**Programozás tréning**

[Jelmagyarázat 1](#_30j0zll)

[Bevezető 2](#_1fob9te)

[1.](#_3znysh7) Személyes bemutatkozás 2

[2.](#_2et92p0) EIK és a programozás tréning bemutatása, célja 2

[3.](#_tyjcwt) EIK bemutatása 3

[Szemlélet 3](#_3dy6vkm)

[4.](#_1t3h5sf) Programozás lényege 3

[5.](#_4d34og8) Algoritmizálás 3

[6.](#_2s8eyo1) Kódolás, programtervezés 6

[7.](#_17dp8vu) Programozási tételek 7

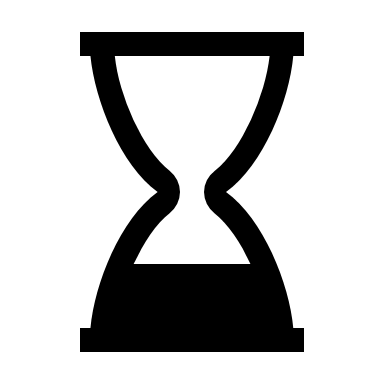
[8.](#_3rdcrjn) Jó programozó 8

[Befejezés 8](#_26in1rg)

[Clean Code feladat 9](#_lnxbz9)

## Jelmagyarázat

* Komment a trénereknek.
* Interaktív rész. Ha nagyon inaktívak, nem kell erőltetni.

 Ez a rész akkor, ha van rá idő.

## Bevezető

### Személyes bemutatkozás

* Mivel a tréningen alapból körben ülnek, érdemes csatlakozni hozzájuk a bemutatkozás erejéig, majd utána megkérni őket, hogy üljenek félkörbe, mivel a tréning tábla előtt fog zajlani.
* Itt mindenki szabadságot kap, de szeretnénk, ha közvetlen, pozitív hangulatú és építő jellegű lenne. Ha bármikor párbeszéd alakul ki a hallgatók, vagy a hallgatók és a tréner között, azt hagyjuk kibontakozni, csak akkor rekesszük be, ha értelmetlennek találjuk.
* Mit érdemes elmondania a trénernek a bemutatkozás során?
  + Név, szak, ha programtervező informatikus, akkor szakirány, ha tanáris akkor a két tárgy.
  + Miért, mióta vagy az egyetemen, mi a célod a diplomával? Bsc. után Msc., vagy akár doktori, esetleg egyből mennél dolgozni?
  + Megkérdezhetjük, ők miért jöttek az egyetemre, miért szeretnének az informatikával foglalkozni. Ki mivel szeretne foglalkozni? Játékfejlesztés, mesterséges intelligencia...?
  + Dolgoztál-e már a szakmában, vannak-e tapasztalataid, amit érdemes lenne megosztani?
  + Megkérdezhetjük, van-e álomcégük, ahol szeretnének dolgozni. Esetleg saját céget szeretnének-e alapítani?
  + Mi az, ami szerinted első félévben fontos? (tanácsok a gólyáknak, pl. fontos a Matematikai alapok.
* Érdemes mindenkinek a bemutatkozást előre átgondolni.
* A Matematikai alapok fontosságára érdemes lenne itt 2 mondatban felhívni a figyelmet. Pl.: erre épül minden matek, hamar le lehet maradni, amit aztán nehéz pótolni.

### EIK és a programozás tréning bemutatása, célja

* A tanulásmódszertan tréning általános, de mi konkrét programozási és matematikai tréninggel készültünk: ezen az alkalmon programozás tréning lesz, a szeptember végén (szeptember 21 és 28) pedig matek.
* **A programozás tréningről:** a tanulásmódszertan kurzus része, 90 perces, szünet nélkül tartjuk. A tréning három fő része:
  + **Programozási alapok:** Algoritmizálás, kódolás.
  + **Az egyetemi programozás oktatás bevezetése:** Miben más, miben több az ELTE IK-n a programozás, mint például egy középiskolában, vagy OKJ-s tanfolyamon? *(papíron programozás, dokumentáció, specifikáció…)*
  + **Jó programozó bemutatása:** Mire kell törekedni az egyetemi anyag elsajátítása mellett? *(clean code, soft skillek…)*
  + A tréning tartalmának vázlatát érdemes felírni a táblára
* **A tréning célja:** az elsőévesek programozási ismereteinek megalapozásával segítjük a Programozás (korábbi nevén: Programozási alapismeretek) tárgy elvégzését, majd a tréning végén olyan ismereteket vezetünk be, melyek segítségükre lehetnek mind az egyetemi pályafutásuk alatt, mind a karrierjükben.

