BUDAPESTI CORVINUS EGYETEM

Oktatási segédlet

Bevezetés a programozásba tantárgy oktatásához

Szerzők: Fodor Szabina, Molnár Géza

Budapest, 2025

Tartalom

Bevezetés

A *Bevezetés a programozásba* tantárgy célja, hogy a hallgatók elsajátítsák a programozás alapjait, különös tekintettel az adatok kezelésére és elemzésére, a Python programozási nyelv segítségével. A kurzus során a hallgatók megismerkednek a problémamegoldás algoritmikus megközelítésével, a programkód tervezésének, megvalósításának, tesztelésének és hibakeresésének alapelveivel. A tantárgy különös hangsúlyt fektet a strukturált gondolkodásra, az olvasható és újrafelhasználható kód írására, valamint a gyakorlati alkalmazásokra.

A feladatgyűjtemény célja, hogy támogassa a hallgatók elméleti tudásának gyakorlati elmélyítését, és lehetőséget biztosítson a tanult programozási technikák alkalmazására valószerű problémák megoldásán keresztül. A gyűjtemény feladatai lépésről lépésre vezetik be a hallgatókat a Python nyelv szintaxisába, az adattípusok és vezérlési szerkezetek használatába, a függvények és modulok alkalmazásába, valamint az egyszerűbb algoritmusok és adatstruktúrák megvalósításába.

A feladatok tematikusan követik a tantárgy során feldolgozott témaköröket, így a fejezetek olyan kulcsfontosságú területeket fednek le, mint a változók és kifejezések kezelése, ciklusok és elágazások használata, fájlkezelés, hibakezelés, valamint az adatok strukturált feldolgozása listák, szótárak és egyéb beépített adattípusok segítségével. A későbbi fejezetek bevezetést nyújtanak az objektumorientált programozás alapjaiba és az egyszerűbb algoritmikus problémák megoldásába is.

A feladatok megoldásához szükséges Python környezet (pl. Python 3.11, Jupyter Notebook vagy bármely más fejlesztői eszköz) telepítéséhez és használatához részletes útmutatók állnak rendelkezésre az egyetem Moodle rendszerében. A feladatgyűjtemény célja, hogy a hallgatók önállóan is képesek legyenek a tanultak alkalmazására, és megalapozzák további programozási ismereteik bővítését.

1. Bevezetés a Python nyelvbe

1.1 feladat: Írjon egy programot, amely megjeleníti a képernyőn: “This is my second Python program”.

a) Minden szót új sorba helyez.
b) Minden szó külön oszlopba kerül.
c) Használjon “- “ elválasztó karaktert a szavak között

1.2 feladat: Írjon programot, amely bemutatja Önt a következő formában:

a) Name: “Edit Fehér” & Neptun-code: “ABCDEF” & Age: 25 years
b) Introduction
   Name: Edit White
   Neptun-code: ABCDEF
   Age: 25 years

1.3 feladat: Írjon programot, amely a következő műveleteket végzi el, és az eredményeket a következőképpen jeleníti meg:

a) 2*1000 + 2*10 + 1*3 = 2023
b) The solution of the equation 4x + 2 = 14 x = 3
c) The number of seconds in a week is 604800
   (*számolja ki az értéket*)

1.4 feladat: Írja meg a 7-es szorzótáblát 1-től 10-ig a következő formában:
Például a 7-es szorzótábla:

1 * 7 = 7

2 * 7 = 14

…

10 * 7 = 70

(Az ötödik oszlopban ki kell számolni az értékeket)

1.5 Házi feladat: Írjon egy programot, amely egy adott hónap és év naptárát jeleníti meg a következőképpen:
November 2023

