Ágazati alapvizsga

Programozás Pythonban

vizsgarészéhez gyakorló feladatsorok

Informatika és távközlés ágazat

2020. december 15.

Szerző: Varga Péter (peter.varga@thesmart.academy)

Készítette A Hálózati Tudás Terjesztéséért Programiroda Alapítvány (**HTTP Alapítvány**)
megbízásából a The Smart Solutions Kft. (**theSMART.academy**)

Tipp a feladatlap összeállításához

A feladatlapon különítsünk el bevezető szövegnek helyet. A bevezető szöveg javasolt tartalma:

- *Az Ön feladata az alábbiakban olvasható leírás alapján három program elkészítése.*
- *A három Python-feladat elvégzésére összesen kb. 60 perc áll rendelkezésre.*
- *A programokat a (megadott hely)-re kell mentenie.*
- *A programok elkészítése során a felhasználó által megadott adatok helyességét nem kell ellenőriznie – ha például a program egy 1 és 5 közé eső szám megadását kéri a felhasználótól, akkor feltételezheti, hogy a felhasználó számot, és a megadott feltételeknek megfelelő számot ad meg.*
- *Törekedjen arra, hogy a tanult programozási elveknek megfelelő adatszerkezeteket, vezérlési szerkezeteket alkalmazzon!*
- *Munkáját rendszeresen mentse! Amennyiben a vizsga során a számítógép nem megfelelő működését tapasztalja, jelezze a felügyelő tanárnak!*

A feladatokat adjuk úgy ki, hogy egy-egy feladat egy oldalra kerüljön – a feladat megoldása közben ne kelljen lapozni.

1 Első feladatok

A feladatban célszerűen néhány szám vagy szöveges adat bekérése és valamilyen szempont szerinti összevetése, átalakítása történik. A feladat szövegének lényegi része egy-két mondat. A vizsgázónak elég például egy bonyolultabb elágazást, vagy egy ciklust és egy vele kapcsolatban álló elágazást megvalósítani. Ha ciklust készíttetünk, az ismétlődések száma legyen kellően alacsony, hogy a tesztelést ne nyújtsa el nagyon.

Törekedjünk arra, hogy a kimenetben ne kelljen változót mondat közepén elhelyezni: Ha van rá mód, akkor „Az ajtóba a *piros* kulcs illik.” kimenetet fogalmazzuk át „Az ajtóba illő kulcs színe: *piros*”-ra.

Érdemes ezeknél az egyszerű feladatoknál is a program futását bemutató ábrát megadni. Adjuk meg a megírandó program nevét (a többi feladatnál is)!

