ha egybeesnek akkor egy sztring („jóválasz" vagy „rosszválasz") a visszatérési érték, ellenkező esetben False – ekkor nem lett válaszlehetőségre kattintva.
A felezés használata befolyásolja a tartalmat: a honnan változóban felvett Valasz objektumok alapján kiszedi a válaszlehetőségek közül azokat, amelyek ki lettek szűrve a felezés hatására. Így a felhasználó csak valós lehetőségre kattinthat.

6.) segitseg_hasznalat(negyzet): Az egér pozícióját hasonlítja a segítség pozíciójával: ha belekattintok akkor True a visszatérési érték. Ellenkező esetben False.

7.) felezes(lehetosegek): A lehetosegek paraméter egy tömb, amely tartalmazza az adott kérdés Kerdes objektumát. Ebből a kellő információ a honnan tömbbe a korábban említett módon Valasz objektumokként kerül. Elsőként a helyes válasz kerül kiszűrésre, hogy az semmiképp ne kerüljön „törlésre". Ezt követően a random modul alkalmazásával véletlenszerűen kiválaszt 2 különböző választ és törli azok szövegét.

8.) kozonseg_szavazas(lehetosegek): A lehetosegek paraméterből szükséges információ itt is a honnan tömbbe kerül tárolásra. A UI-nál bemutatott kozonsegszavazat() függvényt hívja meg, 1-es statusz-szal, a szövege pedig a kalkulált százalékok. A jó válasz százaléka mindig 33-70% közötti érték, amely az esetek többségében a jó választ jelöli. A maradék a többi válasz között oszlik el.

9.) gameplay(szint): A játékmenetet működtető függvény. Paramétere a játék indításakor kiválasztott szint (amely egy Kerdes objektumokat tároló tömb), és ez kerül a kerdes változóba. A kérdéseket tehát innen „tölti be" a játék.
Miután az alapadatok, és a futáshoz szükséges paraméterek beállításra kerültek (azaz a korábban bemutatott objektumok létrehozása 1-1 változóban), a játék futása elindul. A játék 2 egymásba ágyazott ciklus: külső ciklus = 15-ször fut le, mert 15 kérdésből áll egy kör. Belső ciklus = ez maga a játék futása, végtelen ciklus megfelelő kattintásig. A játékon belüli időmérés úgy zajlik, hogy a játék futása előtt mindig a 6000 integer kerül tárolásra az mp változóban. A játék futása alatt minden képernyőfrissítés során 1-gyel csökken ez a változó. Ha ezt „integerosztással" osztjuk 100-zal, akkor egy adott képernyő betöltése során nagyjából 1 másodperc eltelte történik, így a képernyőn ez alapján 1-gyel csökken a kijelzett másodperc értéke. Ha mp == 0, akkor a játék véget ér. A kattintási interakciók a pygame „event" függvényei segítségével történnek. Ilyen, a programban leggyakrabban használt ciklus:

```
for event in pygame.event.get():
        if event.type == pygame.MOUSEBUTTONDOWN:
            if event.button == 1: #...
```

Azaz minden „esemény”-re a pygame futása alatt, ha az „1”-es egérbillentyű lenyomásra kerül, csinál valamit.
A már korábban említett objektumok logikai értéke változik ekkor:

- ha a tippeles() „jóválasz”-t ad vissza, a Jatek objektum jovalasza = True lesz.
- ha „rosszválasz”-t ad, akkor ugyanez False lesz.
- a segítségek felhasználására és megállásra vonatkozóan már kitárgyaltam a funkciók működését.

A kattintások hatására a futas attribútum mindig False-ra vált, erre azért van szükség, hogy a színek megfelelően betöltésre kerüljenek jó, és rossz válasz esetén. Mielőtt véget ér a gameplay() futása meghívja a vege(kerdes_szama, megallt_e) függvényt. Visszatérési értéke a dicsolista_adatok(ido_halad, segitseg.kozonseg, segitseg.felezes, vegpontszam) visszatérési értéke.

10.) vege(kerdes_szama, megallt_e): paraméterei a kérdés azon száma, ahol véget ért a játék, illetve a megállást jelző logikai érték. Ha megállt a felhasználó, és így lett vége a játéknak, akkor a megallt_e az 1-es statuszt állítja be, ellenkező esetben 0-t.
A nyeremeny változóba elteszi a dicsoseglista modulban számolt pontszámot, majd egy végtelen ciklusban, amely kilépésig (pygame.quit(), vagy kilepes_jelzo() függvények) tart. Itt a UI jatek_vege() függvény kerül meghívásra.