Mo Tu We Th Fr Sa Su

```
       1  2  3  4  5
6  7  8  9  10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

2. Egyszerű adattípusok

2.1 feladat: Írj egy programot, amely kiszámítja és megjeleníti a képernyőn a következőket
a) két egész szám összegét, különbségét, szorzatát és hányadosát.
b) az előző számok osztásának maradékát
c) az első számot 1-gyel növelte, a második számot pedig 1-gyel csökkentette

2.2 feladat: Írj programot, amely megold egy elsőfokú egyenletet
a) az egyenlet együtthatóit kérdezze meg a felhasználótól!

2.3 feladat: Írjon programot, amely kiszámítja és megjeleníti a képernyőn egy kocka térfogatát és felületét!
a) az élek lebegőszámok legyenek, és a felhasználó adja meg őket

2.4 feladat: Ön egy új laptopot szeretne vásárolni adott nettó áron és adókulccsal. Számítsa ki és jelenítse meg a laptop bruttó árát!
a) kérje el a felhasználótól a szükséges adatokat

2.5 feladat: A félév végén egy diák 5 tantárgyból kap jegyeket. Írjon programot, amely
a) egyenként beolvassa a jegyek értékeit
b) kiszámítja a jegyek átlagát

2.6 feladat: Egy adott összeget adott éves kamatláb mellett beteszünk a bankba, mennyi pénzünk lesz 5 év múlva? Írj egy programot, amely
a) beolvassa az összeget és a kamatlábat
b) és kiszámítja a végösszeget.

2.7 Házi feladat: Írjon programot, amely egy dollárban megadott összeget euróra és forintra konvertál!

3. Feltételes elágazások

3.1 feladat: Írjon egy programot, amely lekérdez egy egész számot, és ellenőrzi, hogy az pozitív-e vagy sem.

3.2 feladat: Írj programot, amely két lebegőszámot olvas be, és a nagyobbat jeleníti meg.

- Várható kimenet: "Az adott számok 23.4 és 11.2, a nagyobbik 23.4"

3.3 feladat: Írj programot, amely megoldja a kvadratikus egyenletet ($ax^2 + bx + c = 0$)!

- Olvassa be az a, b és c lebegőszámokat!
- Ha nincs valós megoldás, adjon hibaüzenetet, ellenkező esetben számítsa ki, és jelenítse meg az egyenlet megoldását (megoldásait)!
- Használja a kvadratikus képletet: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

3.4 feladat: Írj programot, amely három egész számot fogad el, és megkeresi a legnagyobbat.

- Várható kimenet: "A megadott számok: 14, 22, 11, a legnagyobb a 22."

3.5 feladat: A félév végén a diák három tantárgyból (fizika, matematika és kémia) kap jegyet. Az egyes jegyek 0 és 100 között vannak.

- Olvassa fel a tanuló nevét és a három jegyet.
- Számítsa ki az összes jegyet és a százalékos arányt.
- A szabályok szerint legalább 50%-os pontszámot kell elérni.
- További feltétel, hogy a tanulónak egyik tantárgyból sem lehet 40 alatti jegye.
- A tanuló nevének és az eredménynek a megjelenítése (megfelelt vagy nem felelt meg).

3.6 Házi feladat: Írjon programot, amely leolvassa a hőmérsékletet Celsius-fokban, és a hőmérséklet értékének megfelelően üzenetet jelenít meg az alábbiak szerint:

- "Forró", ha a hőmérséklet nagyobb, mint 30
- "Normál", ha a hőmérséklet 10 és 30 között van.
- "Hideg", ha a hőmérséklet 10-nél kisebb, de 0-nál nagyobb vagy azzal egyenlő
- "Fagy", ha a hőmérséklet 0 alatt van

4. Ciklusok

4.1 feladat: Írj programot, amely beolvas egy n egész számot, és 2 oszlopban megjeleníti az 1-től n-ig terjedő számokat és azok négyzetgyökeit.

4.2 feladat: Írj programot, amely összeadja az összes páratlan számot 1 és 2023 között.

4.3 feladat: Írj programot, amely ellenőrzi, hogy egy n egész szám prím-e.

- n legyen egy véletlen szám 1 és 1000 között.
- Várható kimenet: "A véletlen szám 27, nem prímszám"

4.4 feladat: Írjon programot, amely készpénzt 1000, 100, 10 és 1 értékű címletekre vált át.

- A címletek száma legyen a lehető legkevesebb.

4.5 feladat: Írj programot, amely kitalál egy n véletlen egész számot 1 és 200 között.

- Adjon tippeket a felhasználónak, mondja meg neki, hogy a kitalált szám kisebb, nagyobb vagy egyenlő-e, mint N
- Legfeljebb 8 tippet engedélyezzen.

4.6 feladat: Írjon olyan programot, amely beolvassa a felhasználó keresztnevét és vezetéknevét, és

- megjeleníti a felhasználó monogramját, amelyet a keresztnév első két betűjének és a vezetéknév utolsó két betűjének összekapcsolásával kapunk,