1.1 Nagyobb szám

Feladat szövege

Írjon programot **nagyobb.py** néven! A program kérjen be két számot a felhasználótól, majd írja ki, hogy melyik a nagyobb! A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

```
C:\Users\raerek\programok>nagyobb.py
Adj meg egy számot! 1
Adj meg egy másik számot! 17
A nagyobb érték 17
C:\Users\raerek\programok>nagyobb.py
Adj meg egy számot! 28
Adj meg egy másik számot! -2
A nagyobb érték: 28
C:\Users\raerek\programok>nagyobb.py
Adj meg egy számot! 999
Adj meg egy másik számot! 999
A két szám egyenlő
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot nagyobb.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználótól az egyik számot, és tárolja.
3. A bekért számot szám típusúvá alakítja.
4. Az előző két lépést a második számmal is elvégzi.
5. Elágazást használ a különböző esetek kezelésére.
6. Ha a két szám nem egyenlő, helyesen állapítja meg és írja ki, hogy melyik a nagyobb.
7. Helyesen állapítja meg, és írja ki, ha a két szám egyenlő.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

Megjegyzés

Variáció: osztható-e hárommal, illetve öttel a bekért szám?

1.2 Szavak

Feladat szövege

Írjon programot **szavak.py** néven! A program kérjen be két szót a felhasználótól, majd írja ki, hogy melyik a hosszabb! A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

C:\Users\raerek\programok>***szavak.py***
Adj meg egy szót! ***kelkáposzta***
Adj meg egy másik szót! ***kisegér***
A hosszabb szó: kelkáposzta
C:\Users\raerek\programok>***szavak.py***
Adj meg egy szót! ***kelkáposzta***
Adj meg egy másik szót! ***káposztafej***
A két szó egyforma hosszú.
C:\Users\raerek\programok>***szavak.py***
Adj meg egy szót! ***árvíztűrő***
Adj meg egy másik szót! ***tükörfúrógép***
A hosszabb szó: tükörfúrógép

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehozott programot szavak.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználótól az egyik szót, és tárolja.
3. Meghatározza az egyik szó hosszát.
4. Az előző két lépést a második szóval is elvégzi.
5. Elágazást használ a különböző esetek kezelésére.
6. Ha a két szó hossza eltér, helyesen állapítja meg és írja ki, hogy melyik a hosszabb.
7. Helyesen állapítja meg, és írja ki, ha a két szó egyforma hosszú.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

1.3 Nyitvatartás

Feladat szövege

Egy bolt pontban reggel nyolc órakor nyit, és pontban délután tizenhat órakor zár be – azaz 8:00-kor már nyitva van és 16:00-kor már nincs nyitva.

Írjon programot **nyitvatartas.py** néven! A program kérjen be egy egész órát jelző számot a felhasználótól, majd írja ki, hogy a megadott időpontban nyitva van-e a bolt! Amennyiben igen, akkor azt is írja ki, hogy mennyi idő van még zárásig, azaz hány egész óra áll rendelkezésre odaérni a boltba! A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

```
C:\Users\raerek\programok>nyitvatartas.py
Hány óra van most? 8
A bolt nyitva van.
Ennyi órád van még odaérni: 8
C:\Users\raerek\programok>nyitvatartas.py
Hány óra van most? 7
A bolt zárva van.
C:\Users\raerek\programok>nyitvatartas.py
Hány óra van most? 17
A bolt zárva van.
C:\Users\raerek\programok>nyitvatartas.py
Hány óra van most? 16
A bolt zárva van.
C:\Users\raerek\programok>nyitvatartas.py
Hány óra van most? 12
A bolt nyitva van.
Ennyi órád van még odaérni: 4
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot nyitvatartas.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználótól egy számot, és tárolja.
3. A bekért számot szám típusúvá alakítja.
4. Elágazást használ a különböző esetek kezelésére.
5. Helyesen állapítja meg és írja ki, ha a bolt nyitva van.
6. Helyesen állapítja meg és írja ki, hogy hány óra van még odaérni a boltba.
7. Helyesen állapítja meg és írja ki, ha a bolt zárva van.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

1.4 Visszaszámlálás

Feladat szövege

Egy rakéta indítása előtt több órával visszaszámlálást kezdenek és óránként egyet számolnak vissza a rakéta indításáig. A felhasználó határozza meg, hogy hány órás a visszaszámlálás. A visszaszámlálást minden órában egy órára felfüggeszthetik, ha valamilyen váratlan esemény – műszaki hiba, időjárási probléma – merül fel. Amikor a visszaszámlálás eléri a 0-t, a rakétát fellövik.

Írjon programot **raketa.py** néven, amely a visszaszámlálás számait jeleníti meg a képernyőn! Természetesen nem kell a visszaszámlálás lépései között eltelni időnek – minden üzenet megjelenését azonnal követheti a következő. A visszaszámlálás minden lépésénél kérdezze meg a felhasználót, hogy az adott órában szükség volt-e a visszaszámlálás fölfüggesztésére! A visszaszámlálás megjelenítését követően a program írja ki, hogy a visszaszámlálás kezdetétől hány óra telt el – a visszaszámlálás eredetileg tervezett hosszát a felfüggesztésekkel megnövelve!