11.) dicsolista_adatok(ido, kozonseg, felezes, pont): a felhasználó elért pontjait, használt-e segítséget, és a játékidőt tárolja egy tömbben. Ez a tömb lesz a visszatérési értéke is.

Név: Doszpod Patrik
NK: IOS9WL

# A dicsőséglista – NHF_dicsoseglista.py

A modul logikája a következő:

- importálja a sys, pygame, pygame.gfxdraw, NHF_UI_v1, NHF_adatszerk_v1 modulokat
- létrehozza a Jatekos objektumot, melyben tárolja a felhasználó adatait
- UI függvényhívással kiírja a top 20 ranglistát.

A modul a Jatekos objektumra alapozódik. A főprogram modulja helyezi egy ilyen Jatekos objektumba a játékos adatait, majd ez kerül fájlba írásra a dicsőséglistában.

```
class Jatekos:
    def __init__(self, nev, pontszam, nehezseg, ido, segitseg):
        self.nev = nev
        self.pontszam = pontszam
        self.nehezseg = nehezseg
        self.ido = ido
        self.segitseg = segitseg
```

A nev a játékos neve, a pontszam az elért pontja, a nehezseg a választott nehézségi szint, az idő a teljesítés ideje és a segitseg egy logikai True/False, hogy használt-e segítséget.

*A dicsőséglista függvényeinek ismertetése:*

1.) nyeremeny_szamitas(stadium, statusz): a stadium paraméterben azt a számot kapja meg, ahol épp tartott a játék vége során a felhasználó (azaz a pontlistában a pontszám indexét). A statusz azt jelzi, hogy megállást követően történt-e a függvényhívás: amennyiben ez 1-es integer, akkor a visszatérési érték az a tömb, amely tartalmazza a pontszámot sztringként, és integerként is. Ha 0, a játékszabályoknak megfelelően kisebb, garantált nyereményt ugyanilyen elven tároló tömb.

2.) listaba_felvesz(adatok): az adatok a játékos adatai, egy Jatekos objektumban. Ezt a függvényt a játék végén hívja meg a főprogram. Elsőként átalakítja az adatokat sztringgé, majd „at" módban hozzáadja az adatokat egy sorban a „dicsoseglista.txt"-hez.

3.) atalakit(adat): A paraméterként kapott adatot átalakítja Jatekos objektummá.

4.) lista_rendezes(): Elsőként „rt" módban beolvassa a „dicsoseglista.txt"-t, majd a következő sorrendben rendezi: 1.) szint szerint, 2.) pontszám szerint, 3.) felhasznált segítség szerint. Ezeket a megfelelő függvények meghívásával teszi (rendezes_szint_szerint(), rendezes_pontszam_szerint, rendezes_segitseg_szerint()). Miután elkészült a rendezésekkel, és ezt egy tömbben tárolta, beírja a rendezett adatokat a „dicsoseglista.txt" fájlba „w+" módban.

5.) rendezes_szint_szerint(osszes, szint): szintek alapján rendezi a játékosok elért eredményeit. Ha a paraméterben megadott szint változóval egyenlő a az adott indexű Jatekos.nehezseg, akkor egy kezdetben üres tömbbe teszi. Ez a tömb lesz a visszatérési érték.
6.) rendezes_pontszam_szerint(lista): pontszám alapján rendezi a paraméterként kapott tömböt: a rendezés kiválasztásos rendezéses algoritmussal történik. Amennyiben pontegyenlőség van, a játékidőt veszi számításba.
7.) rendezes_segitseg_szerint(lista): felhasznált segítségek alapján rendezi a tömböt: ha valaki max. pontot ért el, és nem használt segítséget, akkor adott szinten a lista élére ugrik.
8.) lista_beolvas(): „rt" módú fájl beolvasás, melyben a fájl a „dicsoseglista.txt". Egy kezdetben üres tömbbe teszi a beolvasott, átalakított sorokat, és ezt adja vissza értékként.
9.) dicsoseglista_kiiras(kepernyo): A dicsőséglista grafikus megjelenítése. Elsőként rendezi a dicsőséglistát a lista_rendez() függvényhívással, majd az UI dicsoseglista_UI függvényhívással megmutatja a grafikus felületet a képernyőn. A pygame.quit() esemény hatására bezáródik.

A lista_rendezes() függvényt minden beolvasáskor meghívja a modul, hogy mindig frissüljön a dicsőséglista.