- majd megjeleníti a felhasználó teljes nevét úgy, hogy a betűk fordított sorrendben jelennek meg (azaz a teljes név "John White", a fordított sorrend "etihW nhoJ").

4.7 feladat: Írjon számkitalálós játékot, amelyben 6 tippből kell egy 1 és 100 közötti számot kitalálni.

4.8 Házi feladat: Írjon egy programot, amely megjeleníti az összes prímszámot "a" és "b" között (a < b). Az a-t és b-t a felhasználótól kérje be.

5. Összetett adattípusok Pythonban

5.1 feladat: Írj egy programot, amely 5 lottószámot (1 és 90 közötti egész számokat) generál, és egy listában tárolja őket.

- Jelenítse meg a lista elemeit
- Cserélje fel az első és az utolsó elemet
- A lista újbóli megjelenítése.

5.2 feladat: Egy mozi n sorból áll, minden sorban m ember fér el. A foglalt helyeket egy kétdimenziós listában tároljuk a következőképpen:

- ha a lista értéke 1, akkor a hely foglalt.
- ha nulla, akkor a hely üres.

5.3 feladat: Írj programot két 3x3-as mátrix szorzására.

- A mátrixokat kétdimenziós listákban tároljuk.
- Jelenítsd meg a mátrixokat és a szorzatot

5.4 feladat: Írj olyan programot, amely egy adott sokaság minden egyes elemének gyakoriságát megkeresi.

- Például, adott t = (2, 4, 3, 2, 2, 2, 3) tuple
- Az eredmény legyen {2: 3, 4:1, 3:2}.

5.5 feladat: Adott két keresztneveket tartalmazó lista. Írjunk programot, amely kinyomtatja

- a listák közös elemeit
- az összes olyan elemet, amely az egyik listában szerepel (mindegyikben csak egyszer).
- az összes olyan elemet, amely az első listában szerepel, de a másodikban nem (mindegyik csak egyszer)

Oldja meg ezt a feladatot halmazokkal és halmazok nélkül.

5.6 Házi feladat: Írj programot, amely megjeleníti az 1 és n közötti összes egész szám összes osztóját.

- Kérdezze meg a felhasználótól az n értékét
- Használjon szótárat az adatok tárolására a következők szerint: {1: [1], 2: [1, 2], 3: [1, 3], 4: [1, 2, 4]}
- (Választható) Mely számoknak van páratlan számú osztója?

6. Alapvető algoritmusok I.

6.1 feladat: Egy osztályban 10 diák van, mindegyiküknek ismert a magassága. Döntsük el, hogy van-e közöttük 185 cm-nél magasabb!

- Tároljuk az adatokat egy listában
- Használjuk a Döntés algoritmust

6.2 feladat: Számítsa ki a heti átlaghőmérsékletet, ha az egyes napok átlaghőmérséklete ismert.

- 10,0 és 25,0 Celsius-fok közötti véletlenszerű lebegőszámok generálása (a 25,0 nem szerepel).
- Tárolja az adatokat egy tuple-ben
- Használja az összegzés algoritmusát

6.3 feladat: Készítsen programot, amely megszámolja, hány osztója van egy adott pozitív egész számnak.

- Kérje a számot a felhasználótól
- Ha az osztók száma 2, akkor írja ki: ez egy prímszám.
- Máskülönben print: ez nem prímszám.
- Használja a számolás algoritmusát

6.4 feladat: Dobjunk fel egy tisztességes érmét n-szer (n 10 és 100 között van). Hány dobás után kapunk először fejet?

- Az n értékét és a dobások értékét véletlenszerűen kell generálni.
- az adatokat megfelelő adatstruktúrákban tároljuk, és feltételezhetjük, hogy mindig van fej a dobások között
- Használjuk az adott tulajdonságú elem kiválasztásának algoritmusát

6.5 feladat: Adott n keresztnév a kérdés, hogy "Péter" köztük van-e? Ha igen, akkor a program adja meg, hogy hányadik keresztnév az.

- Az eredmény egy szám lesz.
- Ha Péter nincs a névlistában, akkor a visszaadott érték -1 legyen.
- Használja a lineáris keresés algoritmusát

6.6 feladat: Generáljon véletlenszerűen 15 egész számot 1000 és 2023 között, és határozza meg közülük a legnagyobbat.

- Használjuk a maximális kiválasztás algoritmusát.

6.7 Házi feladat: Egy számot tökéletes számnak nevezünk, ha egyenlő a nála kisebb osztóinak összegével.
Írj programot, amely megtalálja az első tökéletes számot egy adott intervallumban.

- Az intervallum alsó és felső határai véletlenszerűen generálódnak.
- ha nincs eredmény, adja meg a következő üzenetet: "nincs tökéletes szám ebben az intervallumban".

7. Alapvető algoritmusok II.

7.1 feladat: Adott N darab hőmérsékletérték (valós számok) Celsius fokban. Alakítsuk át őket Fahrenheitre, majd jelenítsük meg a képernyőn az eredeti és az új értékeket is!