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

C:\Users\raerek\programok>***raketa.py***
Hány órás visszaszámlálást tervezünk? ***5***
Visszaszámlálás: 5
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
Visszaszámlálás: 4
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
Visszaszámlálás: 3
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
Visszaszámlálás: 2
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
Visszaszámlálás: 1
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
A rakéta a visszaszámlálást követően ennyi órával indult: 5
C:\Users\raerek\programok>***raketa.py***
Hány órás visszaszámlálást tervezünk? ***4***
Visszaszámlálás: 4
Fel kell függeszteni a visszaszámlálást (i/n)? ***i***
Visszaszámlálás: 3
Fel kell függeszteni a visszaszámlálást (i/n)? ***i***
Visszaszámlálás: 2
Fel kell függeszteni a visszaszámlálást (i/n)? ***i***
Visszaszámlálás: 1
Fel kell függeszteni a visszaszámlálást (i/n)? ***n***
A rakéta a visszaszámlálást követően ennyi órával indult: 7

Egy lehetséges megoldás kódja

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot raketa.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználótól egy számot, és tárolja.
3. A bekért számot szám típusúvá alakítja.
4. Ciklust szervez a visszaszámlálás megjelenítésére
5. A ciklusmagban megkérdezi a felhasználót, hogy fel kellett-e függeszteni a visszaszámlálást.
6. Megszámolja a felfüggesztéseket.

7. Meghatározza és kiírja, hogy a rakéta a visszaszámlálást kezdetét követően hány órával indult útjára.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

1.5 Bejelentkezés

Feladat szövege

Írjon programot **jelszo.py** néven, amely azt vizsgálja, hogy egy felhasználó helyesen adja-e meg a jelszavát! A program addig kérdezi újra a felhasználónév-jelszó párost, amíg a felhasználó mindkettőt hibátlanul meg nem adja. A program egyetlen felhasználó (bori99) jelszavát (Szivecske<3) ismeri, csak ezt a párost fogadja el helyesként. Mind a sikertelen, mind a sikeres bejelentkezési kísérletekről üzenetet ír a képernyőre.

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

C:\Users\raerek\programok>***jelszo.py***
Adja meg a felhasználónevét! ***bori99***
Adja meg a jelszavát! ***Szivecske<3***
Belépés engedélyezve.
C:\Users\raerek\programok>***jelszo.py***
Adja meg a felhasználónevét! ***Bagaméri***
Adja meg a jelszavát! ***A kankalin sötétben virágzik!***
Belépés megtagadva.
Adja meg a felhasználónevét! ***bori99***
Adja meg a jelszavát! ***hibásjelszó***
Belépés megtagadva.
Adja meg a felhasználónevét! ***hibásfelhasználó***
Adja meg a jelszavát! ***Szivecske<3***
Belépés megtagadva.
Adja meg a felhasználónevét! ***bori99***
Adja meg a jelszavát! ***Szivecske<3***
Belépés engedélyezve.

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot jelszo.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználónevet és tárolja.
3. Bekéri a jelszót és tárolja.
4. Helyesen állapítja meg és írja ki, amikor be kell engedni a felhasználót.
5. Helyesen állapítja meg és írja ki, amikor meg kell tagadni a belépést.
6. Helyesen működő ciklust szervez az ismétlődő tevékenység elvégzésére.
7. Kilép a ciklusból, ha a felhasználónév és a jelszó is helyes volt.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

2 Második feladatok

A második feladatok picit bonyolultabbak az elsőknél – ezért a feladatszövegek elején röviden összefoglaljuk, hogy mit csinál a program, és ezt követik a részletesebb utasítások.

A feladatban érdemes helyet adni egyszerű függvény készítésének vagy részben megírt függvény kiegészítésének – esetleg több kész függvény megfelelő használatának. A függvényhasználat életszerűvé tétele miatt szinte bizonyosan ciklus is kerül a feladatba – azaz nem probléma, ha az első feladatban nem kértük számon ezt az ismeretanyagot.

A második feladatokban érdemes lehet egyszerűbb listaműveleteknek helyet adni, vagy listákban tárolt adatokat feldolgozó ciklusokat használtatni.

Ha kész programrészre támaszkodunk, akkor azt fájlként vagy fájlként is kapja meg a vizsgázó.