## A funkciók vezérlése – NHF_main_menu.py

A modul logikája a következő:
- importálja a pygame, pygame.gfxdraw, sys és NHF_UI_v1 modulokat
- létrehozza a Menu objektumot, amelyben logikai változókat tárol
- kattintások hatására választ funkciót a felhasználó

```
class Menu:
    def __init__(self):
        self.inditas = None
        self.dicsoseg = None
        self.kilepes = None
```

A Menu objektumban tárolt inditas, dicsoseg és kilepes abban az esetben vesz fel True értéket, ha az adott funkció négyzetére kattint a felhasználó. Amennyiben így történt a futás megáll, és visszatérési értékként a megfelelő sztringet adja.

Név: Doszpod Patrik
NK: IOS9WL

*A menüt vezérlő függvények ismertetése:*

1.) def kattintas(funkcio): adott funkcióra történő kattintás alapján ad vissza True, vagy False értéket.
2.) main_menu(): egy addig futó végtelen ciklus, amíg nem lett választva funkció („1”-es egérgomb lenyomás, és kattintas True értéke alapján). Amint ez megtörtént, visszatérési értékként a funkciónak megfelelő sztringet ad vissza.

## A főprogram – NHF_4_foprogram.py

A modul logikája a következő:

- a korábban felsorolt modulok mindegyikének meghívása
- pygame.init() inicializáció, majd képernyő létrehozása – pygame.display.set_mode((1024, 720))
- Szint objektum létrehozása, amely a játék indításakor elérhető 3 szint kiválasztásának státuszát tárolja
- egy végtelen ciklus, mely a menü, majd az ott elérni kívánt funkciót tölti be.

A Szint objektum felépítésére azért van szükség, hogy a felhasználó kizárólag 1 szintet tudjon választani, és az alapján történjen a kérdések betöltése. Erre a statusz szolgál, mely ez alapján vesz fel integer értéket. A valasztott arra szolgál, hogy a játék csak akkor legyen indítható, ha választásra került szint is.

```
class Szint:
    def __init__(self, statusz, valasztott):
        self.statusz = statusz
        self.valasztott = valasztott
```

*A főprogram függvényeinek ismertetése:*

1.) szint_valasztas(): Adott szint kiválasztása kattintásra. Azt az integert adja vissza, amely szintre kattintott a felhasználó (pl. ha kezdő szint, akkor viszatérési érték = 1).
2.) kilepes(): A kilépés gombra kattintva, amely a UI kilepes_negyzet() függvényének a helye – True értéket ad vissza.
3.) inditas(): működése azonos a kilepes()-sel, de ebben az esetben a gombot az inditogomb() függvény helyzete adja.
4.) main(): A főprogram futása. Ez kerül meghívásra, ha megnyitjuk a modult. Elsőként a tovabblep változóba teszi a main_menu() függvény meghívásának visszatérési értékét – azaz először meghívja a main_menu()-t, majd annak a visszatérési értékeként kapott sztringjének referenciáját tárolja a változóban. Ez fogja indításra, a dicsőséglista megnyitására, vagy kilépésre serkenteni a játékot. Ha „indítás” történt, akkor elsőként egy „fo_futas, amíg True” ciklusba lép, ami addig tart, amíg a felhasználó: 1.) beírja/nem írja be a nevét 2.) választ

szintet 3.) rákattint az indítás gombra. Amennyiben úgy akar indítani a játékos, hogy nem választott szintet, ezt kiírja a képernyőre a program, jelezve, hogy enélkül nem tud továbbhaladni. Ha ezen túllép, a program meghívja a játékmenet gameplay() függvényét. Amikor ez véget ért, a visszatérési értékében adott adatokat tárolja egy felhasznaloi_adatok nevű tömbben, amit a dicsőséglista listaba_felvesz() függvényével ír a dicsőséglistára. Végezetül a lista_rendezes() meghívásával rendezi a listát. Ha a felhasználó a dicsőséglistát akarta megnyitni, akkor a dicsoseglista_kiiras() függvény kerül meghívásra.

Végezetül a program a pygame.quit() és sys.exit() függvények meghívásával bezárul.

Név: Doszpod Patrik
NK: IOS9WL

# Felhasznált források

A játékban használt logok linkjei:

"mil_logo.gif":

https://cdn.rtl.hu/9f/ce/legyen-on-is-milliomos_image_848cf66cb55d90898e9883a27d4e_16-9?size=2"

"felezes.png":
https://static.wikia.nocookie.net/millionaire/images/7/73/Classic5050.png/revision/latest?cb=20160407180209

"kozonseg.png":
https://static.wikia.nocookie.net/millionaire/images/c/c6/ClassicATA.png/revision/latest?cb=20160407180412

*A program írása előtt a pygame modul használatát nem ismertem, így a működés jobb megismerésére a következő linkeket használtam:*

- https://infopy.eet.bme.hu/pygame/
- https://www.youtube.com/watch?v=FfWpgLFMI7w