7.2 feladat: Adott N tanuló vizsgaeredménye (0 és M közötti egész számok). A vizsga sikeres letételéhez legalább 50%-os eredmény szükséges. Válasszuk ki és jelenítsük meg a vizsgán megfelelt hallgatók pontszámát!

7.3 feladat: Egy kockával N-szer dobunk, és a dobások értékeit egy listában tároljuk. Válasszuk szét a páros és páratlan értékeket két külön listába.

7.4 feladat: Generáljon 5 véletlen számot, majd rendezze őket növekvő sorrendbe a swap sort algoritmus segítségével. Mutassa meg minden egyes lépésnél, amikor egy csere megtörténik.

7.5 feladat: Adott N lebegőszám. Rendezze őket csökkenő sorrendbe a maximális kiválasztási rendezési algoritmus segítségével. Mutassa meg az egyes lépéseket, amikor egy csere megtörténik.

7.6 feladat: Adott egy mondat. Rendezze betűit növekvő sorrendbe ábécé szerint a buborékrendező algoritmus segítségével.

7.7 Házi feladat: Van egy N nevet tartalmazó listánk. Válasszuk ki azokat, amelyek rövidebbek az átlagos hosszúságnál, egy új listába.

8. Eljárások, függvények, modulok

8.1 feladat: Írjon egy függvényt, amely ellenőrzi, hogy egy N szám prím-e!

- keresse meg a 2-től √N-ig terjedő osztókat
- Használd ezt a függvényt az 1 és 100 közötti prímszámok megjelenítésére!
- Opcionálisan: használjon más prímszámtesztet, például a Fermat-módszert. A Fermat-módszer esetében próbáld ki egy álprímszámmal is, pl. 341.

8.2 feladat: Írjon egy eljárást az első N Fibonacci-szám megjelenítésére!

- Az N legyen kötelező paraméter
- A nulladik és az első elem legyen opcionális paraméter, alapértelmezett értéke
- Az alábbi képlet segítségével számítsa ki az n-edik Fibonacci-számot: F_n=F_(n-1)+F_(n-2) (n > 1)

8.3 feladat: Írjunk függvényt az n szám faktoriálisának kiszámítására. Ennek segítségével ..

- készítsen új függvényt a binomiális együttható kiszámítására a következő képlet szerint:
- nCr = n!/(r! * (n-r)!)
- hozzon létre egy eljárást a Pascal-háromszög első N sorának megjelenítésére.

8.4 feladat: Írjon egy programot, amely három logikai részre van osztva:

- Az első rész n véletlen egész számot generál a és b között, és egy listába helyezi őket.
- A második rész a lista elemeit növekvő sorrendbe rendezi (bármilyen tanult rendezési algoritmust használhat).
- A harmadik rész megjeleníti az eredeti és a rendezett listát.
- Használjon függvényeket és/vagy eljárásokat a feladat megoldásához. Kérdezze el a felhasználótól az n, a és b értékeit.

8.5 feladat: Írjon egy olyan függvényt, amely egy paraméterként kapott szöveg karaktereinek gyakoriságát tartalmazó szótárat ad vissza.

- Például, ha a szöveg a következő: "Hello World",
- Az eredmény legyen: {H: 1, e: 1, l: 3, o: 2, W: 1, r: 1, d: 1 }
- Kettőzzük meg a szótár minden egyes értékét a map és lambda függvények segítségével!
- Szűrje a szótárat filter és lambda függvények segítségével úgy, hogy csak azokat a kulcs-érték párokat tartja meg, amelyek értéke osztható 3-mal!.

8.6 feladat: Írjon egy olyan modult, amely függvények segítségével alapvető műveleteket hajt végre háromdimenziós vektorokon.

- Megvalósítandó: összeadás, kivonás, szorzás egy számmal, ponttétel
- Írjon egy olyan programot, amely importálja ezt a modult, és végrehajt néhány vektorműveletet (minden típusból legalább egyet).

8.7 Házi feladat: Írjon egy Python függvényt annak ellenőrzésére, hogy egy szám palindromikus-e vagy sem. (A palindromikus szám olyan szám (például a 121), amely ugyanaz marad, ha a számjegyeit felcseréljük.)
A függvény segítségével jelenítse meg az összes palindromikus számot 1 és 500 között.

9. Összetett adattípusok II.
Szöveges fájlok

9.1 feladat: Írjon programot, amely megjeleníti az 'a.txt' forrásfájl tartalmát!

- Először karakterenként olvassa be a fájlt
- Másodszor olvassa be a fájlt soronként
- Harmadszor, olvassa be egyszerre az egész fájlt

9.2 feladat: Írjon programot, amely statisztikát készít a „b.txt" forrásfájl tartalmáról az alábbiak szerint:

- A sorok száma (a len függvény helyett használjuk a számláló algoritmust).
- A fájlban lévő szavak száma (oldja meg split és len függvényekkel és anélkül)
- Az egyes magánhangzók száma

9.3 feladat: A 'c.csv' forrásfájl egy struktúrát tartalmaz (BookID, BookTitle, BookAuthor, BookPrice), az elválasztó karakter pontosvessző. **Írjunk függvényt, amely visszaadja a BookAuthor szerint rendezett sorokat!**

- Bármilyen megtanult rendezési algoritmust használhat.
- Ne használja a beépített rendezési függvényt.
- Ezzel a függvénnyel írja ki az eredményt a 'sorted_c.txt' fájlba.
- Ne használjon külső modult.

9.4 feladat: A 'c.csv' forrásfájl struktúrája (BookID, BookTitle, BookAuthor, BookPrice). Írjon eljárásokat a következő feladatok megoldására:

- Sorolja fel azokat a könyveket, amelyek szerzőjének neve B betűvel kezdődik.
- Mutassa ki a képernyőn, hogy az egyes szerzőknek hány könyve van.
- Mekkora a különbség a legdrágább és a legolcsóbb könyv ára között? (Használja a minimum/maximum kiválasztási algoritmust)

9.5 feladat: Írjon eljárásokat/függvényeket a következő fájlműveletek végrehajtására a paraméterként kapott szöveges fájlon:

- létrehozás, átnevezés, törlés, csatolás
- mindegyik esetben használjon kivételkezelést
- mentse őket egy új modulba

9.6 Házi feladat: A 'c.csv' forrásfájlnak a struktúrája (BookID, BookTitle, BookAuthor, BookPrice). Írjon függvényt, amely visszaadja a paraméterként kapott szerző könyveinek átlagárát!

10. OOP Python programozási nyelvben

10.1 feladat: Írjon egy Person nevű osztályt olyan példányváltozókkal, mint id, name, birth_year, salary!

- A konstruktor segítségével állítsuk be a változók értékeit!
- Hozzon létre descructor-t, amely megjelenít egy üzenetet: "Viszlát, név"
- Hozzon létre egy p nevű Person példányt a következő adatokkal: id: 2023, name: 'Joe White', birth_year: 1995, fizetés: 5600

10.2 feladat: Adjuk hozzá a következő metódusokat a Person osztályhoz a kapcsolódó változó értékének beállításához/megkapásához

- set_salary
- get_name
- Ezután használja a @property a birth_year értékének beállításához/megkapásához!

10.3 feladat: Örököljön egy új osztályt a Person osztályból Student néven!

- Adjon hozzá két új példányváltozót a Student osztályhoz: neptuncode és semester.
- Hozzon létre konstruktort a Student osztályban, amely a szülő osztály konstruktorát hívja meg.
- Hozzon létre egy metódust, amely visszaadja a diák életkorát években kifejezve!
- Hozzon létre egy metódust a Student példányok nyomtatásához
- Tesztelje a Student osztályt legalább egy példány létrehozásával!

10.4 feladat: Hozzon létre egy új osztályt Shape néven. Tartalmazzon (üres) metódusokat a terület és a kerület kiszámításához. Tárolja a PI hozzávetőleges értékét egy rejtett osztályváltozóban.

- Hozzunk létre alosztályokat Téglalap és Kör néven. Írjuk felül a terület és a kerület metódusokat a megfelelő paraméterekkel és képletekkel!
- Módosítsa a Téglalap osztályt, amely automatikusan négyzetet hoz létre, ha csak a szélesség paramétert adja meg!
- Tesztelje az osztályokat példányok létrehozásával és metódusok hívásával!

10.5 feladat: Hozzon létre egy olyan Frakció osztályt, amely az operátorok túlterhelését használja a törtek négy alapműveletének (+, -, *, /) megvalósítására!

- Írja felül a __str__() metódust a frakció objektumok string reprezentációjának definiálásához.
- Módosítsa a Fraction osztályt: ha csak a számlálót adja meg, akkor a nevező alapértelmezés szerint 1 lesz (egész szám).
- Teszteljük a Fraction osztályt a következők szerint:
    - f1 = Fraction(1, 2)
    - f2 = Tört(3, 4)
    - f3 = Tört(4)
    - print(f1 + f2) # 5/4

- print(f1 - f2) #-1/4
- print(f1*f2) # 3/8
- print(f1/f2) # 2/3
- print(f3-f1) #7/2

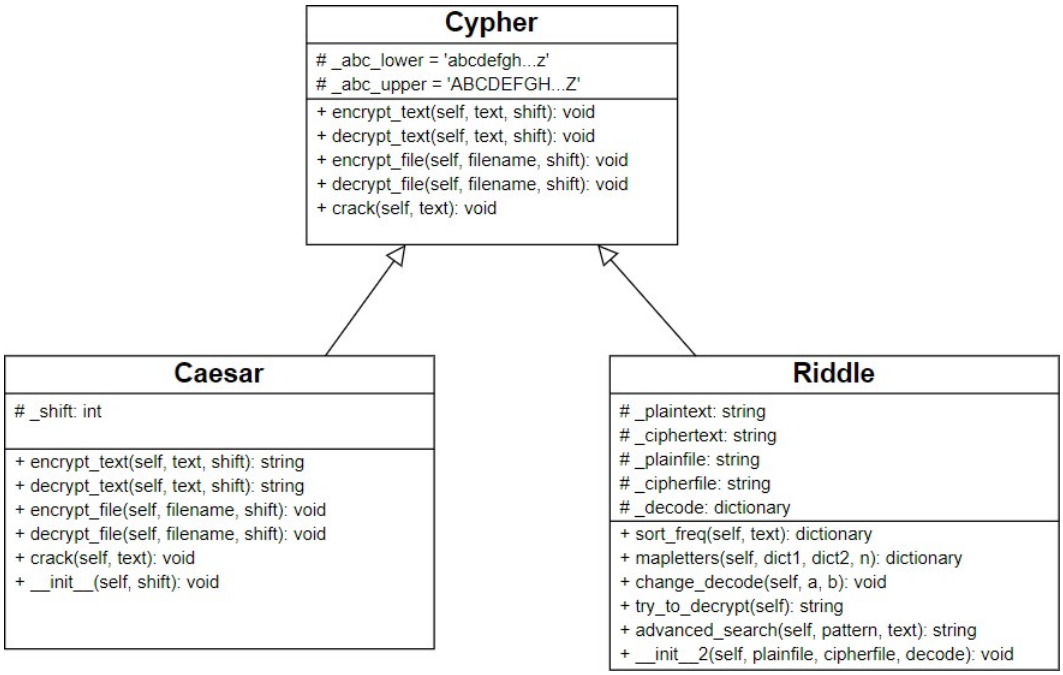
10.6 Házi feladat: A létrehozott Fraction osztály új metódusokkal való bővítése a következő műveletek végrehajtásához: ==, <, >.
Minden esetben használjuk az operátorok túlterhelését!

11. Egy összetett probléma megoldása Python nyelven

11.1 Készítsen egy objektumorientált programot Python nyelven, amely a következő feladatokat valósítja meg:

- Szövegek és fájlok kódolása és dekódolása Caesar titkosítással
- A szöveg dekódolása gyakoriság elemzéssel

Az osztályokat az alábbi UML-diagram alapján kell létrehozni:



A következő táblázat az adatok leírását tartalmazza:

Osztály	Adatok	Statikus/tárgy	Leírás
Cypher	_abc_lower	Statikus	Az angol ábécé kis betűi _abc_lower = 'abcdefghijklmnopqrstuvwxyz'
Cypher	_abc_upper	Statikus	Az angol ábécé nagybetűi _abc_upper =
Caesar	_shift	Objektum	Az eltolás értéke a Caesar-
Ridle	_plaintext	Statikus	A plainfile szövegét tartalmazó karakterlánc
Ridle	_ciphertext	Statikus	A titkosírási fájl szövegét tartalmazó karakterlánc
Ridle	_decode	Objektum	Szótár, amely a titkosított és a visszafejtett szövegek közötti leképezést tartalmazza.
Ridle	_cipherfile	Objektum	A titkos szöveget tartalmazó fájl neve
Ridle	_plainfile	Objektum	Az egyszerű szöveget tartalmazó fájl neve

A következő táblázat a módszerek leírását tartalmazza:

Osztály	Módszer	Statikus/tárg	Leírás
Cypher	encrypt_text	Objektum	üres módszer, csak öröklési céllal
Cypher	decrypt_text	Objektum	üres módszer, csak öröklési céllal
Cypher	encrypt_file	Objektum	üres módszer, csak öröklési céllal
Cypher	decrypt_file	Objektum	üres módszer, csak öröklési céllal
Cypher	crack	Objektum	üres módszer, csak öröklési céllal
Caesar	__init__	Objektum	a _shift változó kezdeti értékének beállítása
Caesar	encrypt_text	Objektum	a szöveget Caesar titkosítással titkosítja.
Caesar	decrypt_text	Objektum	a szöveg visszafejtése Caesar visszafejtés
Caesar	encrypt_file	Objektum	titkosítja a fájlt Caesar titkosítással, és elmenti azt a lemezre
Caesar	decrypt_file	Objektum	visszafejti a fájlt a Caesar dekódolás segítségével, és elmenti azt a lemezre
Caesar	crack	Objektum	visszafejti a szöveget az eltolás kitalálásával (megpróbálja a lehetséges értékek)
Ridle	sort_freq	Objektum	létrehoz egy szótárat a szövegben található karakterek gyakorisága alapján, majd csökkenő sorrendbe rendezi azt. értékek szerinti sorrend
Ridle	mapletters	Objektum	létrehoz egy szótárat, amelynek kulcsai az első szótárból származnak, az értékek pedig a második szótárból a sorrendnek megfelelően. Az alapértelmezett érték az "n" paraméter értéke 26.
Ridle	change_decode	Objektum	megtalálja a kulcsot a _decode szótárban, amely a _decode szótárhoz tartozik az 'a' értékét, és megváltoztatja az értékét 'b'-re.
Ridle	try_to_decrypt	Objektum	beolvassa a _chiphertextet és visszaadja a visszafejtett szöveget a _decode használatával
Ridle	advanced_searc	Statikus	regex mintát talál egy szövegben
Ridle	__init__	Objektum	beállítja a plainfile, cipherfile és decode kezdeti értékeit. A decode paraméter opcionális. Ha a decode nincs megadva, akkor azt a következővel számítja ki mapletters módszer _plaintext és _chipertext paraméterekkel. A dekódolás nagybetűket használ.

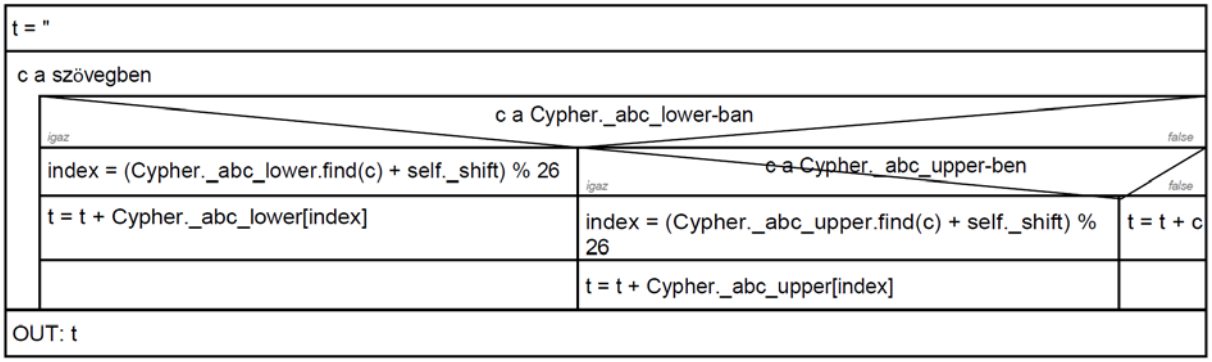
Bemeneti adatok:

- szöveg, amelyen megpróbálja Caesar titkosítást (shift = 3)
  But consider! I said, earnestly.
- szöveg, amelyen megpróbálja Caesar dekódolás (shift = 3)
  Exw frqvlghu! L vdlg, hduqhvwob.
- c.csv
  szükséges a Caesar osztály encrypt_file metódusához

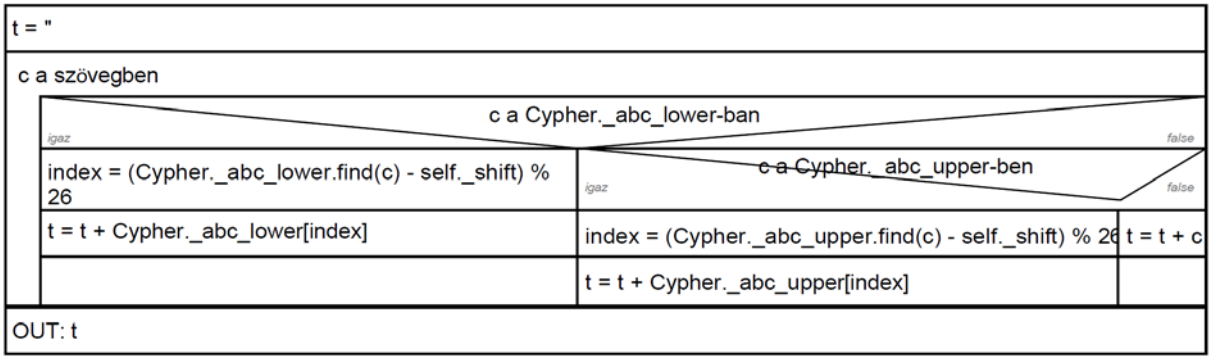
- caesar_decrypted. txt
  szükséges a decrypt_file módszerhez a Caesar osztályban
- plaintext.txt
  szükséges az __init__ eljáráshoz a Riddle osztályban
- ciphertext.txt
  szükséges az __init__ eljáráshoz a Riddle osztályban

A nem triviális algoritmusok leírása:

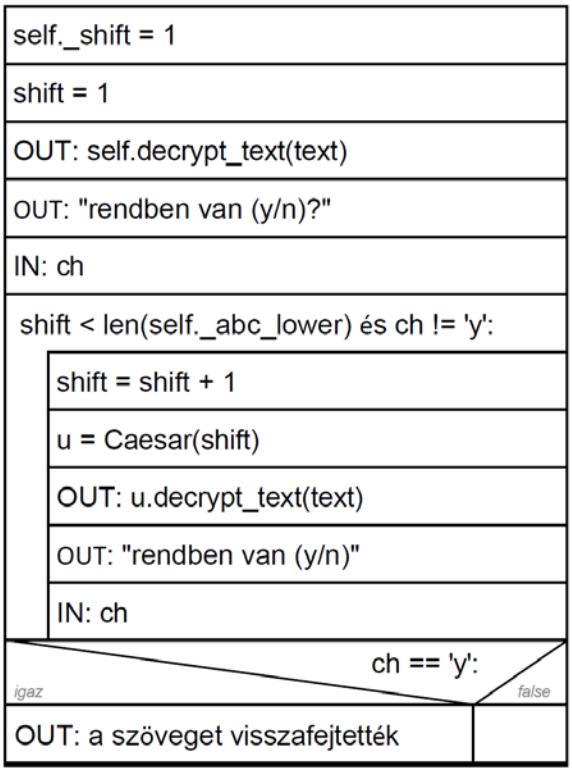
Caesar encrypt_text:



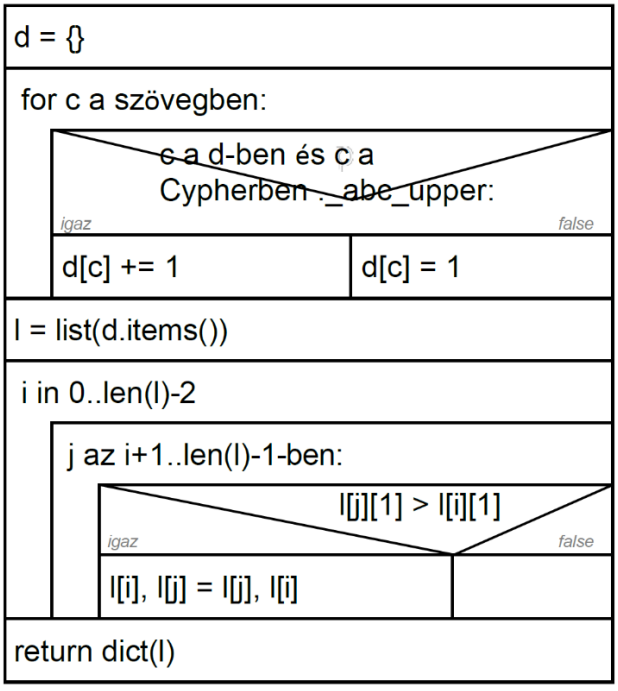
Caesar decrypt_text:



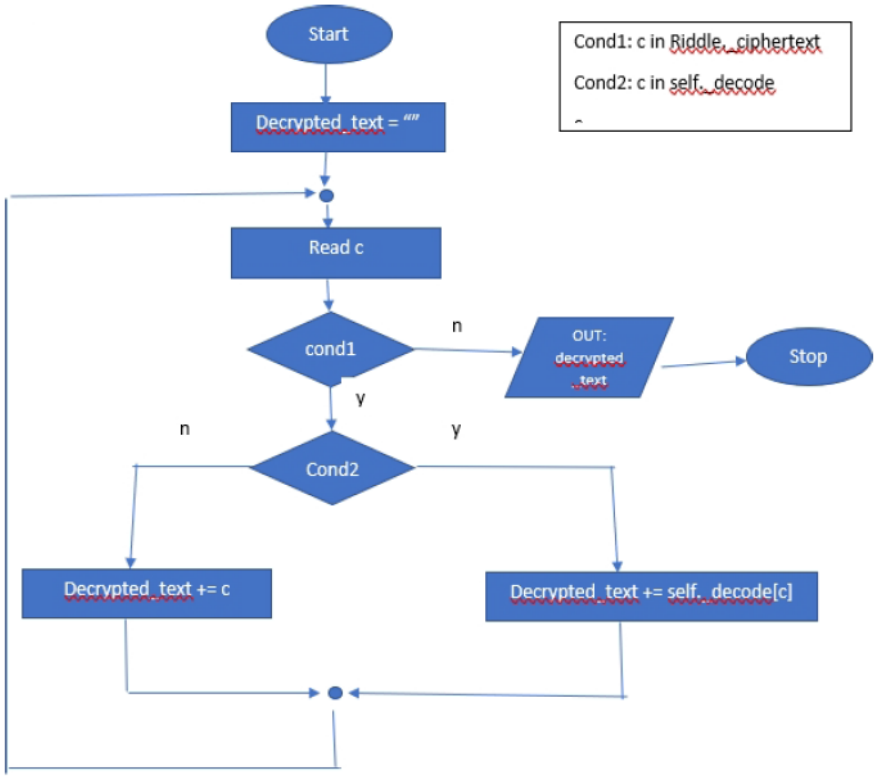
Caesar crack:



Riddle sort_freq:



Rejtvény try_to_decrypt:



Caesar-kód

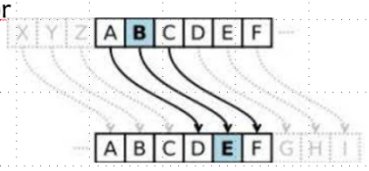
A Caesar-kód egy olyan helyettesítő kódolás, amelyben az egyszerű szöveg minden egyes betűje az ábécében bizonyos számú helyet lefelé vagy felfelé tolódik. Ezt az eltolást kulcsnak nevezzük.

Képlet: E_n x = (x + n) % 26

Ha shift = 3, akkor

- A-ból D lesz
- B-ből E lesz
- A C-ből F lesz

és így tovább



Ezt a kódot könnyű feltörni, mivel csak 25 lehetséges kulcs van.

Példa:

```
abc_upper = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
shift = 3
message = 'HELLO WORLD' #use capital letters only

#encoding
encoded_message = ""
for c in message:
  if c in abc_upper:
    index = 0
    while abc_upper[index] != c:
        index = index + 1
    c = abc_upper[(index + shift) % len(abc_upper)]
  encoded_message += c
print(encoded_message)

```

KHOOR ZRUOG

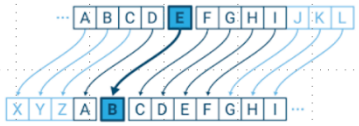
Caesar megfejtése

A Caesar-fejtés során a betűket az eredeti titkosítással ellentétes irányba kell eltolni.

Képlet: D_n x = (x - n) % 26

Ha shift = 3, akkor
- A D-ből A lesz
- Az E-ből B les:
- F-ből C lesz

és így tovább



Példa:

```
abc_upper = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
shift = 3
encoded_message = 'KHOOR ZRUOG' #use capital letters only

#decoding
decoded_message = ""
for c in encoded_message:
  if c in abc_upper:
    index = 0
    while abc_upper[index] != c:
        index = index + 1
    c = abc_upper[(index - shift) % 26]
  decoded_message += c
print(decoded_message)

```

```
HELLO WORLD
```

11.2 Készítsen olyan programot, amely kezeli a diákok vizsgajegyzékét!

- A programnak lehetővé kell tennie a felhasználók számára, hogy CRUD műveleteket (Create, Read, Update, Delete) hajtsanak végre szöveges fájl segítségével. Az OOP megközelítés használata előnyös.

Lépésről lépésre útmutató a feladat megoldásához:

- Definiáljon egy Student nevű osztályt, amely a diákokat és vizsgaeredményeiket reprezentálja. Az osztálynak olyan attribútumokkal kell rendelkeznie, mint name, id és scores. A pontszámokat egy szótárban kell tárolni kulcs-érték párokkal, mint subject: score.
- Implementáljon módszereket a Student osztályon belül a CRUD műveletek végrehajtásához:
  - Új hallgatói rekordot hoz létre a hallgató adatainak és vizsgaeredményeinek elfogadásával.
  - Megjeleníti az összes meglévő diákrekordot.
  - Egy adott diák vizsgaeredményeinek frissítése.
  - Egy adott diákrekord törlése.
- Használjon szöveges fájlt a diákrekordok tárolására. A fájl minden egyes sora egyetlen tanulói rekordot képviseljen, a következő formátumban: név, azonosító, pontszámok.
- A következő módszerek végrehajtása a szöveges fájl kezeléséhez
  - A tanulói rekordokat egy szöveges fájlba menti.
  - Betölti a diákrekordokat a szöveges fájlból.
- A program indításakor töltse be a meglévő tanulói rekordokat a szöveges fájlból (ha vannak).
- Implementáljon egy főprogramot, amely megjelenít egy menüt a felhasználónak, és a kiválasztás alapján meghívja a megfelelő metódusokat.
- Amikor a program véget ér, mentse a tanulói rekordokat a szöveges fájlba.

12. Gyakorló feladatok

Rövid gyakorlatok - lehetőleg csak algoritmikus struktúrákat használjunk

12.1 Írjon egy rövid programot vagy függvényt, amely visszaadja egy adott egész szám szám számjegyeinek számát!

Például:
x = 5436 --> kimenet: # x = 5436 --> output: 4

12.2 Írj egy rövid programot vagy függvényt, amely egy karakterláncban az összes nagybetűt kicseréli kisbetűkre és fordítva.

Például:
kimenet: hELLO wORLD

12.3 Írj egy rövid programot vagy függvényt, amely eltávolítja egy adott elem összes előfordulását egy listából.

Például:
item = 4
l = [4, 5, 3, 3, 2, 4, 1] output: [5, 3, 3, 2, 1]

12.4 Van egy szótárunk. Írjunk egy rövid programot vagy függvényt, amely visszaadja, ha egy kulcs szerepel a szótárban.

Például:
kulcs = "ország"
d = {'name': 'Jessica', 'country': 'UK', 'phone': 1111} --> output: True

12.5 Van egy tuple. Írjunk egy rövid programot egy olyan függvényről, amely visszaadja egy adott szám előfordulási számát.

Például:
num = 8
t = (3, 4, 4, 4, 8, 1, 11, 8) --> kimenet: 2

Hosszabb feladatok a grades.csv fájl alapján

12.6 Hozzon létre egy függvényt, amely eldönti, hogy egy adott azonosítóval rendelkező diák létezik-e a grades.csv fájlban.

Például:
student_id = "98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1" --> kimenet:
student_id = „98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1” → output: True

12.7 Hozzon létre egy függvényt, amely visszaadja egy adott diák jegyeit! Kerekítsen 2 tizedesjegyre.

Például:
student_id = "98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1" -->
output: 86.79 86.29 69.77 55.10 49.59 44.63

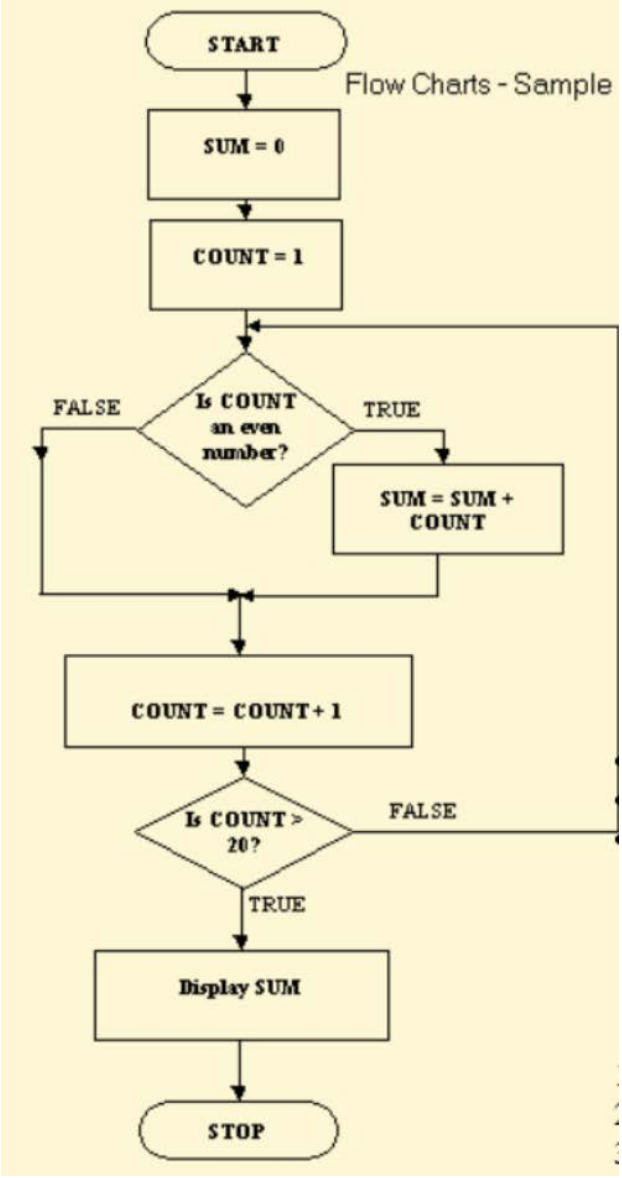
12.8 Készítsen egy függvényt, amely visszaadja az első feladat legjobb osztályzatát!

12.9 Írjon egy függvényt, amely visszaad egy szótárat, amely a kulcsokat évszámként, az értékeket pedig az adott évben történt beosztások1 számaként tartalmazza!

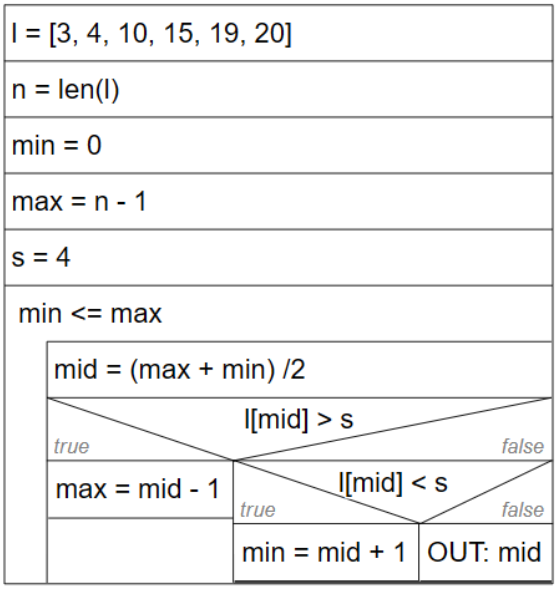
12.10 Írj egy függvényt, amely megadja egy adott feladat átlagos osztályzatát.

Gyakorlatok algoritmus leíró eszközökkel

12.11 Flowchart



12.12 Structogram



12.13 Pseudocode

```
In: n > 3
p = 2
factors = []
loop while n >= p*p
   if n % p =0
        factors += p
        n = n / p
   else
        p = p + 1
   end if
end loop
factors += n
Out: factors
```

13. Minta ZH feladatok

13.1 A **hianyzas.txt** szöveges fájl egy szóközzel elválasztott szöveges állomány, amely egy félévben egy adott osztály hiányzásait tartalmazza. A hiányzások napok szerint vannak csoportosítva, minden nap # karakterrel kezdődik, majd a hónap sorszáma, majd a nap sorszáma következik egy-egy szóközzel elválasztva.

Az adott napra vonatkozó hiányzások tanulónként külön sorban szerepelnek, a tanuló napi hiányzásait hét karakterből álló karaktersorozat írja le.

Ahol O: ha a tanuló jelen volt az adott órán, X: igazolt a hiányzás, I: igazolatlan hiányzás, N: a tanulónak az adott időpontban nem volt órája.

Például:

Január 16-án Alma Hedvignek hét órája volt, de csak hat órán volt jelen, és igazolatlanul hiányzott.

```
# 01 16
Alma Hedvig OOOOOOOIO
```

Készítsen Python nyelvű programot, amelyik az igazolatlanul hiányzó tanulók nevét megjeleníti (minden név csak egyszer jelenjen meg, akkor is, ha több alkalommal, több órát hiányzott)

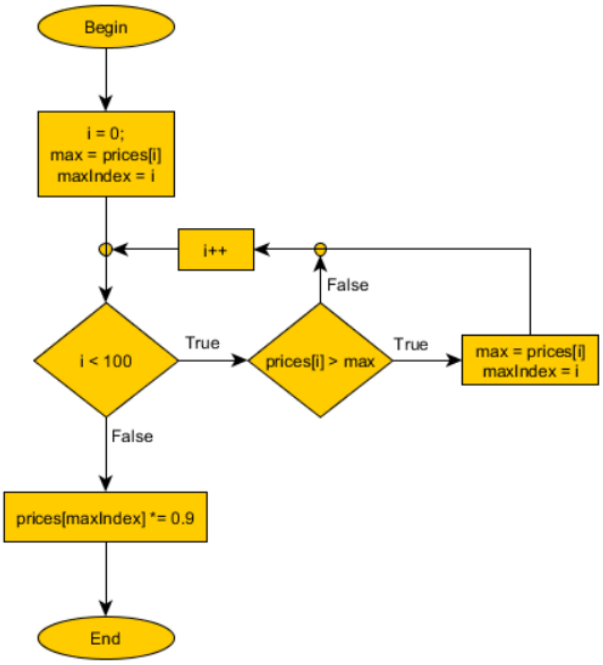
13.2 Készítsen egy rövid programot vagy függvényt, amely egy egész számot vár bemenő értékként és a szám középső számjegyét / számjegyeit adja vissza eredményül.
Pl. be: 12345 kimenet: 3
be: 123456 kimenet:34

13.3 Készítsen egy programot vagy függvényt, amelyik egy kosárlabda csapat bajnokságának adata alapján eldönti, hogy hány fordulón keresztül tartott a leghosszabb győzelmi széria, melyben a csapat megszakítás nélkül legyőzte ellenfelét. (A csapat 12 mérkőzést játszott és a győzelem 3, a döntetlen 1, a vereség 0 pontot ér.)

pl. be [3, 3, 3, 1, 0, 3, 0, 0, 1, 1,1,0] kimenet:3

be [1, 1, 1, 1, 1, 0, 0, 0, 0, 1,1,1] kiment: 0

13.4 Készítsen el a következő folyamatábra alapján a Python nyelvű implementációt.



13.5 Egy szótár kulcsként és értékként tantárgyakat és osztályzatokat tartalmaz. Írjon egy függvényt, amely ezt a szótárat használja paraméterként, és visszaadja az osztályzatok átlagát!

Például:

d = {'matematika': 5, 'természettudomány': 4, 'történelem': 3}

kimenet: print(atlag_jegy(d)) --> 4