A feladatszöveget érdemes több bekezdésre tagolni – esetleg betűzni a részfeladatokat.

Mindenképp érdemes a program futását bemutató ábrát elhelyezni a feladatban.

2.1 Ágazati alapvizsga

Feladat szövege

A program vizsgázók nevét és pontszámát kéri be. Eldönti és kiírja, hogy a vizsgázó sikeresen vizsgázott-e. A vizsga sikeres, ha legalább 48 pontot ért el a vizsgázó.

Írjon programot **vizsga.py** néven!

Kérje be a vizsgázók nevét és az elért pontszámokat! Írja meg azt a függvényt, ami eldönti, hogy a vizsga sikeres-e! A függvény paramétere a vizsgázó által elért pontszám, a visszatérési értéke logikai érték: igaz, ha a vizsga sikeres, hamis, ha sikertelen. Ezt a függvényt használja fel a programjában!

A program kérdezgesse addig újabb és újabb vizsgázó nevét és pontszámát, amíg a vizsgázó nevének megadásakor üres bemenetet nem kap! Ilyen akkor történik, ha a felhasználó egyszerűen Entert nyom, anélkül hogy bármit is begépelne.

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

```
C:\Users\raerek\programok>vizsga.py
Add meg a vizsgázó nevét! Linus Torvalds
Add meg a pontszámát! 121
Linus Torvalds vizsgája sikeres.
Add meg a vizsgázó nevét! Dennis Ritchie
Add meg a pontszámát! 119
Dennis Ritchie vizsgája sikeres.
Add meg a vizsgázó nevét! Steve Ballmer
Add meg a pontszámát! 27
Steve Ballmer vizsgája sikertelen.
Add meg a vizsgázó nevét!

C:\Users\raerek\programok>
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot vizsga.py néven, a program hibaüzenet nélkül lefut.
2. Bekér egy nevet és tárolja.
3. Bekér egy pontszámot.
4. Egy bekért számot szám típusúvá alakít.
5. Egy pontszám alapján helyesen állapítja meg, hogy a vizsga sikeres-e vagy sem.
6. Egy esetben helyesen jelenít meg üzenetet vizsga eredményességéről. Az üzenet a vizsgázó nevét is feltünteti.
7. Ciklust szervez a nevek és a pontszámok bekérésére, illetve a vizsga sikerességének kiírására.
8. A ciklus futása véget ér, ha a név megadásakor üres bemenetet kap a program.
9. Függvényt hozott létre a vizsga sikerességének eldöntésére.
10. A függvény paramétere a vizsga pontszáma.
11. A függvényt helyesen hívja.
12. A függvény visszatérési értéke alapján a főprogram (vagy az annak megfelelő függvény) írja ki a vizsga eredményességét.
13. A függvényhívás a ciklusmagba kerül.
14. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

2.2 Kedvenc filmjeink

A feladatban elkészítendő program bekéri három film címét, illetve percben kifejezett hosszát. Egy-egy filmcím-filmhossz adatpár megadását követően a program a percben kifejezett időtartamot átszámolja órákra és pecekre – például a 61 percet 1 óra 1 percre. Az eredményt a film címével együtt kiírja.

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

A program kiindulási alapja a filmalap.py fájlban található. Ennek felhasználásával írjon programot **kedvencfilm.py** néven! Egészítse ki a megkapott függvényt úgy, hogy az alkalmas legyen percben megadott időtartamot órában és percben visszaadni! A program többi részében használja az így kiegészített függvényt!

```
C:\Users\raerek\programamok>kedvencfilm.py
Add meg egy film címét! Indul a bakterház
Hány perces a film? 66
A(z) Indul a bakterház című film 1 óra 6 perc hosszú.
Add meg egy film címét! Ben-Hur
Hány perces a film? 224
A(z) Ben-Hur című film 3 óra 44 perc hosszú.
Add meg egy film címét! Bérlők
Hány perces a film? 1
A(z) Bérlők című film 0 óra 1 perc hosszú.