### EIK bemutatása

* Pár mondatban ismertetjük az EIK-et, hogy a gólyák be tudják határolni, kik vagyunk…
  + Az EIK az Eötvös Informatikai Kör rövidítése.
  + Az ELTE IK PTI és tanáris hallgatói.
  + A létrejöttének célja az, hogy segítse a kar hallgatóit.
  + Saját tapasztalatok, élmények.

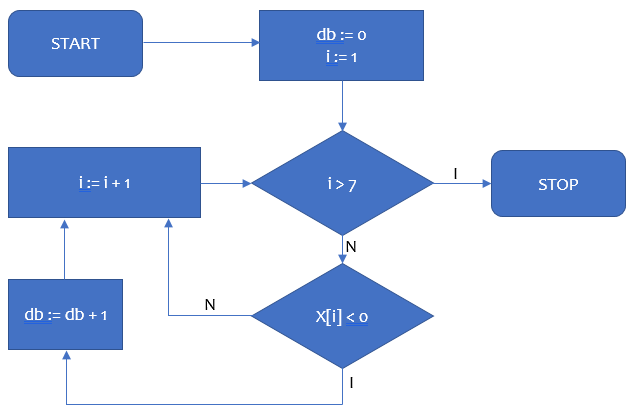
## Szemlélet

### Programozás lényege

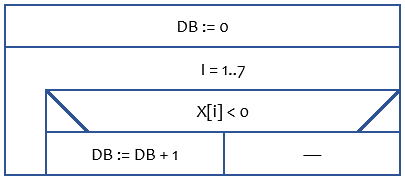
* **Kérdés:** Ki programozott már közületek? Mennyit? Milyen nyelven? Mit tanultatok középiskolában? Csak kódoltatok, vagy volt más is?
* **Az egész tréning alatt az olyan fogalmakat, amelyek újak a gólyáknak , érdemes felírni a táblára és megbizonyosodni róla, hogy értik azt.**
* „Nem baj, ha még nem programoztál, itt nulláról kezdik a dolgokat, de fel kell készülni, hogy gyors a tempó!”
* Valószínűleg a legtöbb gólya programozott már valamilyen szinten. Ha úgy vesszük észre, hogy nagyon profi a csapat, akkor nem kell túlrészletezni a bevezető dolgokat.
* **A program céljának egyik megfogalmazása:** a bemenő adatok feldolgozása, az előírt feladat megoldása.
* **Nagyon konyhanyelven a programozás lényege:** egy nagy feladat felbontása kisebb részfeladatokra, amíg olyan műveletekhez nem jutunk, melyeknek a megoldása triviális. *(Például egy recept akkor jó, ha részletesen vannak leírva a lépések; egy program nem működik jól, ha nem elég pontosan határozzuk meg a lépéseket, nem térünk ki minden esetre.)*
* A képzés elején egyszerű programokkal fogtok foglalkozni, ami a haladóbbaknak gagyinak tűnhet. **Fontos azonban, hogy átszokjanak ők is arra, hogy az IK szabályai alapján programozzanak.**