C:\Users\raerek\programamok>
```

A filmalap.py tartalma

```
def óraperc(): # Egésztse ki a függvénydefiníciót paraméterrel!
    # Írja meg a függvény többi részét!
    return str(óra) + ' óra ' + str(perc) + ' perc'
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot kedvencfilm.py néven, a program hibaüzenet nélkül lefut.
2. Bekér egy címet.
3. Bekéri egy film hosszát percben.
4. Egy bekért számot szám típusúvá alakít.
5. Egy percadatról helyesen állapítja meg, hogy hány órát jelent.
6. Egy percadatról helyesen állapítja meg, hogy az egész órákon kívül hány percet jelent.
7. Egy esetben helyesen jelenít meg üzenetet a film hosszáról. Az üzenet a film hosszát is tartalmazza.
8. Ciklust szervez a címek és az időtartamok bekérésére, illetve az üzenetek kiírására.
9. Függvényt használ a film hosszának eldöntésére.
10. A függvény egy paramétere a film percben kifejezett hossza.
11. A függvényt helyesen hívja.
12. A függvény visszatérési értéke alapján a főprogram (vagy az annak megfelelő függvény) írja ki a film hosszát órában és percben.
13. A függvényhívás a ciklusmagba kerül.
14. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

2.3 Mondatok

Feladat szövege

Az elkészítendő program főneveket kér be a felhasználótól – összesen hármat – és a főnév felhasználásával egyszerű mondatokat alkot. A mondatok három szóból állnak: Az „a” vagy az „az” névelőből, a főnévből és egy véletlenszerűen választott jelzőből. A program működését az alábbi minta szemlélteti:

a) Írjon programot „**mondatok.py**” néven! Rendelkezésére áll a mondatok_alap.py fájl, benne egy félig megírt függvénnyel névelő() néven. E függvény feladata a főnévhez illeszkedő névelő meghatározása. Egészítse ki úgy a függvényt úgy, hogy „a” névelőt adjon vissza, ha a szó magánhangzóval kezdődik, és „az” névelőt minden más esetben! Használja a függvényt a feladat további részének elkészítése során!

b) Kérjen be három főnevet a felhasználótól! Határoztassa meg a főnévhez illeszkedő névelőt a kiegészített névelő() függvénnyel!

c) A mondatok_alap.py fájlban található, előre elkészített jelző() nevű függvény a benne tárolt három jelző közül ad egyet véletlenszerűen vissza. Írja ki a soron következő főnév névelőjét, magát a főnevet és egy jelzőt!

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

```
C:\Users\raerek\programok>mondatok.py
Adj meg három főnevet!
1. főnév: egér
Az egér könnyed.
2. főnév: bányarém
A bányarém piros.
3. főnév: repülőgéphordozó
A repülőgéphordozó könnyed.
```

A mondatok_alap.py tartalma

```
def névelő():
    magánhangzók = 'aáeéiíoóöőuúüű'
    if szó[0].lower() in magánhangzók:
    # Egészítse ki a függvényt a visszatérést végző résszel!

def jelző():
    return random.choice(['piros', 'nagy', 'könnyed'])
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot mondatok.py néven, a program hibaüzenet nélkül lefut.
2. Helyesen egészíti ki a névelő() függvényt a lehetséges visszatérési értékekkel.
3. Helyesen egészíti ki a névelő függvényt a paraméterrel.
4. Bekér egy főnevet.
5. Helyesen hívja a névelő() függvényt.
6. A névelő() függvény hívásával helyesen írja ki a névelőt egy mondat elejére, akár kisbetűvel kezdve a mondatot.
7. A névelő() függvény visszatérési értékét felhasználva nagy kezdőbetűvel kezd egy mondatot.
8. Helyesen hívja a jelző() függvényt.
9. A jelző() függvény hívásával helyesen írja ki a jelzőt egy mondat végére.
10. Egy mondat végére pont kerül.
11. Ciklust szervez a három főnév bekérésére és a mondatok kiírására.

12. A ciklusmag üzeneteiben változó értékét felhasználva megjeleníti, hogy hányadik főnevet kéri be.
13. Mindhárom mondatot helyesen írja ki.
14. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

3 Harmadik feladatok

A második feladatokhoz hasonlóan itt is először röviden ismertetjük a program működését, mielőtt a részletes utasításokra térnénk. Szerepeljen a program futását bemutató ábra a feladatszövegben.

Ebben a feladatban érdemes foglalkoznunk az osztályok és/vagy a modulok használatával. A kevés rendelkezésre álló idő és pont is azt kívánja, hogy legyünk óvatosak teljes osztály vagy modul elkészíttetésével – célszerűbb lehet egy-egy félig elkészített kiegészíttetése.

Ez az a feladat, ahol érdemes elhelyezni a fájlkezelést számonkérő részfeladatokat. A közölt példafeladatokban ez a rész kevésbé hangsúlyos, de elképzelhető olyan feladat is, ahol a fájlkezelés kap nagyobb hangsúlyt az osztályok – modulok ismeretével szemben. Ilyenkor az sem baj, ha csak használni kell osztályokat, vagy modulokat – akár úgy, hogy a működésüket a megírt kód alapján kell felismernie a vizsgázónak.

A feladat hossza miatt érdemes betűzni a részfeladatokat.

A diákjaink ismeretében esetleg nem baj, ha a feladatszöveg egy-egy markánsabb része redundánssá válik.

3.1 Híres nők

Feladatszöveg

Az elkészítendő program bekéri három híres nő nevét, foglalkozását, illetve nemzetiségét, amely angol vagy német lehet. Ezt a három adatot minden esetben egy-egy objektumban tárolja. Az adatok megadását követően a program a mintának megfelelően, a nemzetiségtől függően Ms. (angolok) vagy Frau (németek) előtaggal együtt kiírja az objektumokban tárolt neveket és foglalkozásokat.

a) Írjon programot **hiresek.py** néven!

b) Az adatok tárolására használt objektumok alapját képező osztályt a hiresek_alap.py fájl tartalmazza részben elkészítve. Egészítse ki az osztálydefiníciót úgy, hogy az objektumok alkalmasak legyenek a nemzetiség tárolására is!

c) Bővítse az osztályt egy olyan tagfüggvénnyel, amely a tárolt nemzetiségtől függően „Ms." vagy „Frau" értékkel tér vissza!

d) Kérje be a felhasználótól az adatokat és tárolja őket! Az adatbekérést követően írja ki a megadott emberek nevének előtagját, a nevet és a foglalkozást!

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

```
C:\Users\raerek\programok>hiresek.py
Add meg egy híres nő nevét! Katarina Witt
Add meg a foglalkozását! műkorcsolyázó
Add meg a nemzetiségét (a/n)! n
Add meg egy híres nő nevét! Ada Lovelace
Add meg a foglalkozását! informatikus
Add meg a nemzetiségét (a/n)! a
Add meg egy híres nő nevét! Diana Frances Spencer
Add meg a foglalkozását! hercegnő
Add meg a nemzetiségét (a/n)! a
Frau Katarina Witt egy híres műkorcsolyázó
Ms. Ada Lovelace egy híres informatikus
Ms. Diana Frances Spencer egy híres hercegnő
```

A hiresek_alap.py tartalma

```
class HíresNő:
  def __init__(self, név, foglalkozás):
    self.név = név
    self.foglalkozás = foglalkozás
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot hiresek.py néven, a program hibaüzenet nélkül lefut.
2. Adatszerkezetet hoz létre a három HíresNő osztályú objektum tárolására.
3. Bekéri egy nő nevét.
4. Bekéri egy nő foglalkozását.
5. A név és a foglalkozás felhasználásával HíresNő osztályú objektumot hoz létre.
6. Egy HíresNő osztályú objektumot elhelyez a létrehozott adatszerkezetben.
7. Három bekért név és foglalkozás alapján három objektumot hoz létre.
8. Mindhárom objektumot elhelyezi az adatszerkezetben.
9. Az adatszerkezetben tárolt objektumok alapján megjelenít egy objektumot a NÉV egy híres FOGLALKOZÁS formában.
10. Mindhárom objektumot megjeleníti.