### Algoritmizálás

* **Algoritmus fogalma:** egy utasítássorozat, amely megoldja a felmerülő problémát. Például mikor útbaigazítasz valakit, lépésenként magyarázol – ez egy algoritmus.
* Kezdésnek kérdezzük meg, van-e vállalkozó, aki szeretne felrajzolni a táblára egy folyamatábrát, vagy struktogrammot.
* Valószínűleg a folyamatábra lesz a legismertebb, viszont hozzá kell szoktatnunk őket, hogy itt struktogrammot fognak használni. Azt ne hangsúlyozzuk, hogy f.-t nem fognak haszn.!
* **Kérdés:** Foglalkozott már valaki **folyamatábrával?** Esetleg **struktogrammal?** Tapasztalatok?
* **Érdemes az algoritmusleíró eszközöknek utánanézni, egy jó összefoglaló a mellékletek között található.**
* **Algoritmusleíró módszerek bevezetése:** Egy programot nem csak kóddal lehet leírni! A programtervezési fázisban érdemes az algoritmust felvázolni, hogy jobban átlássuk.
* Erre használható többféle algoritmusleíró módszer: az IK-n a képzés elején struktogrammot használunk, de mi most folyamatábrával is foglalkozunk.
* **Írjuk fel a táblára a feladat szövegét: adott egy januári héten a minden nap mért legalacsonyabb hőmérséklet Celsiusban. Számoljuk meg, ezek közül hány napon volt fagypont alatti a hőmérséklet! (<0)**
* Szükségünk van tehát egy algoritmusra, amely megszámolja, a 7 darab hőmérséklet közül melyik negatív, és ezt az algoritmust valahogy ábrázolnunk kell. **Kezdjük a folymatábrával**.
  + Itt érdemes lehet a tömb fogalmát tisztázni.
* Kérdezzük meg, van-e valaki, aki fel tudja sorolni a folyamatábra elemeit és felrajzolni őket a táblára. Ha nincs, akkor rajzoljuk fel mi, esetleg javítsunk:
  + Program belépési pontja, kilépési pontja 
  + Utasítás 
  + Döntési pont, elágazás 
  + Program folyásának iránya 
* **Tegyük hozzá, hogy a folyamatábrához még tartozhatnak további jelölések - lásd a mellékletben - illetve előfordulhat hogy különböző helyeken esetleg egy-egy jelölésnek más jelentése van. Ha a hallgatók javasolnak ciklusmagra, csomópontra vonatkozó külön jelölést, lehet használni, de mindenképp érdemes még az elején tisztázni, hogy milyen eszközkészlettel akarunk dolgozni**
* **Itt érdemes a jelölések kapcsán elmondani, hogy az egyetemen alkalmazkodni kell a tanárok által megadott jelölésformákhoz. Később ezek alapos ismerete mellett majd el lehet térni, de beadandókban, számonkérésekben azt kell használni, amit tanítottak.**
* Rendezzük őket csoportokba, majd adjuk ki feladatnak, hogy próbáljanak folyamatábrával levezetni egy algoritmust, ami megoldja az előbb felírt feladatot. Próbáljuk meg úgy szervezni, hogy haladók és kezdők is legyenek egy csoportban, és hívjuk fel a figyelmet arra, hogy közösen dolgozzanak, segítsenek egymásnak!



* A feladatnak több jó megoldása is van, hiszen a feladatot nem csak egy algoritmus oldja meg. Ha nem érkezik jó megoldás, akkor rajzoljuk fel a fenti folyamatábrát (megszámolás tétel).
* **Lehet használni a ciklusmag jelölést is a megfelelő helyen!**
* A folyamatábra után megkérdezzük, mindenkinek érthető-e ez a módszer. Ha nem, akkor esetleg egyszerűbb algoritmusokat is felrajzolhatunk, hogy szemléltessük a program részeit (emberibb témán kereszül, például teafőzés algoritmusa, de ha arra van igény, részletesebben elmagyarázhatjuk a program részeit is, pl. pontosan hogy működik az elágazás).
* Miután letisztáztuk a folyamatábrát, **következzen a struktogramm**. Tudassuk velük, hogy ezt a módszert kell majd használniuk a Programozás tárgy keretein belül, így fogják leadni az anyagot és számonkérni is.
* Kérdezzük meg, van-e valaki, aki fel tud rajzolni egy struktogrammot a táblára és elmondani, hogy kell olvasni, mik a részei. Ha nincs, akkor rajzoljuk fel mi, esetleg javítsunk bele:
* Először egy teljes ábrát kellene rajzolni, hogy lássanak egyet egészben, utána jöhet a részeinek elmagyarázása.
* A struktogramm úgy néz ki, mint egy táblázat, és felülről lefelé olvassuk.
  + Utasítás 
  + Ciklus 
  + Döntési pont, elágazás 
* A folyamatábrához képest tehát az a nagy különbség, hogy itt mindig egyértelműen fentről lefelé haladunk, valamint, hogy nincs külön megjelölve az algoritmus kezdete és vége, mivel egyértelmű.
* Maradhatnak az előző csoportok. Adjuk ki feladatnak, hogy próbáljanak struktogrammal levezetni egy algoritmust, ami megoldja az előbb felírt feladatot. Hívjuk fel a figyelmet arra, hogy közösen dolgozzanak, segítsenek egymásnak!

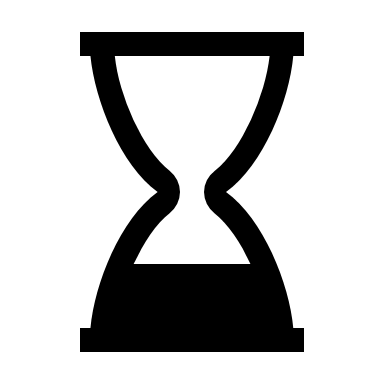


* Fontos, hogy elkezdjenek foglalkozni már az első hetekben az új fogalmakkal a gólyák, mivel ebből lehetnek kisZH-k és a nagyZH-k is! Továbbá, fontos, hogy minél előbb készség szinten tudják kezelni ezeket a fogalmakat, mert ezek szolgálnak alapot a későbbiekben, ezekre építkezik tovább a képzés.

### Kódolás, programtervezés

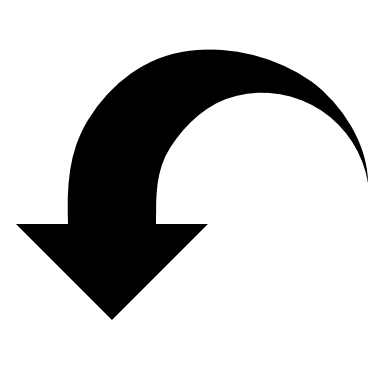
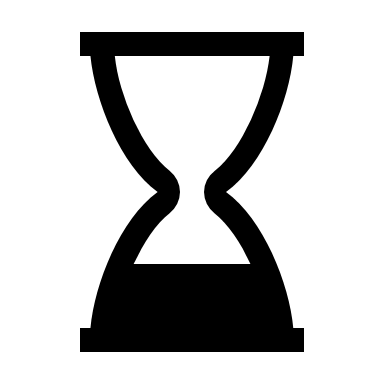
* **Kérdés:** Írt már valaki közületek **specifikációt,** **dokumentációt?**
* Fontos, hogy itt nem arról szól a programozás, hogy leültök a gép elé és elkezdetek kódolni gondolkodás nélkül.
* **Programtervezés:** A programtervező informatikusi szak arra készít fel titeket, hogy minél hatékonyabb, minél hibamentesebb programot készítsetek, valamint, hogy **minél intelligensebben álljatok hozzá a programozáshoz**. *(„Nem úgy próbálsz meg megoldani egy hibát a programban, hogy átírogatsz relációs jeleket.”)* Ehhez elengedhetetlen az előzetes tervezés, bármennyire is fölöslegesnek tűnhet itt az egyetemen, mikor még egyszerű programokkal foglalkoztok.
* **Specifikáció bevezetése:** A program meghatározása. Szerepel benne milyen be- és kimenetekkel dolgozik a program, valamint az, hogy milyen feltételek mellett. Röviden: miből mit akarunk kihozni. Azért fontos, hogy megértsük a program matematikai működését.
* **Dokumentáció bevezetése:** A program használati utasítása. Készülhet külön a felhasználók, vagy a fejlesztők számára. A fejlesztői dokumentációt írhatjuk magunknak vagy munkatársainknak.
* **Papíron programozás:** A papíron programozás alatt ne feltétlenül kódolásra gondoljatok!Kódot ritkán fogtok papíron írni, főleg specifikálás és struktogramm rajzolás lesz írásban. **Viszont állásinterjúkon előfordulhat, hogy papírra/táblára kell kódot írnod azért, hogy lássák, mennyire vagy otthon az adott nyelvben.**

### Programozási tételek

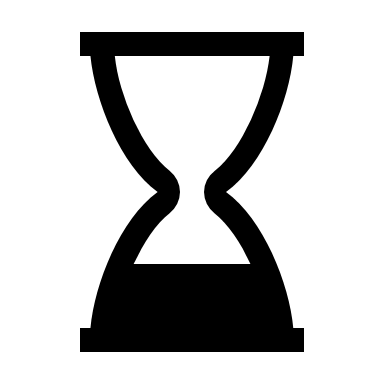
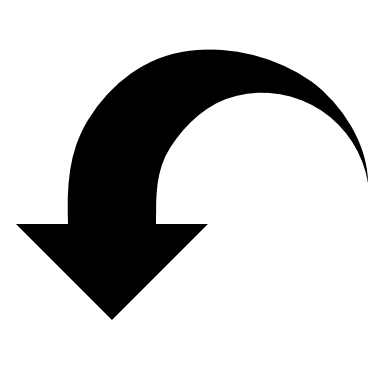
* Itt lehet kiosztani a pszeudokódos feladatot. Miután megcsinálták, meg lehet kérdezni, felismert-e valaki jellegzetes algoritmusokat? Innen át lehet vezetni a témát a tételekre. A feladat során a következőket tartsuk szem előtt:
  + Tisztázzuk az alábbi fogalmakat: tömb, modulo, tömb.elemszám
  + Az algoritmusleíró eszközök 1-töl vannak indexelve, így itt is
* Érdemes átismételni a prog tételeket, a mellékletek között megtalálható.
* **Kérdés:** Hallottatok már a **programozási tételekről?** Tudtok példát mondani?
* Ha mondanak tétel neveket, azokat írjuk fel a táblára. Hagyhatjuk őket beszélni róluk.
* **Programozási tételek bevezetése:** A programozási tételek olyan algoritmusok, amelyek bizonyos feladatok megoldására íródtak. A képzés elején programozási tételeken keresztül fogtok programozni tanulni.
* **Előnyeik, hátrányaik:** A tételek helyes működése matematikailag bizonyítható! Ez azt jelenti, hogy minden előforduló esetben jól működnek. Emellett egységesen lehet oktatni velük és számon kérni őket, amely elengedhetetlen egy ilyen nagy létszámú szakon.
* Az első félévben 6 programozási tételt fogtok tanulni, amik a következők: **összegzés, eldöntés, kiválasztás, keresés, megszámolás, maximumkiválasztás**. (A nevüket esetleg felírhatjuk a táblára.) Egy hasznos módszer a tételek megtanulására:
  + Írd le a tételt egy lapra, miközben olvasod!
  + Fordítsd meg a lapot, és próbáld meg emlékezetből leírni!
  + Ezt a két lépést ismételd és variáld addig, amíg nem vagy biztos a tudásodban!
* Fontos tehát, hogy megszokjátok, nem elég a tanuláshoz az, hogy olvasgatjátok az anyagot! Itt sok mindent szóról szóra, akár karakterhelyesen kell visszaadni majd mind a programozás, mind a matekos tárgyaknál.
* **A végére értünk a tréning azon részének, amely az egyetemi programozásoktatással foglalkozik, és most olyan skill-ekről lesz szó, amelyek segíthetik a hallgatót az egyetemi kihívások leküzdésében, illetve emelheti a programozó értékét a munkaerőpiacon, de kevésbé hangsúlyos az oktatásban.**

### Jó programozó

* Sajnos az egyetemi kurzusokon elsajátított tudás nem elég ahhoz, hogy kiemelkedő szakemberek legyetek a jövőben. Kiemelten fontos ez az informatika világában, ahol mindenféle papírral vagy anélkül is el lehet helyezkedni. Nagyon sokat számít a munkahelyi előmenetelben a társaid és feletteseid véleménye nem csak a tudásodról, hanem a személyedről, a hozzáállásodról is. Szerencsére vannak a munkahelyi sikerességnek olyan egyéb összetevői, melyeket lehet tudatosan fejleszteni!
* Mivel a programozók általában csapatokban dolgoznak, elengedhetetlen az, hogy jól tudj együttműködni és közösen dolgozni munkatársaiddal. Fontos tehát, hogy foglalkozzunk mind a programozói, mint az irodai a munkánk minőségével (irodai alatt értsd például a meetingeket, vagy egy állásinterjút). Ezek kapcsán két fogalmat érdemes megismernetek, amely a **clean code** és a **soft skills**.
* Kérdezzük meg, hallottak-e már róluk. Ha igen, akkor kérjük meg őket, hogy próbálják meg elmondani, mit jelentenek, vagy próbálják lefordítani magyarra.
* A soft skillekkel kapcsolatban érdemes a tréning előtt - ha van erre lehetőség - megkérdezni a pszichológust, hogy beszéltek-e/fognak-e beszélni róla? Ha igen, akkor max 1-2 mondatot szánjunk csak rá, de egyébként sem kell elidőzni ezen a részen.
* **Clean code:** Programtervező informatikusként nagyon fontos, hogy milyen minőségben programozol! Írhatsz olyan programkódot, amin még te sem tudsz eligazodni, vagy olyat, ami logikusan van felépítve és más fejlesztőknek is érthető. Már az egyetemen is fontos, hogy jól felépített, könnyen érthető kódot írjatok, hiszen nem mindegy, hogy megérti-e az oktató a programod.
* Most egy feladaton keresztül megpróbáljuk szemléltetni, milyen hátrányt jelent a rosszul összerakott program. Persze nem kódot fogtok kapni, hanem folyamatábrát.
* Osszuk ki a szöveget, valamint a rossz ábrát - feladat a mellékletekben. Hagyjuk, hadd olvassák el a feladatot, esetleg ismertessük szóban.
* A feladat tehát az, hogy megtaláljuk az algoritmusban, miért sikerült rosszul a gulyásleves!
* Miután egy ideje már kínlódnak vele, osszuk ki a második ábrát is, majd adjunk nekik még pár percet. A feladat összefoglalása közben adjunk körbe egy Clean Code-os lapot, hogy lássák, a kódban mire gondolunk.
* A két folyamatábra között egyértelműen látszik, melyik az olvashatóbb, értelmezhetőbb. Hasonlóképpen a programozáskor is vannak különbségek. Igaz, hogy még egyszerű programokkal fogtok foglalkozni, ahol nem olyan fontos, hogy vannak felépítve, de ha nem kezdtek el odafigyelni és igényesen kódolni, akkor mikor elmentek dolgozni, nagy hátránnyal indultok.



* Egyeztessünk a trénerrel (pszichológus), hogy volt-e már, vagy szeretne-e később ilyen feladatot az ő tréningjén. Ha igen, akkor hagyjuk meg neki, ha nem, és az idő engedi, akkor vezessük le mi.
* **Soft skills:** Amerikai eredetű gyűjtőfogalom, mely lefedi mind a munkához való hozzáállást, mind a munkahelyi szerepvállalást és kommunikációs készséget. Sajnos ez nem mindenkinek jön természetesen, de az IK-n a DTK biztosít képzéseket, amelyek segíthetnek ezek elsajátításában. Ezen és ilyen jellegű képzések mellett/helyett persze önállóan is képezhetjük magunkat, de a valódi fejlődés gyakorlatban történik.
* Rajzoljunk egy ember figurát a táblára, ami a programozónkat szimbolizálja, és írjunk mellé olyan tulajdonságokat, amiket a gólyák mondanak. A figurát ne töröljük le, később még használva lesz.



* Ha nincs idő a soft skilles részre:
* A soft skills kifejezéssel találkoztatok, vagy találkoztok majd a kurzus keretein belül. Mi viszont szerettük volna arra felhívni a figyelmeteket, hogy ez nem csak az egyetemi tanulmányaitokra van hatással, hanem az egész karrieretekre.
* Fontos tehát, hogy ELTE-s programtervező informatikusként arra törekedjetek, hogy minél jobb pozícióban tudjatok elhelyezkedni, amihez elengedhetetlen az, hogy kitűnjetek a tömegből. *(Nem mindegy, hogy napi 8 órában kódolsz, vagy egy céget vezetsz.)* Ehhez nagyon jó alapot nyújt az ELTE-s diploma, de hogy még inkább profitálni tudj az itt eltöltött évekből, muszáj a szakmai fejlődésen kívül másra is fordítani időt.

## Befejezés

* Ezzel a tréning végére értünk, reméljük, hasznotokra vált.
* Ha kérdésetek van a tréninggel, vagy bármi mással kapcsolatban, nyugodtan keressetek meg minket személyesen, vagy írjatok az EIK csoport e-mail címére: [eik@inf.elte.hu](mailto:eik@inf.elte.hu)
* Tegyünk EIK szórólapot a (tanári?) asztalra.
* Találkozni fogunk még egyszer, az Egyetemi Alapozó és Tanulásmódszertan kurzus 2. szakaszában - szeptember 20 és október 4 - amikor matek tréninget fogunk tartani.
* Természetesen, ha ott helyben kérdeznek, vagy szeretnének valamiről beszélni, akkor ne hagyjuk ott őket, de az idősávból ne csússzunk ki!
* A tárgyakkal kapcsolatos kérdésekhez hivatalos információkat a mellékletek között lehet megtalálni
* Elsőéves tárgyak:  
  Számítógépes rendszerek  
  Programozás  
  Imperatív programozás  
  Funkcionális programozás  
  Matematikai alapok  
  Innovatív vállalkozás management - nem kötelező az első félévben  
  Jogi ismeretek - nem kötelező az első félévben
* Szakirányok:  
  Modellező  
  Szoftvertervező  
  Szoftverfejlesztő