11. Az osztály módosításával alkalmassá teszi az objektumokat a nemzetiség tárolásra.
12. A módosítást úgy végzi el, hogy a nemzetiséget az objektum létrejöttekor kelljen megadni.
13. Az osztályban előtag() néven tagfüggvényt hoz létre.
14. Az előtag() tagfüggvény a tárolt nemzetiség alapján az angoloknál a Ms., németeknél a Frau értéket adja vissza.
15. A program adatbekérő része a nemzetiséget is megkérdezi mindhárom esetben.
16. A létrehozott objektumok tárolják a nemzetiséget.
17. Az objektumok megjelenítésekor az előtag() tagfüggvény kimenetét helyesen használja, azaz az angol nemzetiségűek Ms. NÉV egy híres FOGLALKOZÁS, a németek Frau NÉV egy híres FOGLALKOZÁS formában íródnak ki.
18. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).

3.2 Állatok

Feladatszöveg

Az elkészítendő program állatfajok nevét és tömegét tárolja objektumokban. A felhasználótól bekéri három állatfaj nevét és tömegét, majd ezt követően meghatározza és fájlba kiírja a a legnehezebb állatfaj nevét.

a) Írjon programot **nehez.py** néven!

b) Az állatfajok adatainak tárolására szolgáló objektumok alapját képező osztály rendelkezésre áll az allat.py fájlban. A programjában töltse be ezt a modult, és használja a bene lévő osztályt!

c) Kérje be a felhasználótól három állatfaj nevét és tömegét! Az adatok alapján hozzon létre Állatfaj osztályú objektumokat és tárolja őket!

d) Határozza meg a legnehezebb állatfajhoz tartozó tömeget! Feltételezheti, hogy a felhasználó nem ad meg egyező tömegadatokat.

e) Határozza meg, hogy ez a tömeg melyik fajhoz tartozik, és a faj nevét írja be a „legnehezebb.txt” szövegfájlba!

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki. A feladat hibátlan elvégzéséért 18 pont jár.

```
C:\Users\raerek\programok>nehez.py
Add meg egy állatfaj nevét! tőkés réce
Hány kilogramm a tömege egy példánynak? 1
Add meg egy állatfaj nevét! teve
Hány kilogramm a tömege egy példánynak? 500
Add meg egy állatfaj nevét! strucc
Hány kilogramm a tömege egy példánynak? 110
A(z) tőkés réce tömege 1 kg.
A(z) teve tömege 500 kg.
A(z) strucc tömege 110 kg.
```

Az allat.py modul tartalma

```
class Állatfaj:
    def __init__(self, fajnév, tömeg):
        self.fajnév = fajnév
        self.tömeg = tömeg
```

Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot nehez.py néven, a program hibaüzenet nélkül lefut.
2. Betölti az allat nevű modult.
3. Bekéri egy állatfaj nevét.
4. Bekéri egy állatfaj tömegét.
5. A bekért tömeget számmá alakítja.
6. A bekért adatok alapján az allat modul Állatfaj nevű osztályából egyet példányosít.
7. Három bekért adatpár alapján három Állatfaj osztályú objektumot példányosít.
8. Létrehoz adatszerkezetet a három objektumpéldány tárolására.
9. A létrehozott objektumokat tárolja az adatszerkezetben.
10. A három objektum adatinak bekérésére, létrehozására és tárolására ciklust használ.
11. Az adatszerkezetben tárolt objektumok alapján megjeleníti egy objektum jellemzőit „A(z) FAJNÉV tömege TÖMEG kg.” formában.
12. Mindhárom objektumot megjeleníti.
13. Megállapítja, hogy a tárolt objektumok tömeg jellemzői közül melyik a legnagyobb érték.
14. Megállapítja a legnehezebb állatfaj nevét.

15. Megnyitja a legnehezebb.txt fájlt.
16. A legnagyobb tömegértékhez tartozó állat nevéből mondatot képez „A(z) FAJNÉV a legnehezebb.” formában
17. A mondatot a megnyitott fájlba írja.
18. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök).