

Debreceni Egyetem, Informatika Kar
Számítógéptudományi Tanszék

Algoritmizálási megközelítések a 10. osztályban



Dr. Csernoch Mária

egyetemi docens

Korponai Réka

V. éves Informatika–Matema-
tika tanárszakos hallgató

Mező-Nagy Dóra Hanna

V. éves Biológia–Informatika
tanárszakos hallgató

Debrecen, 2024

Bírálatra való beküldéséhez hozzájárulok:

Tartalomjegyzék

1. Bevezetés	4
1.1. Célkitűzések.....	7
1.2. Egyetemi képzés	8
1.3. Vizsgálati módszerek	9
2. Adatkezelés „új” szintre emelve	10
2.1. A hatékony táblázatkezelés és a programozás kreatív eszköze	10
2.2. Sprego módszer.....	10
2.3. Miért alkalmazzuk a Spregot?	11
3. Programozás oktatása: Melyik nyelvet válasszuk?.....	18
3.1. C programozási nyelv	18
3.2. Python ütközői	19
4. Tankönyvi feladatok megoldása és tesztelése	20
4.1. Középiskolások visszajelzése: kérdőív alapú kutatás.....	20
4.2. Algoritmizálási megközelítések a feladatok megoldásában	30
5. Önálló munka részletezése	46
6. Konklúzió	46
7. Köszönetnyilvánítás.....	48

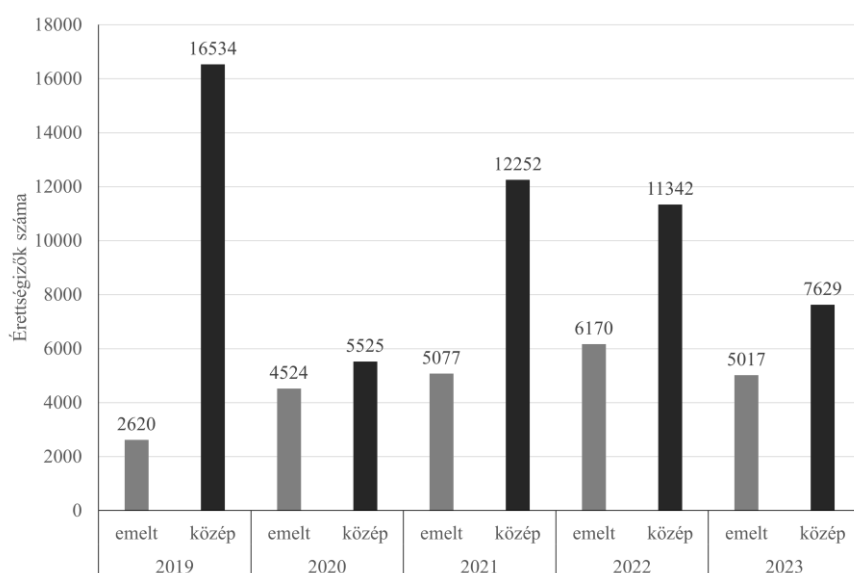
8. Melléklet.....	48
8.1. Kérdőív	48
8.2. Excel függvények megjelenése osztályonként.....	49
9. Irodalomjegyzék	50

1. Bevezetés

Az alkalmazói, felhasználói adatkezelés és -feldolgozás folyamatosan keresi helyét az informatikában. Egyrészt, ezek a tevékenységekhez elengedhetetlenek a mindennapi életben, de a legtöbb ember számára még mindig kihívást jelent ezen tevékenységek azonosítása, informatikához köthető kapcsolatának meghatározása és a hozzájuk köthető alkalmazások hatékony használata [4] [41] [49] [48] [61] [78] [79] [50] [51]. Más részből, annak ellenére történik mindez, hogy az ország többsége, beleértve Magyarországot is, lehetőséget biztosít iskolai keretek között informatika (computer science, CS) tanulására [85] [44] [21]. Korábbi kutatások azt mutatják, hogy ebben az játszik szerepet, hogy az emberek többségénél nem alakul ki a számítógépes gondolkodás, mint alapkészség és hiányoznak az erre épülő informatikai alapismereteket. Az informatika/digitális kultúra oktatása nagyban befolyásolja ezt a tendenciát. Egyik okának tekinthetjük, hogy a szoftvereket eszközcentrikusan alkalmazzák és oktatják. Vizsgálataink során tapasztaltuk, hogy a digitális kultúra tankönyvek gyakran csak a szoftverekre összpontosítanak, túlzott hangsúlyt fektetve az eszközökre. A legtöbb tankönyvi feladat és a hozzájuk tartozó megoldások felületi szintűek, és alig térnek ki az algoritmikus megközelítésre a háttérben meghúzódó informatikai alapokra. Tanárszakos hallgatóként, hamar felismertük, hogy a digitális kultúra tantárgy oktatásában nagy problémát jelent, az is, hogy nem tanítják meg az informatika egyes részterületei közötti átjárhatóságot, hiszen az alkalmazói feladatokra és problémákra felépített algoritmusok könnyen átvihetők más területekre, mint például az adatbázis-kezelés vagy programozás környezetbe. Különböző szoftvereket tanítunk, ugyanazokkal az algoritmusokkal, mégsem jelenik meg tanórákon a kapcsolat, az ezeket összekötő informatikai háttér (CS). Hospitálásaink során észrevettük, hogy egy tanórán gyakran csak egyetlen szoftvert használnak a diákok, miközben az adott feladat megoldásához más szoftverek is szükségesek lehetnének. Újabb problémaként jelenik meg, hogy az alkalmazó szoftverek és az ezekhez köthető adatfeldolgozást nem tekintik az informatika részének, hanem leminősítik digitális írástudásra [85] [44] [21] [3], szétválasztják az informatikát (CS) és a nem professzionális informatikát, nincs lehetőség az informatika megértésére és elfogadására mint tudomány [30]. Egy újabb kihívást jelent, hogy túl sokat akarunk tanítani nagyon rövid idő alatt [38]. Az óraszámok és a tananyag mennyisége nem arányos egymással, ami megnehezíti a tanárok részéről a tananyag hatékony átadását és a diákok részéről a megértését. Ez a tananyag-sűrítés gyakran arra kényszeríti a tanárokat, hogy csak felületesen vagy egyáltalán ne érintsenek

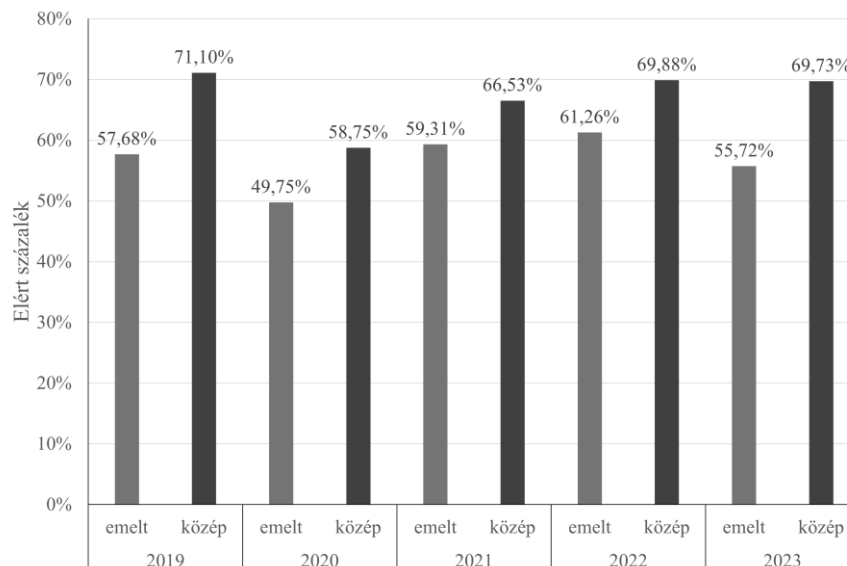
CS témákat, így nem enged meg elegendő időt a mélyebb megértésre és gyakorlásra. Arra viszont elegendő, hogy egy-egy eszközcentrikus vizsgát sikeresen letegyenek a tanulók, így szinte senki nem érzi a megértés hiányát, valamint a folyamatok alacsony hatékonyságát.

Emiatt a diákok nem biztosak tudásukban így a középiskolai tanulmányaik végén nem jelentkeznek érettségi vizsgára, ez az elmúlt 5 évben jelentkezők számából is megfigyelhető. 2022-ben és 2023-ban még lehetőségük volt a 2012-es NAT szerint tanuló diákoknak informatika érettségi vizsgát választani digitális kultúra helyett, miközben az új rendszerben tanulóknak már csak az utóbb említettre volt lehetőségük jelentkezni [83] [84].



1. ábra 2019–2023 informatika érettségi vizsgát tevők száma

Már a jelentkezők számából is látszik, hogy az évek során egyre kevesebb tanuló kíván érettségi vizsgát tenni ebből a tantárgyból, hiszen tudásuk sokszor nem elegendő a megfelelő eredményhez, mely a továbbtanuláshoz sokszor elengedhetetlen.



2. ábra 2019–2023 informatika érettségi vizsgán elért átlageredmények százalékban

Mind a középszintű, mind az emelt szintű érettségi vizsgák átlageredménye azt mutatja, hogy az eszközcentrikus oktatás nem motiválja a diákokat a tanulásra, és a tanórai keretek között nem sajátítják el a számítógépes gondolkodás képességét sem. Hiszen a kiadott feladatok során a megfelelő ismeretek birtokában magasabb eredmény is elérhető lenne. Sokszor azok a tanulók jelentkeznek középszinten erre, akik nem tudják még biztosra, hogy milyen irányban szeretnék folytatni tanulmányaikat. Digitális kultúra tantárgyból 2023 tavaszi érettségi időszakában a fent említett informatika érettségivel párhuzamosan jelentkezhettek a diákok, mivel 11. osztályban kötelező tárgyként befejeződik a digitális kultúra oktatása. Emelt szinten 71, míg középszinten 2100 diák jelentkezett. A 2023–2024-es tanévben már csak digitális kultúrából van lehetőségük a tanulóknak vizsgázni és a kiadott információk alapján középszinten 6601-en, míg emelt szintű vizsgára 3260-an jelentkeztek. Ennek oka a megváltozott, túl sok követelmény, illetve, hogy a tanulók nem sajátítanak el megfelelő tudást az évek során.

Hiányzik az informatikatanárok és az informatikát oktatók megfelelő felkészítése [51]. Ez a hiányosság jelentős kihívást jelent a digitális kultúra oktatásában, mivel a gyorsan változó technológiai környezetben az oktatóknak lépést kell tartaniuk a legfrissebb fejleményekkel és módszerekkel és ki kell tudják választani azokat az eszközöket és módszereket, amelyekkel hatékonyan megvalósítható a tudásátadás és -támogatás. Ugyanakkor az informatikai alapok és a számítógépes gondolkodás hiánya további akadályt jelent, hiszen nincs megfelelő alap az új ismeretek építéséhez. A hiányzó vagy nem megfelelő továbbképzések korlátozhatják az oktatók

képességeit és hatékonyságát az új technológiák és pedagógiai módszerek bevezetésében, aminek következtében a tanítás minősége és hatékonysága is csökkenhet.

Szeretnénk ezekre a problémákra megoldást nyújtani. Célunk egy hatékonyabb adatfeldolgozás- és programozásorientált módszer bemutatása, amely javítja a tanulók problémamegoldó képességét, hatékonyan fejleszti a számítógépes gondolkodását és összekapcsolja a hagyományos programozásorientált képzést az alkalmazói ismeretek tanításával [76] [56] [56] [28].

Kutatásunk ötletét a Debreceni Egyetemen megismert Sprego (Spreadsheet Lego), azaz a táblázatkezelői programozás, valamint az új digitális kultúra tantárgy tantárgyi követelményei és a hozzá tartozó tankönyvek inspirálták [11] [9] [10] [6] [7] [8] [12] [13]. Dolgozatunkban bemutatunk egy módszert, amit a hatékony adatkezelés kreatív eszközeként jellemezhetünk táblázatkezelői, adatbázis-kezelői és programozási környezetekben. A módszer újdonsága a Sprego bevezetésében rejlik, ami egy táblázatkezelő környezetben alkalmazott algoritmikus megközelítés [26] [28] [25]. Kutatásunkban kiemeltünk két nagy témakört, a *Táblázatkezelés*, illetve az *Algoritmizálást és programnyelv használata* című fejezetet a 10. osztályos *Digitális kultúra* tankönyvből [10].

1.1. Célkitűzések

A kutatásunk célját az alábbiakban foglaljuk össze.

- Megvizsgáljuk, hogy a jelenleg érvényben lévő digitális kultúra tantárgy hogyan fejleszti a tanulók adatkezelési, táblázatkezelési, programozási ismereteit, algoritmikus és problémamegoldási készségét és képességét.
- Felmérjük és feltárjuk azokat a rejtett hibákat, amelyek az alacsony hatékonyságú programozásoktatáshoz vezettek az informatika és a digitális kultúra oktatása során.
- Megfogalmazunk olyan javaslatokat, amelyek más programozási paradigmák és nyelvek használatára ösztönöznek, és segítenek a programozás tanításának bevezetésében, valamint ennek hatékonyságnövelésében, bevonva más tantárgyakat is a fejlesztési folyamatba, kihasználva a tantárgyi integráció nyújtotta lehetőségeket.

Egy olyan megközelítést, módszert szeretnénk bemutatni, mellyel hatékonyabbá tehetjük a digitális kultúra oktatását, a számítógépes gondolkodás fejlesztését, felkészítve a tanulókat a

jövő munkáltatói által elvárt készségekre, képességekre, kompetenciákra [20]. Dolgozatunkban prezentáljuk a táblázatkezeléshez és az egyszerűsített funkcionális programozáshoz köthető Sprego megközelítést. Ez a programozási eszköz nem csupán a táblázatkezelés és adatfeldolgozás alapjainak elsajátítását támogatja, hanem alkalmas kezdő programozás tanítására is [26] [29] [25]. Emellett hatékonyan alkalmazható más tantárgyak tartalmának feldolgozására [10]. Kiemelnénk, hogy az informatika oktatásnak elsősorban az algoritmizáló és problémamegoldó [42] képességek kialakítását és fejlesztését, és ezen keresztül a CS alapjainak átadását [53] kellene szolgálnia, nem csupán egy programozási nyelv szintaktikájának, fejlesztő környezetének elsajátítását. A fejlesztéseket nemcsak a tankönyvben tárgyalt Python programozással, hanem a kutatásunkban bemutatott Sprego módszerrel is el lehet érni. Egy olyan felhasználóbarát környezetben, amelyet nemcsak a diákok, de a felhasználók többsége is ismer, és a mindennapi életük során alkalmaz [4] [64] [77] [62] [19] [18] [20]. Korábbi kutatások már bizonyították, hogy a Sprego koncepciója sokkal jobban illeszkedik a tanulók háttérismeretéhez [28] [25] és átfogó képet ad az adatfeldolgozás és algoritmusok tervezéséhez, továbbá támogatja a valódi tartalomfeldolgozást.

Kutatásunkban bemutatjuk, hogy a 10. osztályos tankönyv *Algoritmizálás és programnyelv használata* című fejezetében szereplő feladatok jelentős részét Sprego módszerrel hatékonyan megoldhatjuk. Ezzel a módszerrel ugyanazt az algoritmust alkalmazzuk, mint Python programozással, viszont egyértelműbb és a tanulók számára könnyebben érthető kódokat használunk, ami csökkenti a kódolási terhet, ezzel a kivitelezést hatékonyabbá teszi.

A dolgozatban kitérünk arra is, hogy a tankönyvi feladatok nem biztosítanak elegendő motivációt és gyakorlási teret a tanulók számára és nem megfelelő módon támogatják az algoritmizáló képesség fejlesztését.

1.2. Egyetemi képzés

Képzésünket 2019 szeptemberében kezdtük a Debreceni Egyetemen. Az osztatlan tanárképzés három különböző gyakorlatból áll. Az első gyakorlat a közösségi pedagógiai gyakorlat, amely a tanórán kívüli, szabadidős tevékenységek szervezésével, programkészítésével és közösségépítésével foglalkozik [60] [67]. A második gyakorlat a B típusú gyakorlat, amely középiskolai tanár szakosodásunk miatt csak a képzés 5. évében (9-10. félévben) kezdhető meg. Ezen a gyakorlaton kezdünk el foglalkozni gyerekek oktatásával. Egy iskolában vezetőtanár

irányítása alatt végzett csoportos pedagógiai és önálló tanítási gyakorlat, amelynek keretében megfigyeljük és elemezzük a tanári munkát szakos órákon, osztályfőnöki órákon és nem szakos órákon [60] [65]. Az utolsó gyakorlat pedig az összefüggő gyakorlat, melyre a 6. évben (11–12. félévben) kerül sor, ami a képzés során szerzett elméleti ismeretekre és gyakorlati tapasztalatokra épül. A gyakorlat egy köznevelési intézményben zajlik, és folyamatos irányítás mellett végezzük, ahol mentorok, konzulensek és felsőoktatási tanárképző szakemberek segítenek és tanácsot adnak [60] [66].

A jelenlegi dolgozatunkban a közösségi pedagógiai gyakorlat és a fél éves B típusú gyakorlat tapasztalatait tudjuk felhasználni, mivel az iskolába, osztályba történő kutatás és hipotézis kipróbálása a gyakorlatok hiányában, még nem valósulhatott meg. Emiatt más eszközöket használunk a kutatásunk alátámasztására.

1.3. Vizsgálati módszerek

A dolgozatunkban elméleti alapokkal igyekszünk alátámasztani kutatásunkat, felhasználva releváns szakirodalmi forrásokat és korábbi kutatásokat, amelyek a témához kapcsolódnak. Emellett megbízható forrásból származó statisztikai adatokat és közzétett információkat fogunk használni, amelyek támogatják a kutatásunk eredményeit.

Kutatásunkhoz feladatokat válogattunk a 10. osztályos Digitális kultúra tankönyvből [10]. Minden feladatot két programnyelven oldottunk meg, az Excel és a Python programozási környezetekben. A feladatok kiválasztása és megoldása során fokozott figyelmet fordítottunk a feladatok szövegére, a tantárgyi kapcsolatokra, valamint arra, hogy hány feladatban volt szükség algoritmus tervezésére. Elemeztük, hogy egy-egy feladat, szövege és tartalma alapján mennyire motiválja a tanulókat a megoldásra, a programozástanulásra, mennyire kelti fel az érdeklődésüket a probléma. A kutatáshoz továbbá összeállítottunk egy kérdőívet, amely középiskolás diákok algoritmizálási készségét méri, és a feladatokkal kapcsolatos véleményüket kéri. Ez a kérdőív segíti a középiskolás tanulók programozási ismereteinek feltérképezését, valamint annak feltárását, hogy az algoritmizálási készségeiket hogyan tudják a tantárgyi integrációban kamatoztatni.

A felsőoktatási tantervi háló nem teszi lehetővé, hogy kipróbáljuk az új módszert tantermi keretek között. A gyakorlótanítások, hospitálások, pedagógusok véleménye, és a kutatáshoz

készített teszt azonban lehetőséget nyújt a hipotézisek igazolására. Ezen tapasztalatok és mérési eredmények alapján érvelhetünk és megalapozhatjuk a kutatásunk eredményeit, még akkor is, ha a gyakorlati tesztelés még nem történhetett meg.

2. Adatkezelés „új” szintre emelve

2.1. A hatékony táblázatkezelés és a programozás kreatív eszköze

A Sprego egy összetett név, amely a Spreadsheet Lego kifejezés rövidítése. Ebben a névben két fő jelentés rejlik, amelyek jól összefoglalják a megközelítés lényegét és céljait.

1. Spreadsheet: Táblázatkezelés, mely arra utal, hogy az alkalmazás egy adatfeldolgozó és táblázatkezelő rendszeren alapul. Ez a rész hangsúlyozza az alkalmazás fő funkcióját, miszerint lehetőséget nyújt valós adatok feldolgozásához alkalmazói keretek között.
2. Lego: Másik jellemzőjére utal. Ahogyan a Lego kockákból építhetünk különböző összetett formákat és szerkezeteket, úgy a Spregoban is alapelemekből építhetünk fel összetett műveleteket és algoritmusokat. Ezek az építőkockák a függvények, beépített függvények, amelyeket a táblázatkezelő rendszerekben megtalálhatunk [26].

2.2. Sprego módszer

A Sprego módszer 12 függvény alkalmazásával vagy ezek kompozíciójával lehetővé teszi az alapvető programozási problémák, tételek hatékony megoldását [38] [81]. A Sprego függvényeket három kategóriába soroljuk: Szöveg, Szám és Pro (Professzionális/Programozói) (1. táblázat).

1. táblázat. Sprego függvények.

Szöveg	Szám	Pro
JOBB()	MIN()	HA()
BAL()	MAX()	HIBÁS()
HOSSZ()	SZUM()	HOL.VAN()
SZÖVEG.KERES()	ÁTLAG()	INDEX()

A Szöveg csoport a szöveges adatokat kezelő függvényeket, míg a Szám csoport a számértékekkel foglalkozó függvényeket tartalmazza. A Pro kategória széles spektrumot foglal magában, így lehetővé téve a felhasználók számára, hogy a függvényeket professzionális táblázatkezelőként vagy éppen kezdő programozóként értelmezzék és alkalmazzák.

Tehát, egy tucat általános célú függvény használata, lehetővé teszi sok különféle probléma megoldását. Azonban időnként előfordulhat, hogy egy speciálisabb vagy összetettebb feladat megoldásához szükségünk van olyan függvényekre, amelyek nem szerepelnek az alapfüggvények között. Ilyenkor a Sprego függvénykészletét óvatosan és jól átgondoltan lehet bővíteni kiegészítő függvényekkel (2. táblázat). Az elsődleges szempont a bővítés során az, hogy olyan függvényeket válasszunk, amelyek valóban nem helyettesíthetők Sprego függvénnyel, vagy csak nagyon nehezen és körülményesen oldhatók meg azokkal. Fontos, hogy az új függvények valóban új és hasznos funkcionalitást adjanak az algoritmikus megközelítéshez [26].

2. táblázat. Sprego kiegészítő függvények egy lehetséges csoportja.

SOR()	ÉS()	ELTOLÁS()
OSZLOP()	VAGY()	KEREKÍTÉS()
KICSI()	NEM()	HELYETTE()
NAGY()	VÉL()	INT()

2.3. Miért alkalmazzuk a Spregot?

2.3.1. Grafikus felület és verziófüggetlenség

A Sprego függvényei teljes mértékben kompatibilisek a Microsoft Excel, az OpenOffice és az online táblázatkezelő szoftverekkel, és ez az egyezőség kiterjed a különböző verziókra is. Azaz a szoftver fejlesztése vagy frissítése közben is bizalommal alkalmazhatjuk ezeket a függvényeket, probléma vagy hátrány nélkül [26]. A legújabb Excel verziókban megjelent dinamikus tömbképletek [57] esetében fontos kiemelni, hogy teljes kompatibilitást mutatnak visszafelé is a Sprego részeként tanított és alkalmazott tömbképletekkel [64] [62], ami egy további bizonyíték a Sprego programozás- és felhasználóorientált módszereinek hatékonyságára.

2.3.2. Irodai szoftverek használata különböző mathability szinteken

Mindennapi életünk során rendszeresen találkozunk olyan kihívásokkal, adatfeldolgozási problémákkal, amelyek megoldásához különböző szoftverek segítségét kell igénybe vennünk. Legyen szó munkáról, tanulásról vagy akár személyes ügyek intézéséről. Szinte minden munkakör esetében alapvető elvárás az alkalmazói szoftverek ismerete és használata. Napjainkban széleskörben elterjedt szoftvercsomagok felhasználóbarát felületet biztosítanak. Például az újabb táblázatkezelői szoftververziók folyamatosan bővülnek extra funkciókkal, melyek mindössze néhány kattintással elérhetőek. A hazai informatikaoktatás is erre a „kattingatásos” módszerre, szoftverhasználatra összpontosít, az algoritmusok építése és a számítógépes gondolkodás fejlesztése helyett. Véleményünk szerint, a táblázatkezelésoktatás fő célja – hasonlóan más informatika témakörökhöz –, hogy tanulóinkat megtanítsuk valós adatok feldolgozására és a számítógépes gondolkodás, ezen belül is kiemelten a problémamegoldó gondolkodási képesség tudatos fejlesztésére [80] [42] [53] [31]. A szándékunk az, hogy ne csak tetszőlegesen, cél nélkül kattintgassanak a tanulók egy felületen, hanem tudatosan, hatékonyan és eredményesen dolgozzanak a táblázatkezelői felületen.

Általánosságban elmondható, hogy a táblázatkezelés lehet alacsony vagy magas mathability szintű¹, attól függően, hogy melyik problémamegoldási megközelítés dominál. A hazai oktatásban használt eszközcentrikus megoldások az alacsony mathability szintű megközelítések közé sorolhatók, míg a Sprego programozás a magas mathability szintű megközelítést alkalmazza. Ez a szemlélet hangsúlyozza a problémamegoldás fontosságát anélkül, hogy túlzott hangsúlyt fektetne az eszközökre [30] [43] [40] [15] [16] [22] [23] [31].

Az informatikaoktatásában másik jelentős probléma, hogy az informatikai (CS) témakörök közötti átjárhatóság és összefüggések nem jelennek meg a tanórákon. A Sprego erre a problémára is megoldást nyújt, hiszen az autentikus táblázatok tanórai felhasználásának előnye, hogy ezen táblázatok használatával lehetőség nyílik intenzív fájlkezelésre, fájlkonverzióra, az alkalmazások közötti átjárhatóság megismertetésére, gyakorlására valódi adatokon [29].

¹ A táblázatkezelés mathability szintje azon alapul, hogy mennyire támogatja a problémamegoldást, szemben a minél több beépített függvény és az eszköz bevezetésével és halmozásával.

A módszert alkalmazhatjuk az adatbázis-kezelés előkészítésére is. Az autentikus táblázatok használatával lehetőség nyílik az adatok kezelésének gyakorlására, és példákat mutathatunk a helyes és helytelen adatbázis tervezésre, illetve ezen utóbbiak javítására. Ez segíthet a diákoknak megérteni az adatbázis-kezelést és a helyes adatbázis tervezés fontosságát. Megnézünk egy-egy feladatot ugyanazzal az adathalmazzal (3. ábra), amit először a Sprego módszerrel oldunk meg, majd pedig lekérdezéssel. Ugyanazt a feladatot azonos adatokkal megoldani mind a Sprego módszerrel (1. képlet), mind pedig lekérdezéssel (2. képlet) elősegíti a szoftverek közötti átjárhatóság megértését és a tanulók motiválását. Azért, mert a diákok számára könnyebb egy már ismert adatforrást és algoritmust egy új környezetben alkalmazni, mint egy teljesen új feladatot megoldani. Ez a gyakorlat lehetővé teszi, hogy a diákok lássák, hogyan lehet ugyanazokat az algoritmusokat és problémákat különböző szoftvereken sikeresen megvalósítani, segít dönteni továbbá, hogy melyik platformot válasszák egy adott probléma hatékony megoldására.

Hány ország neve kezdődik L betűvel?

	A	B	C	D	E	F	G
1	ország	foldresz	fovaros	terulet	fo_ezer_02		
2	Afghanistan	Asia	Kabul	647500	27756		L
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		9 ország neve kezdődik L betűvel.
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	Amerika	The Valley	102	12		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

3. ábra Autentikus táblázat a Föld európai országairól

= {SZUM(HA(BAL(A2:A46)=G2;1))}

=G3 & " ország neve kezdődik "&G2&" betűvel."

1. képlet Táblázatkezelői megoldás programozási eszközökkel: tömbképlet, függvényhívások, változók, szöveg konstansok és operátorok

SELECT count(left(ország,1)) as Lbetu

FROM Fold

where left(ország,1)="L";

2. képlet Adatbázis-kezelői megoldás

Az eddigi elemzések és minták igazolják, hogy a Sprego módszert használhatjuk bevezető programozás tanítására is. A kérdőív és a dolgozat többi részében olyan általunk kidolgozott megoldások kerülnek bemutatásra, amelyek megerősítik, hogy a Sprego módszer sikeresen alkalmazható bevezető programozási nyelvként.

2.3.1. Az autentikus táblázatok az iskolai oktatásban

Az iskolai oktatásban kiemelkedő problémát jelent, hogy az iskolai tananyag és a munkahelyi elvárások közötti szakadék egyre mélyül [32] [3]. A diákok olyan adatokkal dolgoznak, amelyek nem tükrözik sem az iskolai, sem a valós világ követelményeit. Ennek következtében kialakul a diákok érdektelensége az iskolai informatika és a computer science (CS) iránt, hiszen nem látják, hogy az informatika órai példákat, leírásokat, ismertetőket hogyan lehetne hatékonyan alkalmazni a mindennapokban [56] [29].

Erre a problémára nyújt megoldást a Sprego használata, hiszen a tanórákon már az elejétől fogva valós adatokat tartalmazó táblázatokkal dolgozhatnak a diákok. Ennek a módszernek a pozitívuma, hogy olyan táblázatokkal használhatunk adatkezeléshez, amelyek az adott korcsoportot érdeklik. Megfelelő táblázatokkal a Sprego lehetőséget kínál más tantárgyak tartalmának közvetlen összekapcsolására is. Ez az oktatási megközelítés elsősorban a digitális kompetenciák fejlesztésére összpontosít, mely illeszkedik a TPACK (Technological Pedagogical Content Knowledge) által definiált követelményekhez és célokhoz [20] [18] [59] [29].

A módszerben alkalmazott autentikus táblázatokkal elkerülhetjük a hagyományos gépelős és értelmetlen feladatokat. A gépelési gyakorlatok sok időt igényelnek, amely az órákon rendelkezésre álló időt csökkentheti a mélyebb tanulásra és a gyakorlati alkalmazásra. Emellett a túlzott gépeléssel a tanulók motivációja is csökken [35] [29] [59], ugyanakkor tudjuk, hogy az eredményes tanuláshoz nélkülözhetetlen a megfelelő motiváció [35].

2.3.2. Tantárgyak közötti átjárhatóság Spregoval

A Sprego módszerben az autentikus táblázatok (2.3.1.) kulcsfontosságú elemek, hiszen megfelelő táblával megfelelő tantárgyközi kapcsolatokat hozhatunk létre. Ezek a táblázatok lehetővé teszik a diákok számára, hogy felismerjék az összefüggéseket [34], megértsék, hogyan lehet egy adott feladatot digitális úton megközelíteni, és kapcsolják össze az információkat a

különböző tantárgyak között. Ennek az eljárásnak köszönhetően a diákoknak szükségük van az adott tantárgyakban megszerzett ismeretekre, valamint az algoritmizálási készségekre annak érdekében, hogy sikeresen megoldják az előttük álló feladatokat. Az autentikus táblázatok alkalmazásával, legyen szó magyar nyelv és irodalomról, angol nyelvről, matematikáról, természetismetről, földrajzról stb., segíthetjük a diákokat az interdiszciplináris kapcsolatok felismerésében és a tanult anyagok megértésében. A Sprego módszerrel történő munka lehetőséget teremt arra, hogy elősegítsük a diákok digitális kompetenciáinak fejlődését, ami a 2020-as Nemzeti Alaptanterv irányelveivel összhangban áll [58].

A Sprego hidakat épít az informatika (CS) és a matematika között is, hiszen a függvény fogalmát helyezi középpontba. Ez a módszer segít hatékonyabban megérteni és elsajátítani a matematikai fogalmakat. Bár a matematika oktatás során már az alsó tagozattól kezdve találkozunk függvényekkel, diszkrét függvényekkel, hozzárendelések megfogalmazásával és alkalmazásával, a függvények mélyebb megértése továbbra is kihívást jelent gimnáziumi szinten, akár egyetemi szinten is.

A függvény fogalmának teljes körű megértése és elsajátítása kulcsfontosságú a programozási ismeretek fejlesztése során. Gondoljuk csak át, hogyan tudnánk hatékonyan hívni és definiálni függvényeket, ha magát a fogalmat sem értjük meg. Tehát amikor valaki mélyebben megérti a függvények fogalmát, hatékonyabban tudja alkalmazni azokat a kódolási gyakorlatban. A függvény fogalom megértését és későbbi alkalmazását a programozásban is elősegíti a Sprego.

Az összetett függvények használata elengedhetetlen a problémamegoldásban. A Sprego módszerrel a tanulók számára átláthatóbbá válik az összetett függvények kezelése és könnyebben elsajátíthatják ezt a koncepciót.

Az emberek többsége, ha függvényábrázolásról van szó, leggyakran a Geogebra szoftvert említi [36]. Sokan azonban nincsenek tisztában azzal, hogy Excelben is lehetőség van függvények ábrázolására, tehát a Sprego megközelítés is alkalmazható a matematikai függvények gépi megjelenítéséhez. A képletek segítségével részletesen vizsgálhatjuk a függvényeket. Az Excelben ábrázolt függvények és alkalmazott képletek segítségével ismét kihasználhatunk két tantárgy témaköreire vonatkozó ismereteinket, fejleszthetjük a matematikai gondolkodásunkat, miközben az algoritmizálási ismereteinkre támaszkodunk.

A Sprego módszerrel matematikai témakörök, például algebrai kifejezések, egyenletek, egyenlőtlenségek, geometriai problémák, statisztika és valószínűségszámítás, valamint matematikai modellezés is hatékonyan taníthatók. Az eszköz segíti a diákokat abban, hogy ne csak az egyes témaköröket értsék meg izoláltan, hanem lássák a két tantárgy közötti kapcsolatokat, kihasználják a számítógép nyújtotta előnyöket. A matematika tanításában fontos szempont, hogy a diákok ne csak a számok és formulák memorizálására koncentráljanak, hanem képesek legyenek azokat problémamegoldásra és valós élethelyzetek megértésére is alkalmazni. A Sprego módszer erre lehetőséget teremt, hiszen a táblázatkezelés-alapú programozás révén a diákok a matematikai koncepciók alkalmazásával valódi problémák megoldásába is bekapcsolódhatnak. Ezenkívül, a módszer a matematika és a programozás integrációjával elősegíti a diákok számítógépes gondolkodás készségeinek fejlődését és a problémamegoldó gondolkodásuk erősítését, ami kulcsfontosságú kompetencia a digitális korban.

2.3.3. Funkcionális programozás alkalmazása

Sprego módszer egy egyszerűsített funkcionális programnyelvet alkalmaz, melynek kifejezőmódja szoros kapcsolatban áll a matematika nyelvével. A matematikai formulák teljes egészében átírhatók funkcionális programnyelvi elemekre [45]. Fontos felismernünk továbbá, hogy a programozásban is alkalmaznunk kell Pólya feladatmegoldási módszerét [34], tehát az első lépés nem a fejlesztőeszköz elindításával és a kód megírásával kezdődik, hanem a probléma teljes megértésével. Booth mérései tovább erősítik, hogy bevezető nyelvként a funkcionális programozási nyelvek hatékonyabbak, mint a deklaratív nyelvek, mivel a funkcionális programnyelv használatakor a problémamegoldás folyamatában a hangsúly a probléma és a koncepció mélyebb megértésén van. Így a funkcionális programnyelvek minimális ismeretek alkalmazásával is hatékonyak a programozásoktatásban [17].

A Sprego kombinálja a funkcionális programozást és az Excel egyik alapvető eszközét, a tömbképletet [62] [64], amely flexibilis megfelelője beépítésre került az újabb Excel verziókba dinamikus képlet néven [57]. A tömbképletek használata lehetővé teszi a képletek másolásának kiváltását, mely az egyik leggyakoribb hibaforrása a táblázatkezelői dokumentumokban [77] [4]. Ezenkívül bevezetjük a tömb és a vektor fogalmát, implicit kódolás nélkül, a grafikus felület eszközeivel. A vektorok segítségével bevezetjük az index és az előre meghatározott

lépésszámú ciklus fogalmát, melyeknek az elsajátítása előkészíti az utat a deklaratív programozási nyelvek oktatásához [29].

2.3.4. Hogyan alkalmazzuk a Spregot?

A Sprego környezetben a problémamegoldás lépései a Pólya György nevéhez köthető koncepcióalapú megközelítést követik [34], amelyet a matematika területén már ismerhetünk. Gyakorlatilag egy általános útmutatót biztosít matematikai feladatok megoldásához, de ugyanígy alkalmazható más tudományterületeken is. Az alábbiakban összefoglaljuk ezeket a lépéseket az Excel programozásban.

1. Értsük meg a feladatot.

Az első lépés a probléma részletes megértése és a kapcsolódó adatok értelmezése, elemzése, áttekintése, szükség esetén táblázatos formára rendezése. Ennek során felmérjük a feladat jellegét, kihívásait és az elvárt eredményt.

2. Készítsünk tervet a megoldáshoz.

A második lépésben tervezünk egy algoritmust, amely lépésről lépésre leírja a probléma megoldását. Itt gondolkodunk a lehetséges megközelítésekben és műveletekben, majd strukturáltan összeállítjuk az algoritmust.

3. Hajtsuk végre a tervet.

A következő lépés a tervezett algoritmus lefordítása Sprego kódra. Képleteket és műveleteket fejezünk ki a megfelelő függvényekkel és operátorokkal, majd futtatjuk a programot a táblázatban szereplő adatokkal. Az összetett függvények létrehozása esetén kiemelten fontos a lépésenként végrehajtás, amely már magába foglalja az eredmények ellenőrzését is.

4. Ellenőrizzük a megoldást.

Miután megkaptuk az eredményeket, ellenőrizzük azok helyességét. Ha szükséges, hibakeresést végzünk és javítjuk az esetleges hibákat. Ezt követően megvitatjuk az eredményeket, esetleg általánosítunk és értékeljük a probléma megoldását.

Programunk tesztelésére alkalmazhatjuk az Excel képletkiértékelő szolgáltatását is. A *Képletkiértékelő* segítségével lépésről, lépésre nyomon követhetjük a függvények kiértékelését ezen a felületen. Ezáltal hatékonyan ellenőrizhetjük és javíthatjuk a kódot, és könnyebben azonosíthatjuk a lehetséges hibákat a folyamat során. Mivel Sprego

függvényeket alkalmazzuk, azaz összetett függvényeket használunk, így lehetőségünk van a megoldások lépésről lépésre történő tesztelésére, ellentétben a speciális célú, többváltozós beépített függvényekkel [34] [29].

3. Programozás oktatása: Melyik nyelvet válasszuk?

A Sprego módszer hatékonysága vitathatatlan, ugyanakkor elismerjük, hogy nem alkalmas minden probléma megoldására. Dolgozatunkban kifejtjük, hogy a kezdő programozás bevezetésére ez az egyik leghatékonyabb módszer. Azonban felismerjük, hogy vannak olyan vezérlési szerkezetek, mint például a saját függvények definiálása és deklarációja, amelyeket Sprego módszerrel nem lehet megvalósítani (az új verziókban már lehetséges, de nem kezdő szinten). Ha az oktatás ezen a pontján kezdünk új ismereteket tanítani a diákoknak az algoritmizálás, programozás területén, akkor választanunk kell egy másik programozási nyelvet.

Szeretnénk felhívni a figyelmet arra, hogy a tanítás során elsősorban a diákoknak az adatok feldolgozására, az algoritmusok megértésére és felépítésére, valamint a már meglévő algoritmizáló képességük hatékony és egyszerű bővítésére kell összpontosítanunk. Több programozási nyelvben elérhetőek az előre megírt függvények, melyek ideálisak a profi programozók számára. Fontos tehát felismernünk, hogy az egyetemi programozás oktatása nem vonható párhuzamba középiskolai programozás oktatásával. Célunk az, hogy a középiskolában a tanulóknak fejlődjön a problémamegoldó képessége, és nem az, hogy egy programnyelvet és a fejlesztőkörnyezetét megismerje. Nem megfelelő programnyelv és fejlesztőkörnyezet használatával a tanórákon a tanulók értelem nélkül gépelnek vagy jó esetben a programozási nyelv szintaktikáját sajátítják el.

3.1. C programozási nyelv

Úgy gondoljuk, hogy a C nyelv az egyik legideálisabb választás a programozás tanítására. Tartalmazza az alapvető utasításokat, egyszerű kezdő szintű kódokat tudunk írni, így gyorsan elsajátítható, ezért kiváló választás kezdőknek az első programozási nyelvként. A C programozási nyelv strukturált jellegű, ahol az utasítások egymás után hajtódnak végre [14]. A problémamegoldó képesség fejlesztésére egy egyszerű, általános célú nyelvet és egy könnyen kezelhető fejlesztőkörnyezetet kell alkalmazni, melyre a C programnyelv maximálisan megfelel.

A C nyelv és a Sprego módszer jelentős hasonlóságot mutatnak. Mindkét programozási nyelvben minimális utasítással dolgozunk, amelyek segítségével számos problémát oldatunk meg. Típusosságát tekintve a C nyelv számos adattípussal rendelkezik. Fontos megjegyezni, hogy a diákoknak az adattípusokkal való első találkozásuk nem a programozás során történik, így ezek használata sem lesz idegen számukra. Már a táblázatkezelő környezetben megismerkednek az adattípusokkal. Tehát a Sprego is típusos nyelvek közé tartozik, amit a grafikus megjelenítéssel támogatnak a táblázatkezelői környezetek. Ahogy a dolgozatunkban kifejtjük, törekednünk kell a szoftverek közötti átjárhatóságra, így az Excelben tanult algoritmusok egyszerűbben, könnyen átültethetők a C nyelvbe.

A világ egyik legrégebbi és legnépszerűbb programozási nyelvének számít, amely olyannyira sikeres volt, hogy más programozási nyelvek ősének tekinthető, mint például a Java, C++ stb. A C programozási nyelv tulajdonságainak és fogalmainak megtanulása, megértése és alkalmazása kiváló alapot biztosít más programozási nyelvek elsajátításához [14] [55]. Ennek köszönhetően a C-ben írt programok könnyen átültethetők más programnyelvekbe is. Konklúzióként megfogalmazhatjuk, hogy a középiskolában a programozás tanítására a C programnyelv egy jó választás lenne.

3.2. Python ütközői

A digitális kultúra tankönyv alapján ajánlott nyelv a Python. Ez a napjainkban elterjedt magasszintű programozási nyelvek közé tartozik, mely az 1990-es években került kifejlesztésre, emiatt egy fiatalnak mondható nyelvről beszélünk [70] [72] [37]. A profi programozók, fejlesztők körében közkedvelt, hiszen alkalmazási területei sokrétűek mint például a játékfejlesztés, a gépi tanulás vagy a webfejlesztés.

Rendkívül egyszerű szintaxisa miatt kitűnik a többi programozási nyelv közül, hiszen tagolása behúzásokkal történik, továbbá dinamikus típusossággal rendelkezik, azaz nem kell külön definiálni azokat [75] [47]. Emellett sok beépített függvénnyel rendelkezik, melyek használata rövid és átlátható kódok írását teszi lehetővé, amely a profi programozók számára elengedhetetlen [68] [46] [74]. Ennek ellenére ebben a programnyelvben megszerzett tudás nehezen építhető át másik nyelvbe, hiszen a rövid kód, hiába könnyen olvasható, ha a működése nem egyértelmű a felhasználó számára [70] [73] [69] [71]. Emiatt, úgy gondoljuk tantermi keretek közé az alapok elsajátítására nem alkalmas.

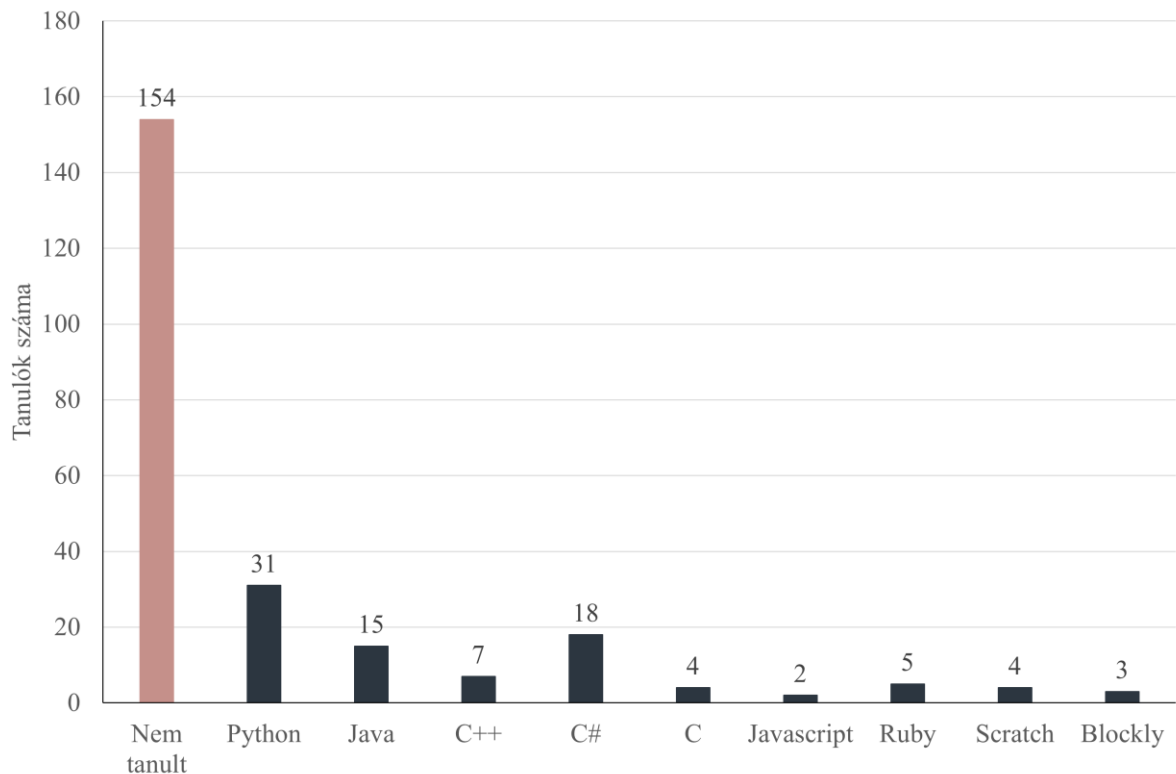
A tanulás folyamatába illeszthető a függvények helyes megírása és használata (hívása), amely ennél a nyelvnél sokszor nem szükséges az előre definiált függvények miatt, melyek hívásával egyszerűen megoldható a legtöbb feladat [37]. Ezzel a diákokat nem segíti a programozás alapjainak megértésére és elsajátítására. Ugyanis a tanórai keretek között, mint már említettük, a CS alapok átadására, a problémamegoldó képesség fejlesztésére, algoritmusok megértésére kell összpontosítanunk, és nem az adott nyelv szintaxisának elsajátítása a cél.

4. Tankönyvi feladatok megoldása és tesztelése

4.1. Középiskolások visszajelzése: kérdőív alapú kutatás

A középiskolások táblázatkezelési és programozási tudását, valamint az összeválogatott feladatokkal kapcsolatos véleményüket kérdőív segítségével vizsgáltuk. A kérdőívet összesen 223 tanuló töltötte ki. A felmérés kontrollált körülmények között, kizárólag tanórákon vagy informatika szakkörökön történt, ahol jelen volt egy felügyelő tanár. Felmérésünk két fő részből állt. Az első részében arra kértük a tanulókat, hogy adják meg, hogy melyik iskolába járnak, hányadik osztályban tanulnak és mi a csoportazonosítójuk (8.1.3). Ezeknek a kérdéseknek a felvetésével azt kívántuk biztosítani, hogy a kérdőív kitöltése anonim maradjon, de osztály szintű visszakövethetőséget tudjunk adni az iskolák számára.

Vizsgáltuk, hogy a tanulók milyen programozási nyelvet ismernek (4. ábra, 8.1.3), valamint mekkora gyakorisággal alkotnak algoritmusokat a digitális kultúra órákon (5. ábra, 8.1.3).

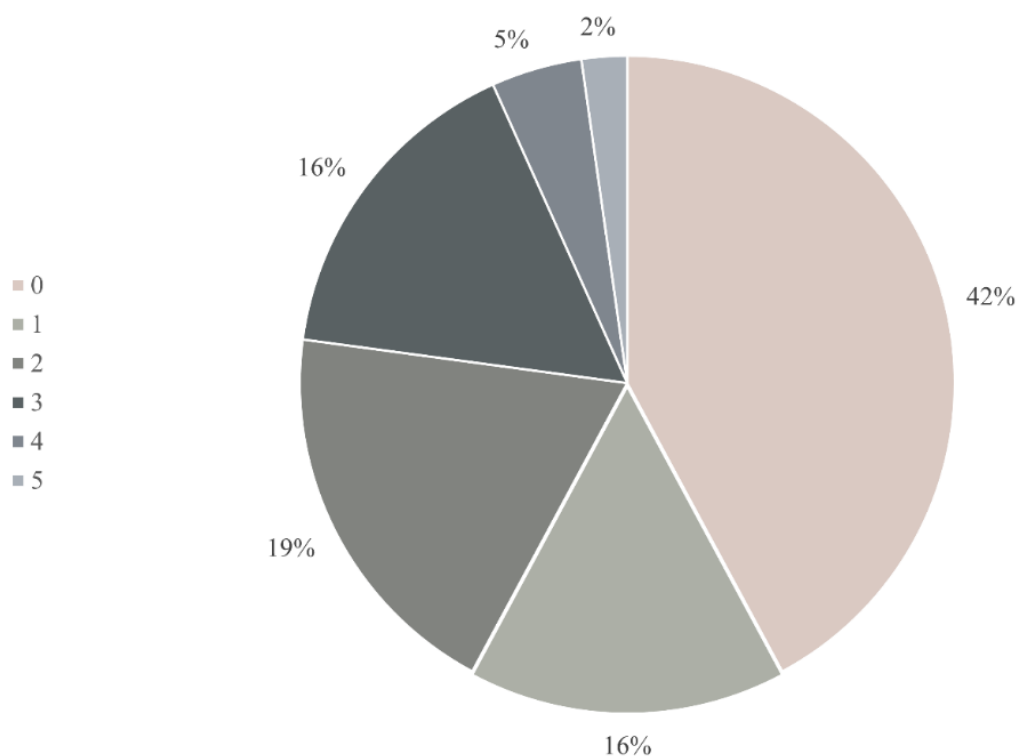


4. ábra. Digitális kultúra/informatika órán megismert programnyelvek (8.1.3). ($n = 223$)

Felmérésünk szerint (4. ábra), 223 résztvevőből 153 tanuló (69,06%) nem részesül programozásoktatásban. Az új digitális kultúra tantárgy tanmenetében 3. osztálytól megjelenik a programozás témakör – A robotika és a kódolás alapjai [12] [13], a Robotika [11] [9] [7] [8], valamint az Algoritmizálás, formális programozási nyelv használata [63] [10] [6] – folyamatosan újabb és újabb programozási környezetek bevezetésével. Az egyórás tanévekben évente 5–6, a kétórás tanévekben ennek a duplája jut a programozás elsajátítására [59]. Általános iskolában ilyen rövid idő alatt 3–4 programozási eszközt és környezetet kellene megismerni és megtanulni, azaz padlórobotokat, microbiteket, robotokat kezelni, blokkprogramozást alkalmazni különböző környezetekben, valamint működőképes algoritmusokat építeni a Flowgorithm alkalmazással. Ezt követően a középiskolában egy konkrét programnyelvet kellene elsajátítani, például a Python nyelvet, amelyet a tankönyv tartalmaz. Az eszközök halmozása azonban nem segíti a programozásoktatás hatékonyságát. Ezzel szemben a tanulók mindössze 30,94%-a (69 fő) rendelkezik tapasztalattal programozás terén. Ennek okai között szerepelhet, hogy a kutatásban részt vevő tanulók még nem tanultak programozást a 3. osztálytól kezdve, ugyanakkor a 2020-as Nemzeti Alaptanterv (NAT) szerint legalább egy év programozási tapasztalattal kellene rendelkezni minden kutatásban részt vevő diáknak, amelyet fontos figyelembe venni a

kutatás során. A második ok, a túl sok anyag feldolgozása rövid idő alatt, ami időbeli csúszásokat eredményezhet a tanítás során. Emiatt lehetetlen, hogy minden diák ugyanolyan mértékben bővítse ismereteit. A Python nyelv rendkívül népszerű, részben azért, mert a tankönyvek ezt a nyelvet mutatják be [63] [10] [6]. A legtöbb tananyag elsajátítása a tankönyv alapján történik, így más programnyelvek ritkábban, vagy egyáltalán nem merülnek fel a programozásoktatásban.

Megvizsgáltuk, hogy milyen rendszerességgel alkalmazzák az algoritmusírást a programozásoktatásban, illetve a táblázatkezelési órák során (5. ábra). Az értékelést egy 0-tól 5-ig terjedő Likert-skálán végeztük, ahol a 0 azt jelentette, hogy egyáltalán nem írnak algoritmusokat, míg az 5 azt, hogy minden órán alkalmazzák. Az algoritmusok írása kiemelkedő fontossággal bír a digitális kultúra tanórákon, hiszen segíti az absztrakt gondolkodás, a logikus következtetés és a problémamegoldó készségek fejlesztését. Azonban fontos lenne az oktatásban megfogalmazni és elkülöníteni az algoritmizálás és a programozás fogalmát, valamint felismerni, hogy a tanítást ne szekvenciálisan alkalmazzuk, hanem a probléma megértésével kezdjük [34]. A programozási nyelvekben írt kód lényegében csak az algoritmusok végrehajtása, így az algoritmusok megértése és írása nélkülözhetetlen a hatékony programozás és algoritmizálás szempontjából.



5. ábra. Algoritmus írás gyakorisága. (8.1.3).

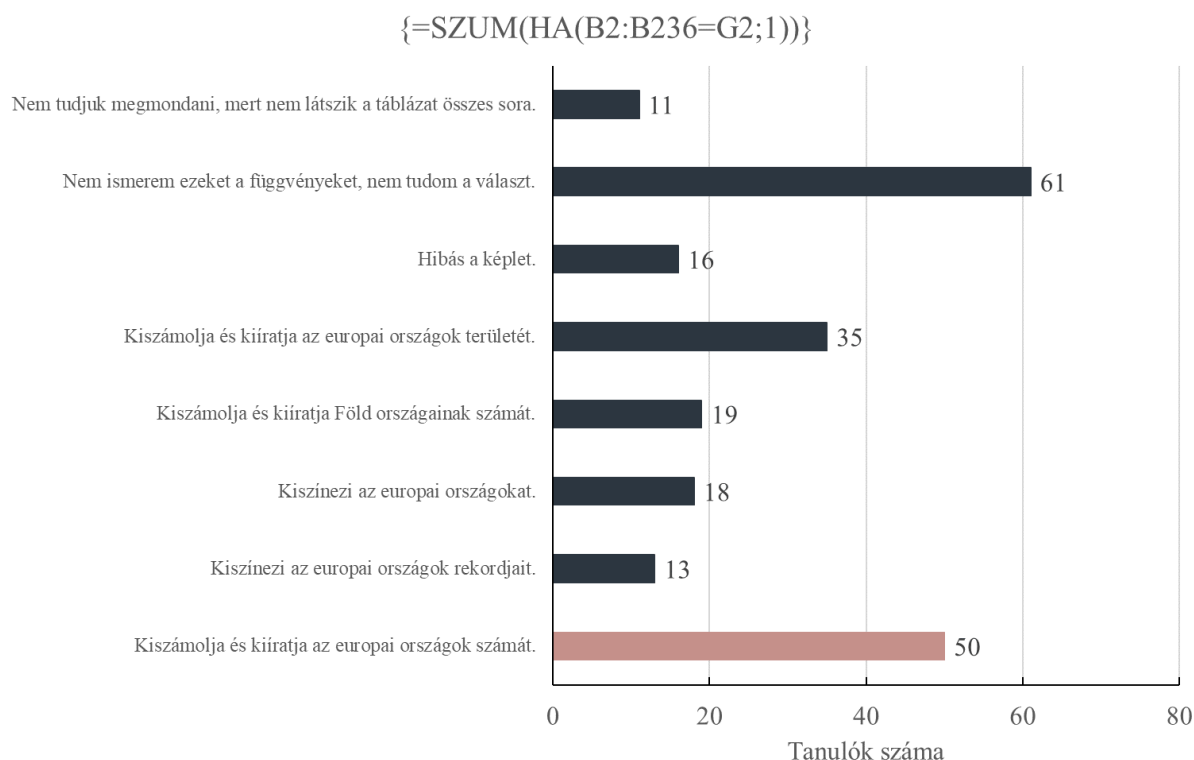
$$(n = 223; \bar{x} = 1,32; \sigma = 1,39; Mo = 0; Me = 1)$$

Ugyanakkor megfigyelhetjük (5. ábra), hogy a tanulók válaszaik alapján átlagosan csak 1,32 gyakorisággal írnak algoritmusokat, ami felveti az oktatási módszerek és a tananyag struktúra felülvizsgálatának szükségességét.

Ezután felmértük a tanulók táblázatkezelési ismereteit négy alapfeladat segítségével. Az első két feladat az algoritmus megértésére irányult, míg a további feladatok a szintaxis felismerésére összpontosítottak. Digitális kultúra órákon és az érettségin leggyakrabban előforduló két algoritmust választottuk ki. A vizsgált algoritmusok között szerepelt a feltételes összegzés (7. ábra, 8.1.3) és a véletlen szám generálás algoritmus (8. ábra, 8.1.3). A célunk az volt, hogy megvizsgáljuk, hogyan tudják a tanulók alkalmazni a tankönyv által feltételezett ismereteiket. Először a feltételes összegzés algoritmusának értelmezését vizsgáltuk.

	A	B	C	D	E	F	G
1	ország	foldresz	fovaros	terulet	fo_ezer_02		
2	Afghanistan	Asia	Kabul	647500	27756		Europe
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	Amerika	The Valley	102	12		
9	Antigua and Barbuda	Amerika	Saint John's	443	67		
231	West Bank	Asia	none	5860	2164		
232	Western Sahara	Africa	none	266000			
233	Yemen	Asia	Sanaa	527970	18701		
234	Yugoslavia	Europe	Belgrade	102350	10657		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

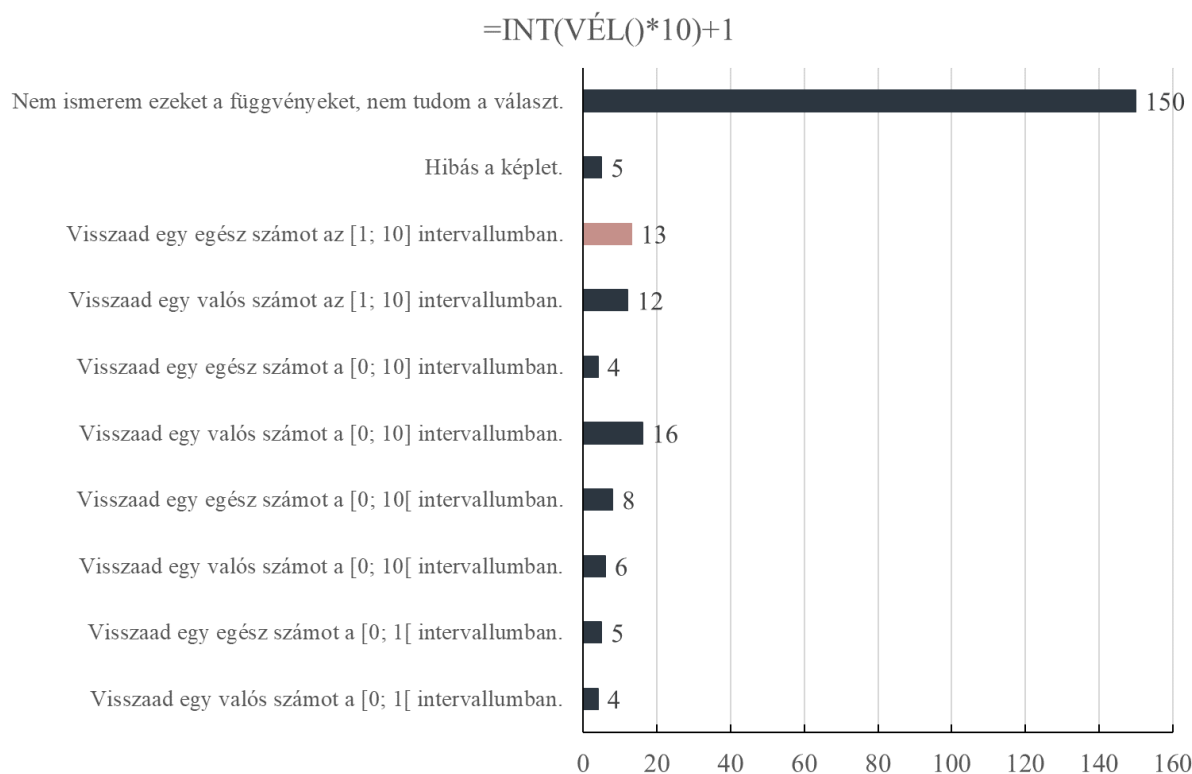
6. ábra. A feladatban szereplő Föld országai tábla (8.1.3).



7. ábra. Feltételes összegzés algoritmusának megértése (8.1.3) ($n = 223$).

Az általunk feltett kérdésre adott válaszok alapján arra következtethetünk, hogy összesen 50 tanuló (22,42%) értelmezte helyesen az algoritmus működését. 61 tanuló (27,35%) jelezte, hogy nem ismeri ezeket a függvényeket. Fontos megjegyezni, hogy a *Táblázatkezelés* témaköre már a 8. osztályban megjelenik, és a tankönyv alapján a SZUM() és a HA() függvények használatát is elsajátítják (8.2). Ennek ellenére az algoritmus működésének megértése továbbra sem minden diák számára egyértelmű (helytelen válasz: 112 tanuló), ami hiányos táblázatkezelői ismeretekre és algoritmikus gondolkodás fejletlenségére is rávilágít.

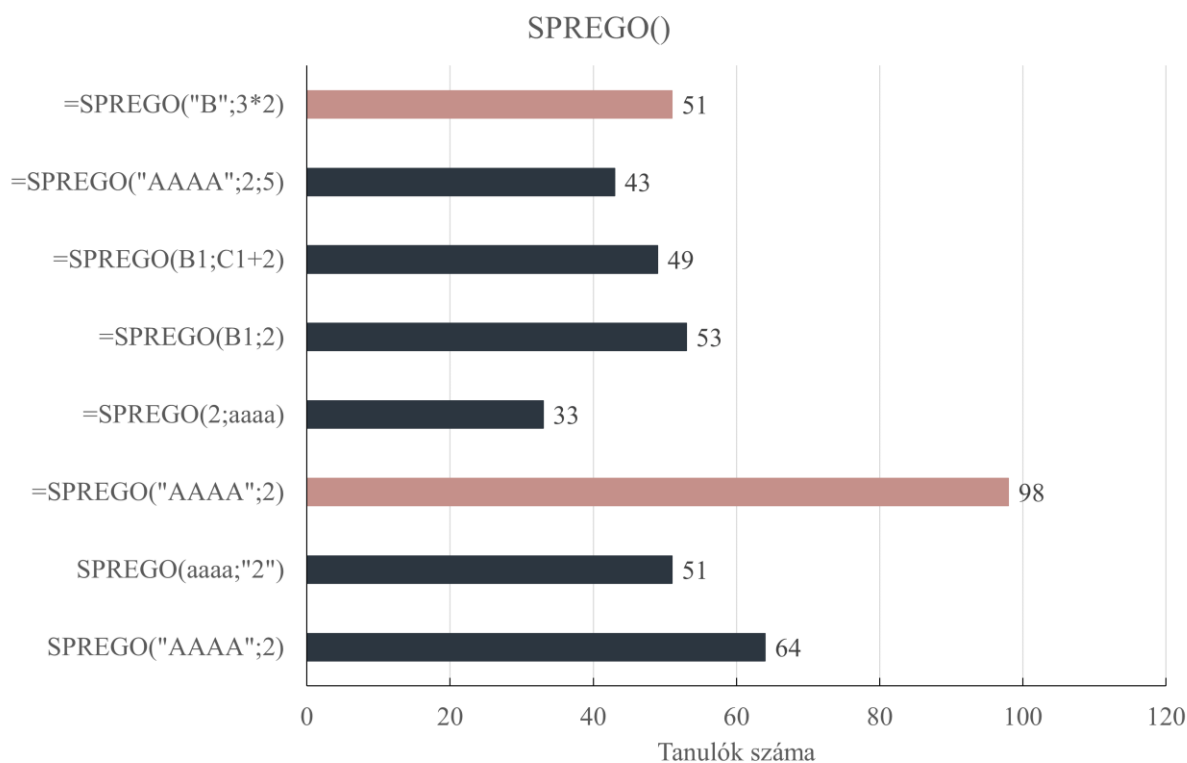
A véletlen szám generálásával már 6. osztályban találkozhatunk [9] programozás területén. A 9. osztályos tankönyvben [63] megismerkedünk az algoritmusával, majd a legtöbb feladatnál 10. osztályban a probléma megoldások során alkalmazzuk, viszont a különböző függvények segítségével. Sprego megoldásban használt VÉL() függvény, amely az algoritmus felépítésében szerepel, a *Táblázatkezelés* témakörben 9. osztályban megjelenik (8.2), így annak ismerete nem okozhat problémát. Az INT() függvényt csak a tizenegyedik osztályban tanulják (8.2), viszont a Python nyelvből való ismeretük, az INT() és a RANDINT() függvény által lehetőséget adhat arra, hogy felismerjék működését.



8. ábra. Véletlenszám generálás algoritmusának megértése (8.1.3) ($n = 223$).

Ezzel szemben 150 diák (67,26%) nem ismeri az algoritmust felépítő függvényeket. Összesen 13 tanuló (5,83%) adott helyest választ a kérdésre.

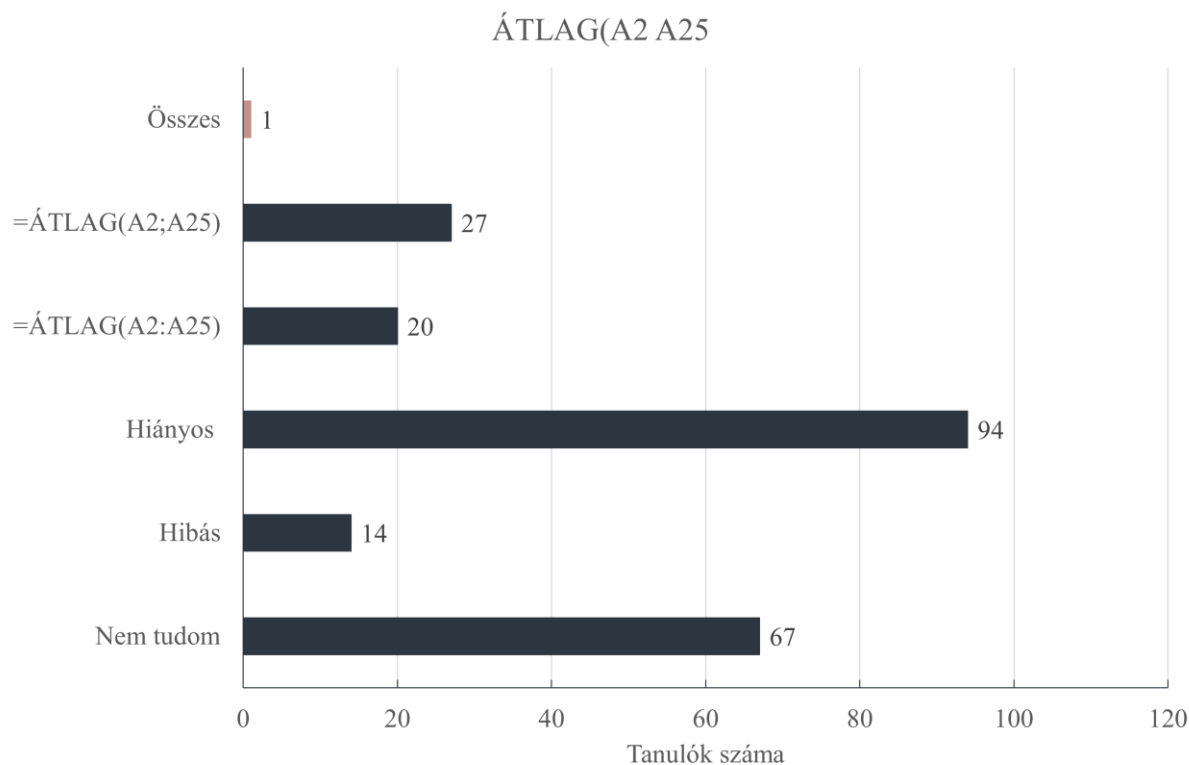
Vizsgálatunk során a diákok szintaxis tudását is mértük Excelben. Következő kérdésben kitaláltunk egy függvényt, melynek két bemenő értéke (argumentuma) van, az első egy szöveg, a második egy egész szám. Vizsgáltuk, hogy a tanulók felismerik-e a különböző argumentumokat. A kérdésre több válaszlehetőséget is megjelölhettek a tanulók.



9. ábra. Sprego(), bemenő értéke: szöveg, egész szám (8.1.3) ($n = 223$).

A feladat két helyes megoldása közül az általánosabb megoldást, a tanulók 43,75%-a (98 fő) jelölte meg, míg a másik helyes megoldást 22,77% (51 fő) ismerte fel (9. ábra). Egyik oka lehet a jelentős eltérés hátterében a precedencia szabály ismeretének hiánya. A másik okának tekinthetjük, hogy a többség nem ismerte fel a szorzás jelét az argumentumban, ami hiányos táblázatkezelési és programozási ismeretekre utal.

A következő feladatban folytattuk a tanulók táblázatkezelési ismereteinek vizsgálatát. Egy olyan függvényt adtunk meg, amelyet a 8. osztályban már megismertek (8.2). A feladatban a helyes képlet kialakítására kellett törekedni. A válaszokat rövid szöveges formában kaptuk meg, majd ezeket kategóriákba soroltuk. Az első kategóriába azoknak a tanulóknak a válaszai kerültek, akik mind a két helyes megoldást megtalálták. Következő kategóriába soroltuk a beérkező válaszok alapján az első helyes megoldást – $\text{=ÁTLAG}(A2;A25)$ – harmadik kategóriába a második helyes megoldást adó tanulói válaszokat – $\text{=ÁTLAG}(A2:A25)$. A negyedik kategóriába a hiányos megoldást tartalmazó tanulók válaszai kerültek, akik csak egy-egy karakterrel egészítették ki a képletet. Ötödik kategóriába soroltuk a hibás megoldásokat. Külön kategóriába azoknak a tanulóknak a válaszai tartoztak, akik jelezték felénk, hogy nem tudják a választ.



10. ábra. *Átlag()* függvény kiegészítése (8.1.3) ($n = 223$).

10. ábra alapján látható, hogy a 223 résztvevőből mindössze egyetlen diák (0,45%) volt az, aki tudta mind a két helyes választ. Emellett érdemes megfigyelni, hogy a vizsgálatban résztvevők 21,52%-a ($1 + 27 + 20 = 48$ fő) tudott legalább egy helyes választ adni, 42,15% (94 fő) pedig hiányos megoldást adott. Érdekes megfigyelni, hogy a tanulók 30,04%-a (67 fő) egyetlen karakterrel sem tudta kiegészíteni a függvényt, ezért arra következtettünk, hogy a válaszaik alapján nem rendelkeznek a középiskolában elvárt táblázatkezelési ismeretekkel. Ez különösen figyelemre méltó, tekintve, hogy 8. osztálytól már elvárható lenne legalább egy hiányzó karakter felismerése.

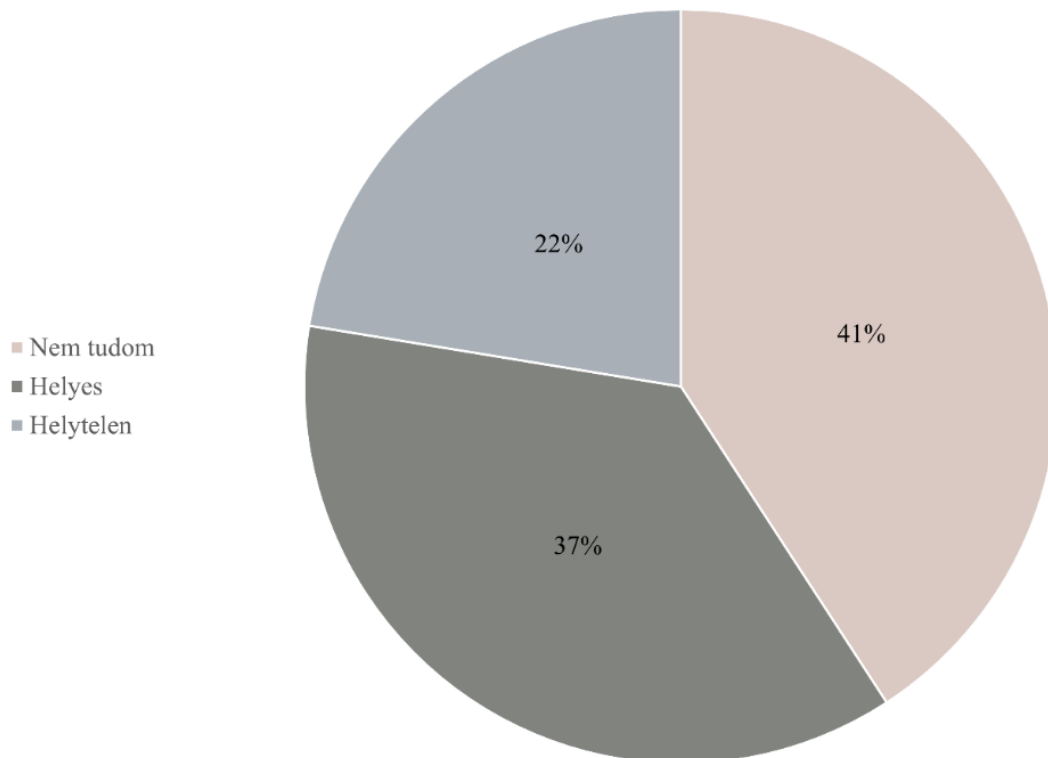
A Python programnyelv szintaktikáját, a programnyelvekben szerzett ismereteiket is vizsgáltuk két 10. osztályos digitális kultúra tankönyv feladattal. Az első kérdés célja az volt, hogy felmérjük, mennyire rögzült a tanulóknak a Python nyelv alapjainak megértése. Ezzel együtt azt is szeretnénk volna megtudni, hogy a 9. osztályban elsajátított tudást mennyire tudják alkalmazni a gyakorlatban. A válaszok rövid szöveges formában érkeztek, így csoportosítottuk őket. Az első kategóriába azoknak a tanulóknak a válaszaik kerültek, akik jelezték felénk, hogy nem tudják értelmezni a kódot. A második kategóriában találhatóak a helyes válaszok, míg az utolsó kategória a helytelen válaszokat tartalmazza.

```

1. idei_év = 2021
2. születési_év = input('Melyik évben születtél? ')
3. születési_év = int(születési_év)
4. volte = input('Volt már idén születésnapod? (i/n) ')
5.
6. szülinapok_száma = idei_év - születési_év - 1
7. if volte == 'i':
8.     szülinapok_száma = szülinapok_száma + 1
9. print('Utoljára a ', szülinapok_száma, '. születésnapodat ünnepelted.', sep=")

```

11. ábra. Python kód, tankönyv 46. oldal 3.feladat [63] (8.1.3).



12. ábra. Python kód értelmezése (11. ábra, 8.1.3) ($n = 223$).

A résztvevők 37%-a (82 fő) tudta csak értelmezni helyesen a fenti kódot. A tanulók 63%-a (141 fő) nem tudta vagy hibásan értelmezte, ami arra enged következtetni, hogy a tanulóknak a 9. osztályban elvárt alapszintű algoritmizálási ismeretet nem sikerült megszerezniük. Ebből az eredményből arra következtethetünk, hogy ilyen szintű tudásra nem lehet alapozni a 10. osztályos hatékony fejlesztést.

A következő feladat (8.1.3) során felmértük, hogy a tanulók milyen vezérlési szerkezeteket ismernek fel egy adott tankönyvi feladatban [10]. A válaszok magukban foglalják az általános iskolában, valamint a 9. és 10. osztályban elsajátítandó tudáselemeket, tehát a tanulók

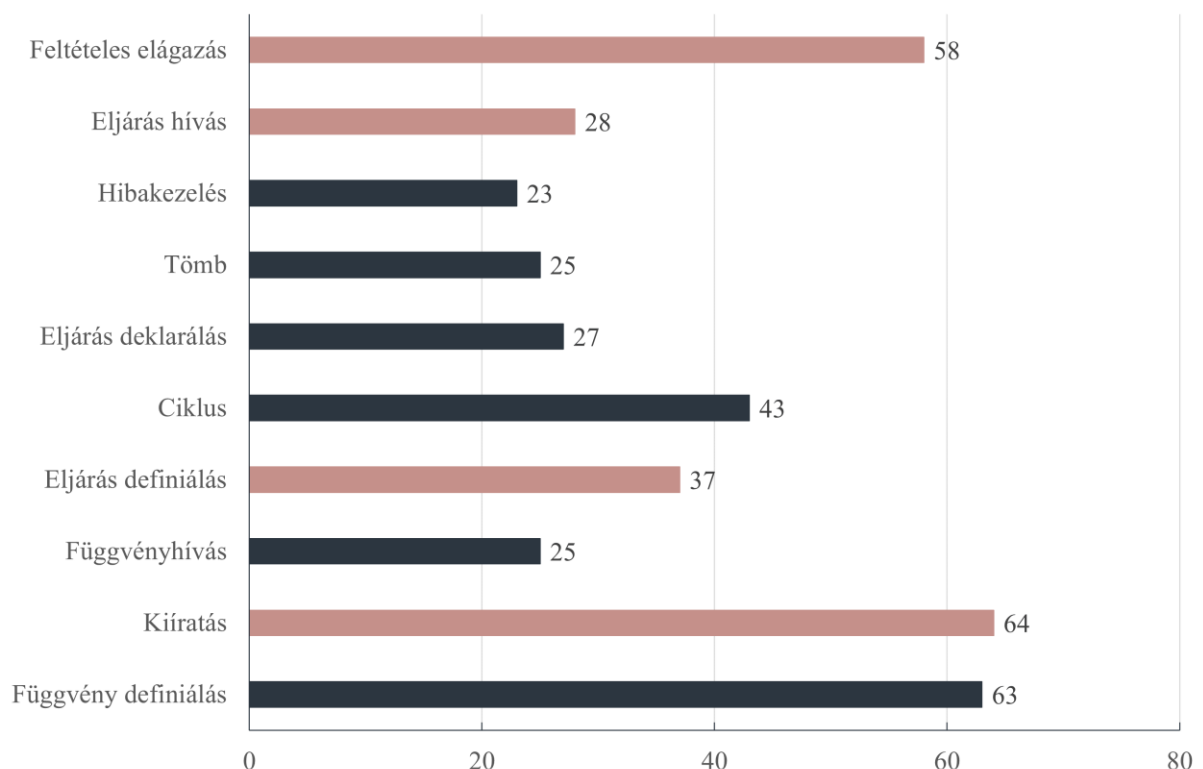
többségének helyesen kellene meghatározni a megfelelő válaszokat. A kérdésre több válaszlehetőséget megadhattak a tanulók.

```

1. def megítél(mondat):
2.     if len(mondat) > 0:
3.         if mondat[-1] != '!' and mondat[-1] != '?' and mondat[-1] != '.':
4.             print('Hát ejnye!')
5.         else:
6.             print('Igazán gyönyörű mondat!')
7.
8. mondat = None
9. while mondat != '':
10.    mondat = input('Írj ide egy mondatot! ')
11.    megítél(mondat)

```

13. ábra. Python kód Tankönyv 56. oldal [10] (8.1.3).



14. ábra. Vezérlési szerkezetek felismerése. (13. ábra, 8.1.3) (n = 223)

A helyes válaszok közül a kiíratást és az elágazást jelölték be a legtöbben, ami csak a tanulók 28,25%-ának, (64 fő) illetve 26,01%-ának (58 fő) volt megfelelő. Az eltérő válaszok alapján megállapíthatjuk, hogy a diákoknak nincs megfelelő mértékű tudásuk a szerkezetek felépítéséről, működéséről, szintaxisáról. Fontos megjegyezni, hogy az általános iskolai ismeretekre építve, a kiíratást és az elágazást általában a 9. osztályban ismétlik és sajátítják el, míg a függvények és az eljárások fogalmaival csak a 10. osztályban találkozhattak a diákok [63] [10]. Úgy

gondoljuk, hogy érdemes lenne a függvények tanítását korábban is bevezetni, hiszen megértése segíthet mind a matematikai ismeretek, mind pedig a programozás terén való fejlődésben, hiszen a használatával az általános funkciója mellett, átláthatóbbá válik a kódunk, jobban megértjük programunk működését. Emellett a függvények tanítása elősegíti a moduláris gondolkodást és a hatékonyabb kódírást, ami későbbi programozói készségeik, képességeik fejlődéséhez elengedhetetlen.

A kérdőív további részében a digitális kultúra tankönyv *Algoritmizálás és programozási nyelv használata* című fejezetéből választottunk 13 különböző feladatot. Minden feladatnál rögzítettük, hogy a tanuló melyik programozási környezetet választaná a feladat megoldásához. Főként az érdekelt minket, hogy a feladatok motiválják-e a tanulókat a probléma megoldására, sikerült-e felkelteni az érdeklődésüket a feladatok iránt. A feladatok során kiemelt figyelmet fordítottunk arra, hogy a tanulók hogyan oldják meg a feladatokat Excelben és Pythonban vagy más programnyelveken.

A dolgozat következő szakaszában részletesen elemezzük a kérdőívben szereplő feladatokra adott válaszokat és véleményeket, illetve vizsgáljuk a tankönyvben megtalálható további feladatokat [10].

4.2. Algoritmizálási megközelítések a feladatok megoldásában

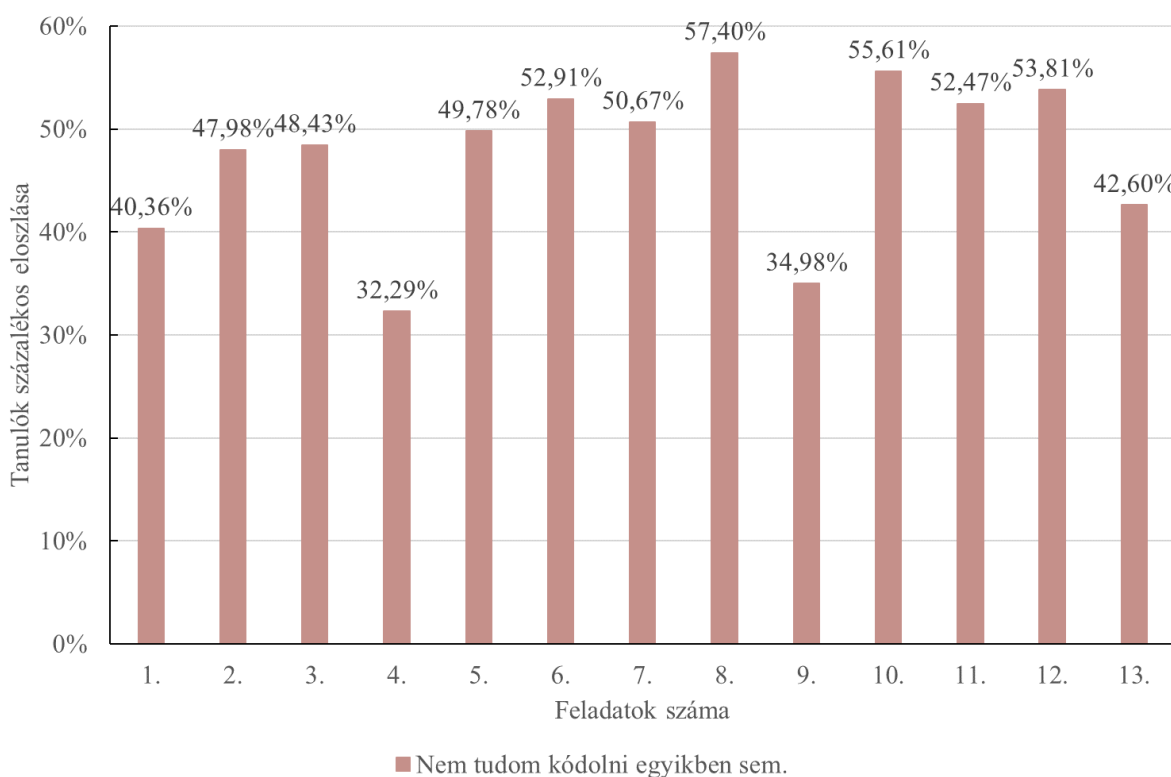
Minden feladatot két programnyelven oldottunk meg, az Excel és a Python programozási környezetekben. Először felmértük, megvizsgáltuk, hogy az általános iskolában és a 9. osztályban feltételezetten megszerzett tudás mennyire alapozza meg a feladatokat, a tanulók hogyan tudják alkalmazni a korábban megszerzett tudást a 10. osztályban. A feladatok kiválasztása és megoldása közben kiemelt figyelmet fordítottunk a feladatok szövegére, a tantárgyi összefüggésekre, és arra, hogy a megoldás folyamán fejlesztjük-e a tanuló problémamegoldó gondolkodását. Megvizsgáltuk, hogy szükség van-e az algoritmikus megközelítés alkalmazására.

4.2.1. Feladatok vizsgálata a kérdőívünk alapján

Dolgozatunkban a kérdőívben szerepelt feladatokra szeretnénk elősorban kitérni. A feladatok választásakor igyekeztünk változatosak lenni, így minden témakörből válogattunk egy-egy feladatot. A feladatok szövegét módosítottuk annak érdekében, hogy ne befolyásolják a

tanulókat, és ne lehessen egyértelműen megállapítani, hogy programozási feladatokról kérdeztük őket. A tankönyvben található feladatok eredeti formában megtalálhatók a 8.1.1. mellékletben.

Minden egyes feladatnál azonos kérdéseket tettünk fel. Először rögzítettük, hogy a tanulók melyik programozási környezetet választanák a feladat megoldásához. Ennek a kérdésnek a célja az volt, hogy felmérje, mennyire érzik magukat magabiztosnak a különböző környezetekben, és hogy melyiket preferálják a leginkább. Négy válaszlehetőség közül választhatnak a tanulók: „Nem tudom kódolni egyik környezetben sem”, „Más programozási nyelven”, „Mindkettőben meg tudom oldani”, valamint „Excel”, vagy „Python” környezetet választaná. Az egyik legkiemelkedőbb adatunk a „Nem tudom kódolni egyik környezetben sem” válaszlehetőség. Ez különösen figyelemre méltó, mivel a táblázatkezelés témakörét már a 8. osztályban tanulják, míg az algoritmizálás már a 3. osztálytól megjelenik, a vizsgált tanulók esetén 5. osztálytól.

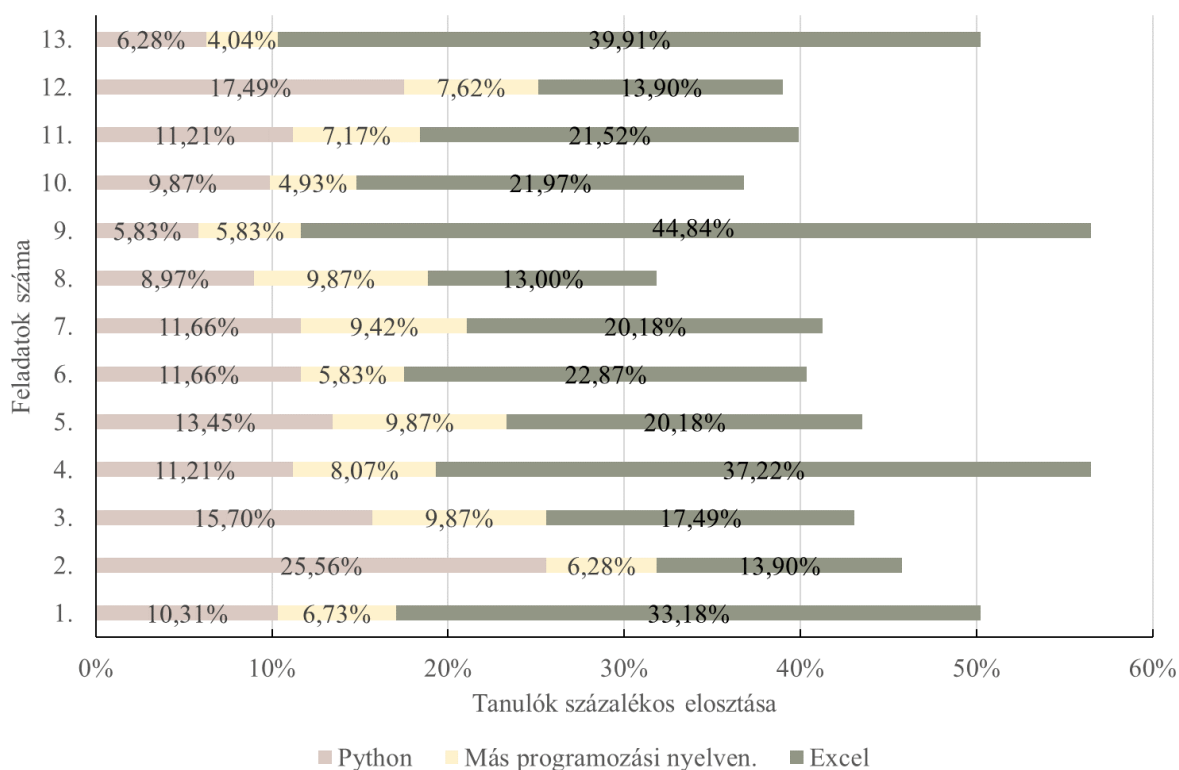


15. ábra. A tanulók véleménye: Nem tudom kódolni egyik környezetben sem (8.1.3) (n=223)

A diákok 13 feladat során 7 esetben nyilatkoztak úgy, hogy egyik környezetben sem tudnának hozzákezdni a probléma megoldásához (15. ábra). Azonban a feladatok megoldása, során

nem volt szükség arra, hogy a tanulók olyan tudáselemeket alkalmazzanak, amelyeket nem sajátítottak el a 10. osztályos korukig. Ez arra utal, hogy az informatika/digitális kultúra oktatása során talán nem sikerült megfelelő alapokat lefektetni a tanulók számára.

Következőekben összehasonlítottuk, a programozási környezetet az Excel környezettel. Bár a feladatok átírásával próbáltunk új megközelítést alkalmazni, azok jellegéből egyértelmű volt, hogy programozási feladatokról van szó.

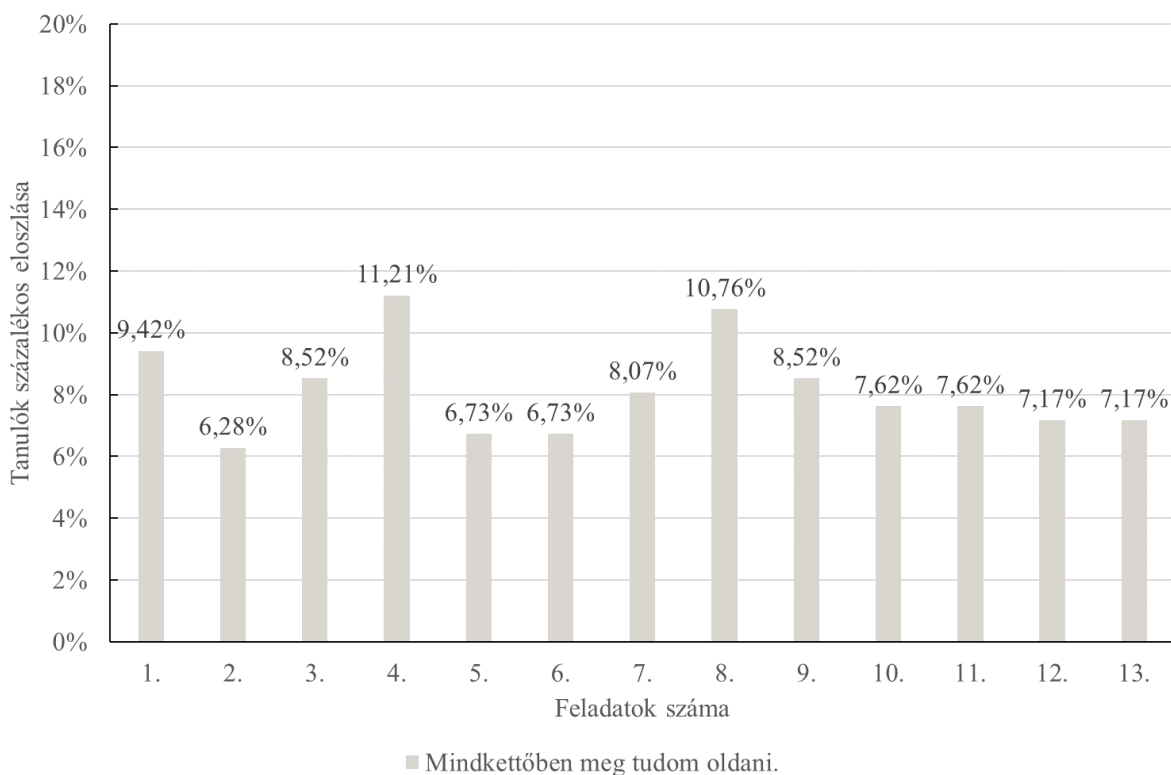


16. ábra. A tanulók véleménye: Melyik platformon tudnád megoldani a feladatot? (8.1.3) (n=223)

Mégis érdekes módon, több tanuló az Excel környezetet jelölte meg megoldási platformként. Ugyancsak 13 feladat közül 7 feladatban a diákok többsége a táblázatkezelő környezetet választaná a probléma megoldására (16. ábra). Ez arra utal, hogy még akkor is, ha egy programozási problémát kell végrehajtani, hogy megoldást találjanak a problémára a diákok gyakran ahhoz a környezethez nyúlnak vissza, melyben tudásukat magabiztosabbnak érzik. A tanulók sokkal jobban preferálják az Excel környezetet. Ennek oka, hogy a grafikus felület segítségével könnyebben tudnak kódolni, folyamatos visszajelzést kapnak a munkájukról, és a meglévő tudásukhoz jobban illeszkedik. Emellett valószínűbb, hogy a mindennapi életükben hamarabb

használnak egy probléma megoldására táblázatkezelő környezetet, mint egy Python vagy más programnyelvet.

Az első kérdésnél a tantárgyi integrációt is vizsgáltuk. Kíváncsiak voltunk arra is, hogyan tudják meglévő tudásukat más környezetben kamatoztatni. A tantárgyakon belüli integráció fontos, mert segíti a diákokat abban, hogy összekapcsolják és átjárják az egyes témakörökben tanult fogalmakat és készségeket. Ezáltal mélyebb és összetettebb megértést fejleszthetnek ki a tananyag területeiről, és jobban felkészülhetnek a valós élet kihívásaira, ahol gyakran szükség van a különböző tudásterületek kombinálására és alkalmazására. Hospitálásaink és tapasztalataink alapján megfigyeltük, hogy a tanulók egy tanórán belül szigorúan egyetlen szoftverrel dolgoznak. Nincs átjárás a témakörökben.



17. ábra. A tanulók véleménye: Mindkettőben meg tudom oldani (8.1.3). (n=223)

Vizsgálatunk során kiderült, hogy a tanulók mindössze átlagosan a tanulók 8, 14%-a (≈ 18 fő) rendelkezik az integrált megközelítés képességével (17. ábra). Ez azt jelenti, hogy a legtöbb tanuló még a középiskola végére sem tudna problémát megoldani más szoftverek segítségével. Ennek hátterében valószínűleg az áll, hogy az informatika/digitális kultúra oktatása

túlzottan szoftvercentrikus és a szoftverek közötti átjárás hiánya miatt a diákok nem tanulják meg, hogy egy adott problémát melyik környezetben tudná megoldani hatékonyabban.

A következő részben Likert-skálán vizsgáltuk véleményüket a feladatokkal kapcsolatban. A skálán az értékek 0-tól 5-ig terjedtek, ahol 0 azt jelentette, hogy nagyon nem értek egyet, az 5 pedig azt, hogy nagyon egyetértek. A beérkező válaszok elemzése során számoltunk átlagot, szórást, móduszt és mediánt.

A tanórán az egyik legfontosabb motiváló tényező a tananyag tartalma, így a következő kérdésben azt vizsgáltuk, hogy a tanulók mennyire találják érdekesnek, értelmesnek tehát motiválóknak a feladatokat.

3. táblázat. A tanulók véleménye a feladatok érdekességéről. (8.1.3) ($n = 223$).

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
$\bar{x}$	1,84	1,65	1,86	1,83	1,61	1,55	1,5	1,61	1,74	1,59	1,69	1,68	1,77
σ	1,55	1,61	1,75	1,74	1,7	1,74	1,65	1,75	1,84	1,8	1,85	1,79	1,85
Mo	0	0	0	0	0	0	0	0	0	0	0	0	0
Me	2	1	2	2	1	1	1	1	1	1	1	1	1

A 3. táblázat alapján azt láthatjuk, hogy a feladatok pontszámai átlagosan sem érték el a 2 értéket. A szórás viszonylag alacsony, ami arra utal, a medián és a módusz értékekkel együtt, hogy a feladatok nem sarkallják a tanulókat a megoldásra.

4. táblázat. A tanulók véleménye a feladatok értelmességéről. (8.1.3) ($n = 223$).

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
$\bar{x}$	2,22	1,92	2,01	1,89	1,78	1,74	1,66	1,77	1,94	1,78	1,82	1,88	1,97
σ	1,65	1,65	1,7	1,75	1,71	1,74	1,68	1,8	1,82	1,74	1,8	1,8	1,83
Mo	0	0	0	0	0	0	0	0	0	0	0	0	0
Me	2	2	2	2	1	1	1	1	2	1	1	2	2

A 3. táblázatot és a 4. táblázatot összehasonlítva láthatjuk, tanulók csak minimális mértékben értékelték magasabban a feladatok értelmességét. Egyes feladatok pontszámai átlagosan már elérték a 2 értéket, és a tanulók fele már 2-es vagy annál nagyobb pontszámot adott 7 feladatra. A feladatok megoldása során, saját tapasztalataink és a tanulók véleménye alapján megállapítottuk, hogy sok feladat utasítása nem egyértelmű. Továbbá, a feladatok tartalma értelmetlennek tűnt számukra, nem foglalkozott a mindennapi életben előforduló kihívásokkal, ami miatt a tanulók nem érzik a megoldás fontosságát. Emellett néhány feladat túl kevés vagy éppen

túl sok mintaelemmel dolgozott, ami nehezítette az ellenőrzést. Konklúzióként megállapíthatjuk, hogy 223 tanuló szerint a kérdőívben szereplő feladatok nem motiválóak, és mint tudjuk, motiváció nélkül nincs tanulás.

A kutatásban részt vevő tanulókat megkérdeztük, hogy az eddigi Excel, Python tudásuk alapján mennyire biztosak egy-egy feladat megoldásában. Kitöltés során hangsúlyoztuk, hogy azok a tanulók, akik más programnyelvet tanultak, ők abban a programnyelvben lévő ismereteiket mérlegeljék.

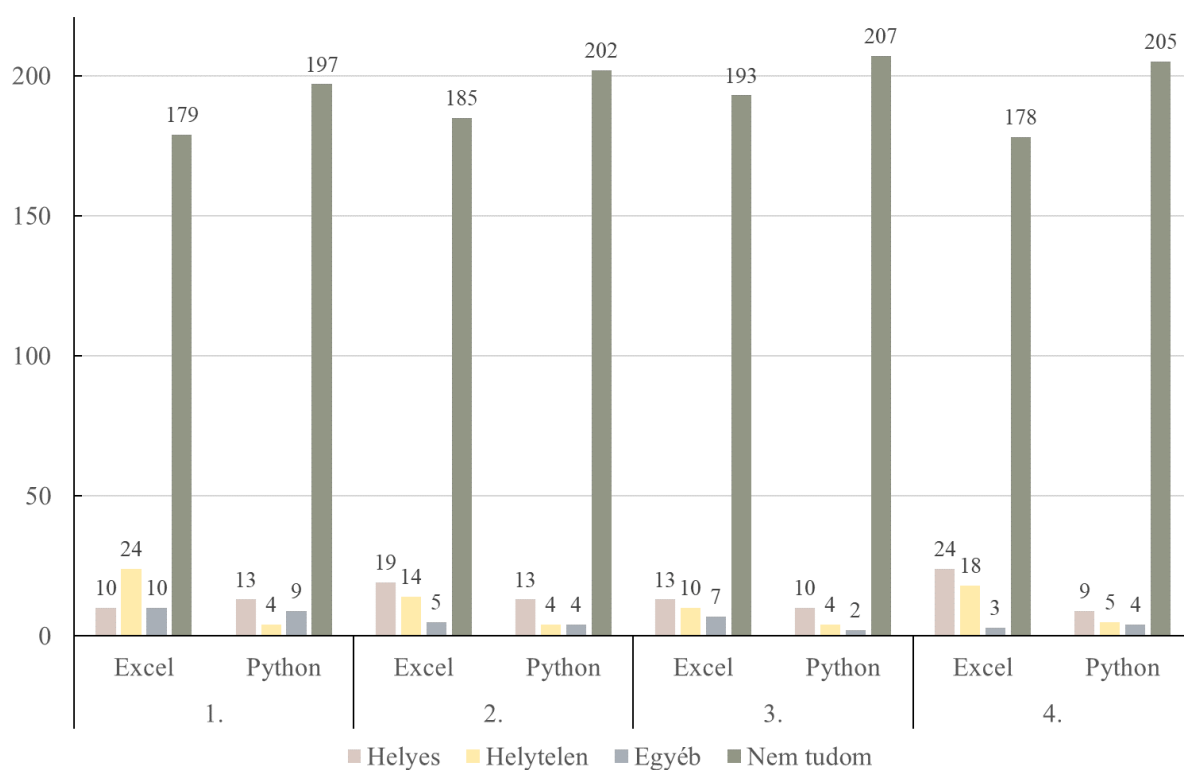
5. táblázat. A tanulók véleménye az Excel és Python tudásukról. (8.1.3) ($n = 223$).

Eddigi tudásom alapján meg tudom oldani Excelben													
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
$\bar{x}$	1,28	1	0,84	1,4	0,89	0,76	0,72	0,71	1,42	0,8	0,81	0,78	1,18
σ	1,49	1,45	1,36	1,71	1,38	1,29	1,3	1,28	1,77	1,39	1,42	1,36	1,65
Mo	0	0	0	0	0	0	0	0	0	0	0	0	0
Me	1	0	0	1	0	0	0	0	0	0	0	0	0
Eddigi tudásom alapján meg tudom oldani Pythonban													
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
$\bar{x}$	0,59	0,59	0,61	0,7	0,69	0,57	0,57	0,58	0,57	0,53	0,54	0,56	0,54
σ	1,32	1,27	1,32	1,38	1,42	1,26	1,29	1,29	1,29	1,25	1,28	1,3	1,28
Mo	0	0	0	0	0	0	0	0	0	0	0	0	0
Me	0	0	0	0	0	0	0	0	0	0	0	0	0

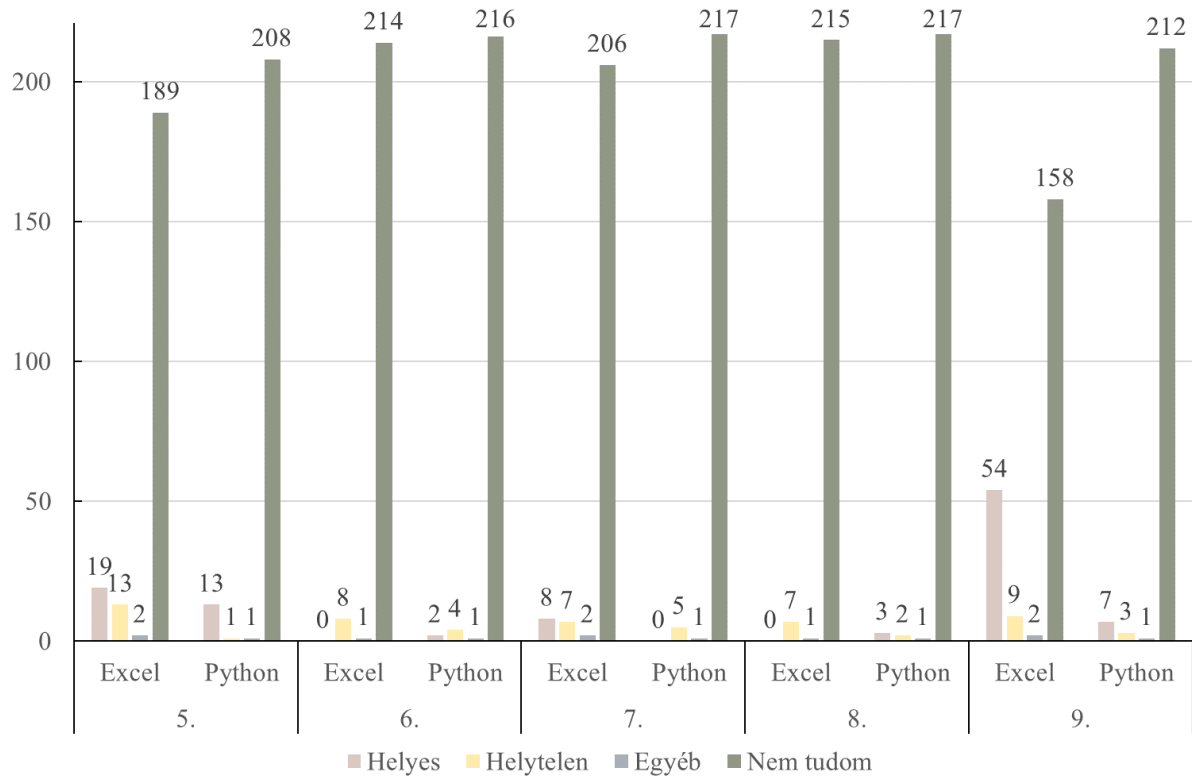
Bár a tanulók alacsonyra értékelték mindkét környezetben a tudásukat az összes feladatnál, az alacsony pontszámokból kiderül, hogy a legtöbb feladatban a táblázatkezelő tudásukban biztosabban érezték magukat, közel kétszer olyan magas pontszámot adtak neki, mint a Python tudásukra. Ezek az adatok egyértelműen bizonyítják, hogy a diákok platformválasztása főként az Excel környezetre összpontosul.

Emellett vizsgáltuk azt is, hogy melyik platformon milyen eszközöket alkalmaznának a tanulók a feladatok megoldásához. A válaszok rövid szöveges formában érkeztek, majd azokat kategorizáltuk. Az első kategóriába a helyes megoldáshoz szükséges elemek válaszai, a másodikba pedig a hibás megoldások tartoztak. A következő csoportba az Egyéb megoldások kerültek. Táblázatkezelés és a Python, vagy más programnyelv esetén egyéb megoldásokba tartoztak például a függvényvarázsló használata, vagy „amit a tanár mond, amit diktál” módszer alkalmazása, valamint a „lemásolom a tábláról” típusú válaszok. Fontosnak tartjuk ezek kiemelését, mivel tapasztalataink alapján a diákok többsége ténylegesen ezt a megközelítést alkalmazza az

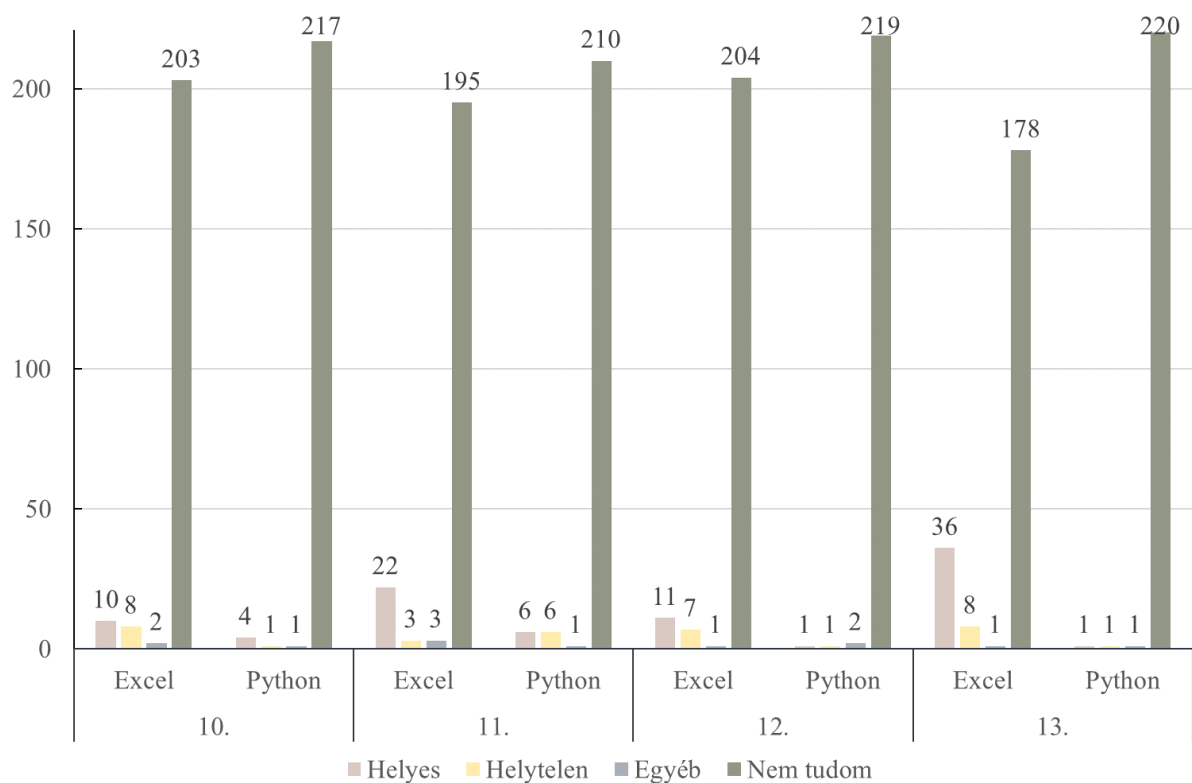
órákon, egész órán át csak gépelik a tábláról, a kivetítőről a tananyagot, amikor ellenőrzik az órai munkát. Az utolsó kategóriát pedig azoknak a tanulóknak a válasza alkották, akik jelezték felénk, hogy nem tudják a feladatot megoldani.



18. ábra. A tanulók 1–4. feladatra adott megoldási ötletei Excel és a Python környezetben. (8.1.3) ($n = 223$).



19. ábra. A tanulók 5–9. feladatra adott megoldási ötletei Excel és a Python környezetben. (8.1.3) ($n = 223$).



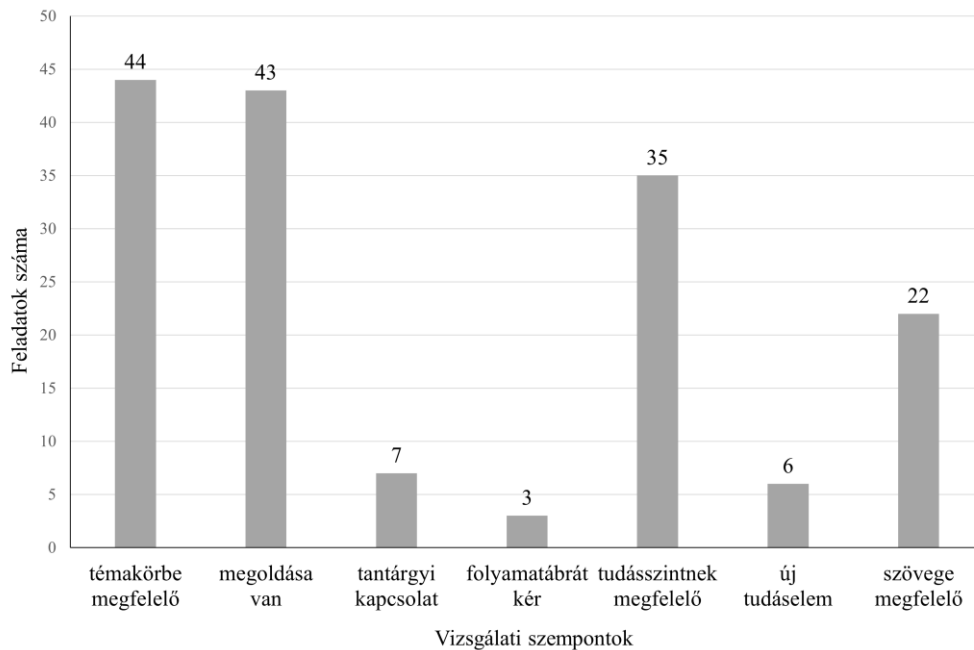
20. ábra. A tanulók 10–13. feladatra adott megoldási ötletei Excel és a Python környezetben. (8.1.3) ($n = 223$).

A válaszok alapján megállapítható, hogy a tanulók nem tudták megoldani a feladatokat semelyik környezetben. Azonban egy-egy feladatnál több próbálkozás történt az Excel környezetben, ezáltal több helyes megoldás született, mint Pythonban. Ez a megfigyelés arra utalhat, hogy a tanulók magabiztosabban és könnyebben alkalmazkodnak a táblázatkezelő környezethez, annak ellenére, hogy a táblázatkezelés témakörét csak a 8. osztályban kezdik el elsajátítani.

Bár az adatok alapján megállapítható, hogy a tanulók algoritmikus képességei egyik környezetben sem fejlődtek ki. Választ kaptunk a feladatok motivátlanságára, és biztosak lehetünk a diákok platformválasztásában. Bár sokan úgy gondolják, hogy egy programozási környezet „színesebb” és nagyobb motivációt eredményez, a táblázatkezelő egyszerűsége és „unalmas” jellege valószínűleg biztonságot nyújt a diákoknak, ezért hamarabb fordulnak ehhez a környezethez a problémák megoldása során. A motiváció fenntartására és az algoritmizáló képességek, készségek fejlesztésére pedig a Sprego módszer nyújthat megoldást a tanulók számára, amely megállja a helyét bevezető programnyelvként is (2.).

4.2.2. A tankönyv feladatainak elemzése

A dolgozatunkban kiemeltünk, minden témakörből 1-1 feladatot, amelyek a kérdőívben is megjelentek, ezek kiválasztása során kiemelt figyelmet fordítottunk arra, hogy jól szemléltessék a fejezetekben található feladatokat, ezáltal megfelelő visszajelzést kaphassunk a tanulóktól. Ezen kívül az összes feladatot megoldottuk mind Python nyelven mind Sprego módszerrel és előre lefektetett szempontok szerint elemeztük azokat. Megvizsgáltuk, hogy a témakörökbe megfelelőek-e, rendelkeznek-e megoldással, tantárgyi kapcsolat fellelhető-e bennük. Továbbá, a feladat megoldása során kér-e folyamatábrás előkészítést, rendelkeznek-e a diákok a megfelelő tudáselemekkel vagy kiegészülnek-e vele az eddigi ismereteik. Végso soron a feladat szövegét vizsgáltuk, hogy megfelelően van-e megfogalmazva a korosztály számára.



21. ábra. Tankönyvi feladatok elemzése [10].

A tankönyvben összesen 50 megoldandó feladat található, ezek közül több rendelkezik részfeladatokkal, melyeket külön-külön is megvizsgáltunk. Vizsgálatunk során megfigyeltük, hogy a feladatok nagy része a témaköröknek megfelelően lettek elhelyezve, megoldásuk van, illetve a kerettanterv és a tankönyvek alapján a diákok rendelkeznek megfelelő tudáselemekkel a megoldáshoz. Fontos megemlíteni, hogy hiába rendelkezik egy feladat megoldással, ha a tartalma nem megfelelő a korosztály számára, hiszen vagy túl egyszerű, vagy olyan ismereteket tartalmaz, melyet a tanuló még nem sajátított el tanulmányai során.

Megfigyelhető, hogy csekély azon feladatok száma, amely más tantárgyi tudást használ fel a megoldás során vagy új tudáselemmel bővítené az eddigi ismereteket. Észrevettük, hogy a megoldások során a ciklusok, elágazások és kiíratások használatára van szükség, amelyet már az előző tanévekben elsajátítottak és gyakoroltak a tanulók. Illetve a feladatok többségénél a diákoknak nincs szükségük algoritmus tervezésére, önálló gondolkozásra, hiszen a megoldás megtalálható a tankönyvben. Tehát, a szintaktika megismerésére és elsajátítására helyezi a hangsúlyt, nem pedig az önálló vagy csoportos problémamegoldásra.

A további vizsgálat céljából bemutatunk olyan feladatokat a tankönyvben, amelyekben ugyanazt az algoritmikus megközelítést alkalmazzuk.

1. tankönyvi feladat

Állítsunk elő egy ezer és tízezer közötti egész számokat tartalmazó, húszelemű listát! A lista elemei azoknak a járműveknek a tömegét adják meg, amiket ma egy komphajó átvitt a folyón. Nehéznek számítanak a 9300 kilónál nehezebb járművek. Írjunk programot, ami válaszol a következő kérdésekre:

- Volt-e olyan jármű ma a hajón, ami nehéznek számít? Írjuk ki, ha volt ilyen!
- Hány ilyen jármű volt?
- Hány kiló járművet vitt át a komp ma összesen?
- Mennyi a ma átvitt, nehéznek számító járművek össztömege?
- Ha a „nehéz” holnaptól nem 9300, hanem 9000 kilogramm, hány helyen kell átírni a programot? Mit kell tennünk, ha azt szeretnénk, hogy az ilyen változások egyszerűen, egyetlen helyen való átírást jelentsenek?

22. ábra. Tankönyv 51. oldal 5. feladat.[10]

	A	B	C	D	E
	Lista	Nehéz jármű	Hány db	Összesen (kg)	Nehéz járművek tömege
1					
2	6495	Volt nehéz jármű	1	111537	9460
3	6846				
4	3298				
5	3739				
6	7972				
7	6629				
15	1399				
16	5539				
17	7874				
18	1494				
19	4839				
20	7537				
21	6939				

23. ábra. Excel megoldás (22. ábra).

képlet helye: A2:A21
`=INT(VÉL()*9000+1000)`

képlet helye: B2

{=HA(SZUM(HA(A2:A21>9300;1))>0;"Volt nehéz jármű"; "Nem volt nehéz jármű.")}

képlet helye: C2

=SZUM(HA(A2:A21>9300;1))

képlet helye: D2

=SZUM(A2:A21)

képlet helye: E2

=SZUM(HA(A2:A21>9300;A2:A21))

3. képlet 1. tankönyvi feladat megoldása

```
import random

tomegek = []
for _ in range(20):
    tomegek.append(random.randint(1000,10000))

print (tomegek)

volt_nehez = False
nehezek_száma = 0
össztömeg = 0
nehezek_össztömege = 0
for tömeg in tomegek:
    össztömeg = össztömeg + tömeg
    if tömeg > 9300:
        volt_nehez = True
        nehezek_száma += 1
        nehezek_össztömege += tömeg

if volt_nehez:
    print('Volt 9300 kilónál nehezebb jármű.')

print(nehezek_száma, '9300 kilónál nehezebb jármű volt.')
print ('Ma', össztömeg, ' kilónyi járművet vitt át a komp.')
print('Ebből nehéznek összesen', nehezek_össztömege, 'kilónyi számít.')
    Volt 9300 kilónál nehezebb jármű.
    2 9300 kilónál nehezebb jármű volt.
    Ma 124319 kilónyi járművet vitt át a komp.
    Ebből nehéznek összesen 19262 kilónyi számít.
```

24. ábra. Python megoldás (22. ábra).

Az első tankönyvi feladat során a feltételes összegzés algoritmusát és a véletlenszám generálás algoritmusát kell alkalmaznunk. A feladatok megoldásában látható, hogy mindkét esetben ugyanazt az algoritmust alkalmazzuk a feltételes összegzés során. Azonban a véletlenszám generálása eltérő módon történik: Python nyelvben a szokásos módon egy beépített függvényt hívunk meg, míg Excelben algoritmizálunk a véletlenszámok előállításához. Ezáltal a táblázatkezelőben való megoldás hatékonyabban fejleszti az algoritmikus gondolkodás kialakulását, mivel a diákoknak több teret biztosít a problémamegoldáshoz és az algoritmusok saját elkészítéséhez..

2. tankönyvi feladat

Írjunk olyan eljárást, amely egy paraméterként kapott szóról eldönti, hogy magánhangzóval vagy mássalhangzóval kezdődik, és a döntését képernyőre írja!

- Hogyan tudjuk egy változóban tárolt szó első betűjét megkapni?
- Meg kell vizsgálnunk, hogy a megkapott első betű benne van-e egy listában. A mássalhangzókat vagy a magánhangzókat érdemes listába gyűjtenünk?



```

eljárás magánhangzó_mássalhangzó(szó)
magánhangzók = ['a', 'á', 'e', 'é', 'i', 'í', ... ]
ha szó[0] eleme magánhangzók-nak:
    ki: „magánhangzó”
különben:
    ki: „mássalhangzó”
eljárás vége
  
```

25. ábra. Tankönyv 60. oldal 5. feladat. [10]

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	magánhangzók	a	á	e	é	i	í	o	ó	ö	ő	u	ú	ü	ű
2															
3	almafa	almafa magánhangzóval kezdődik.													

szavak_kezdet

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	magánhangzók	a	á	e	é	i	í	o	ó	ö	ő	u	ú	ü	ű
2															
3	békatutaj	békatutaj mássalhangzóval kezdődik.													

szavak_kezdet

26. ábra. Excel megoldás (25. ábra).

képlet helye: B3

{=A3&" " &HA(BAL(A3)=B1:O1;

"magánhangzóval kezdődik";"mássalhangzóval kezdődik")&"."} }

4. képlet 2. tankönyvi feladat megoldása

```
def magánhangzó_mássanhangzó(szó):
    magánhangzók = ['a', 'á', 'e', 'é', 'i', 'í', 'o', 'ó',
                    'ö', 'ő', 'u', 'ú', 'ü', 'ű']
    if szó[0] in magánhangzók:
        print(szó, 'magánhangzóval kezdődik.')
    else:
        print(szó, 'mássalhangzóval kezdődik')

magánhangzó_mássanhangzó('almafa')
magánhangzó_mássanhangzó('békátutaj')
almafa magánhangzóval kezdődik.
békátutaj mássalhangzóval kezdődik
```

27. ábra Python megoldás (25. ábra)

A második tankönyvi feladatban szavak kezdőbetűjének vizsgálata történik, annak eldöntése, hogy magánhangzóval vagy mássalhangzóval kezdődik az adott szó. A feladat megoldása során látható, hogy mind a két esetben ugyan azokat a tudáselemeket használjuk. Python esetében a kezdőbetű megállapításához elég a lista nulladik elemeként hivatkozni, míg Excelben algoritmust építünk. Tehát táblázatkezelőben történő megoldás során elősegíti az algoritmikus gondolkodást.

3. tankönyvi feladat

A róka libát lop a faluból. A libák súlyát – pontosabban tömegét – listában adjuk meg. A farkas a dűlőútnál várja a rókát, és a három kilónál nagyobb libákat elveszi – a piciket nagylelkűen otthagyja a rókának. Hány kiló libát ehét meg a róka?

Az előző feladat algoritmusát felhasználva írjuk meg azt az algoritmust, amelyik ezt a problémát oldja meg!

28. ábra Tankönyv 67. oldal 2. feladat [10]

	A	B	C
1	libak	rokanak	A rókának 6 kilónyi liba maradt
2	1	1	
3	5	HAMIS	
4	2	2	
5	3	3	
6	4	HAMIS	

29. ábra Excel megoldás (28. ábra)

képlet helye: B2:B6
 =HA(A2:A6<=3;A2:A6)

képlet helye: C1
 {"A rókának " & SZUM(HA(A2:A6<=3;A2:A6)) & " kilónyi liba maradt."}

5. képlet 3. tankönyvi feladat megoldása

```
libák = [1,5,2,3,4]

rókának_jut = 0
for liba in libák:
    if liba <= 3:
        rókának_jut += liba
print('A rókának', rókának_jut, 'kilónyi liba marad.')
A rókának 6 kilónyi liba marad.
```

30. ábra Python megoldás (28. ábra)

A tankönyv feladata alapján a rókának megmaradt libák tömegét kell meghatározni, az alapján, hogy az elvett libák 3 kilogrammnál nagyobbak. A A Spregoiban alkalmazott megoldás során összetett függvényt alkalmazunk, amelyet lépésről lépésre építünk fel az algoritmus alapján. Ez a módszer lehetővé teszi a tanulók számára, hogy jobban megértsék a képlet működését és az egyes lépéseket. A Python nyelvű megoldás során ugyanazt az algoritmust kódoljuk.

4. tankönyvi feladat

Az alábbi példákban a zenészes listánkkal dolgozunk. A példakódok mindig a hatodik sortól indulnak, mert az első négy sor (meg utána egy üres sor) csak a jelenléti ív adatait tartalmazza – a korábban ismertetett módon.

Írjuk ki, ha volt olyan nap, amikor mindenki jelen volt a próbán!

A feladat nehézsége, hogy ezúttal nem egy vonóst, hanem egy napot vizsgálunk. Úgy is mondhatjuk, hogy nem egy sor értékeit összegezzük, hanem egy oszlopét.

31. ábra Tankönyv 83. oldal 4. feladat [10]

	A	B	C	D	E	F	G	H
1		hétfő	kedd	szerda	csütörtök	péntek	jelenlét	mindig jelen
2	egyik hegedűs	1	1	1	1	1		5 Volt olyan zenész.
3	másik hegedűs	1	1	1	1	0		4
4	brácsás	1	1	0	0	0		2
5	csellós	0	1	1	1	1		4
6								

32. ábra Excel megoldás (31. ábra)

képlet helye: G2
=SZUM(B2:F2)

képlet helye: G3
=SZUM(B3:F3)

képlet helye: G4
=SZUM(B4:F4)

képlet helye: G5
=SZUM(B5:F5)

képlet helye: H2
{=HA(MAX(G2:G5)=5;"Volt olyan zenész."; "Nem volt olyan zenész")}

6. képlet 4. tankönyvi feladat megoldása

```
ív = [[1, 1, 1, 1, 1],
      [1, 1, 1, 1, 0],
      [1, 1, 0, 0, 0],
      [0, 1, 1, 1, 1]]

vonósok_jelenlétei = []
for vonós inív:
    vonós_jelenléte = sum (vonós)
    vonósok_jelenlétei.append(vonós_jelenléte)

for vonós in range (len(vonósok_jelenlétei)):
    if vonósok_jelenlétei[vonós]==5:
        print("Volt olyan zenész.")
        break
    else:
        print("Nem volt olyan zenész.")

    volt olyan zenész.
```

33. ábra Python megoldás (31. ábra)

Ebben a feladatban kétdimenziós listákkal kell dolgoznunk, miközben a megszámlálás algoritmusát, illetve elágazást kell alkalmaznunk. A megoldás során észrevehető, hogy mind a két esetben hasonló az algoritmust alkalmazunk, míg az Excel megoldás jobban hasonlít a klasszikus megoldásokra, addig a Python megoldásnál kétdimenziós listák használatával oldjuk meg a feladatot.

5. tankönyvi feladat

```
1. osztály = [['Noémi', 'l', 15, 'noemi@hipp.hopp'],
2.          ['Dezső', 'f', 17, 'dezso2@nyikk.nyekk'],
3.          ['Gizella', 'l', 16, 'gizi@pikk.pakk'],
4.          ['Edömér', 'f', 16, 'edo@itt.ott']]
5.
```

Mennyi az osztály átlagéletkora?

34. ábra Tankönyv 84. oldal 4. feladat [10]

	A	B	C	D	E
1	Osztály				Átlag életkor
2	Noémi	l	15	noemi@hipp.hopp	Az osztály átlagéletkora 16 év.
3	Dezső	f	17	dezso@nyikk.nyekk	
4	Gizella	l	16	gizi@pikk.pakk	
5	Edömér	f	16	edo@itt.ott	

35. ábra Excel megoldás (34. ábra)

képlet helye E2:

= "Az osztály átlagéletkora " & ÁTLAG(C2:C5) & " év."

7. képlet 5. tankönyvi feladat megoldása

```
osztály = [['Noémi', 'l', 15, 'noemi@hipp.hopp'],
           ['Dezső', 'f', 17, 'dezso@nyikk.nyekk'],
           ['Gizella', 'l', 16, 'gizi@pikk.pakk'],
           ['Edömér', 'f', 16, 'edo@itt.ott']]

összeg = 0
for tag in osztály:
    összeg += tag[2]

print('Az osztály átlagéletkora', összeg/len(osztály), 'év')
```

Az osztály átlagéletkora 16.0 év

36. ábra Python megoldás (34. ábra)

Az utolsó feladatban egy 4 tagú osztály átlagéletkorát kell meghatároznunk (egy teljesen értelmetlen feladat). A diákok adatait kétdimenziós lista adatszerkezetben tárolja a feladat. A Python megoldás során a listából kiolvastva kell a megfelelő algoritmust megalkotnunk. Sprego eszközöket kihasználva egy egyszerű megoldást adtunk az ÁTLAG() függvény hívásával.

5. Önálló munka részletezése

Kutatásunk során Mező-Nagy Dóra Hanna elemezte és megoldotta a feladatokat, valamint vizsgálta a Python programnyelv oktatásának hátrányait. Korponai Réka összeállította a kérdőívet, elvégezte a statisztikai elemzés és felelős volt a szakirodalmi áttekintésért.

6. Konklúzió

Az oktatásban az algoritmikus gondolkodás fejlesztése kiemelkedő jelentőséggel bír mind a tanulók, mind a tanárok számára. A fontosságát felismerve már a 3. osztálytól kezdve beépül a digitális kultúra tantárgyba. A NAT 2020 tankönyvek folyamatosan új programozási környezeteket és nyelveket vezetnek be. Kutatásunk során két különböző témakört vizsgáltunk: az

algoritmizálást táblázatkezelő környezetben és más programnyelv használatában. Az első témakör, a táblázatkezelés csak a 8. osztályban kezdődik, míg a második már az alsó tagozatban szerepel a tananyagban. Az általunk összeállított kérdőív alapján megállapítottuk, hogy a tanulóknak egyik környezetben sem sikerült megfelelő szintű algoritmizáló képességet kifejleszteniük. Ez azt jelenti, hogy hiába tanultak különböző programozási környezeteket és nyelveket, nem alakult ki bennük az a képesség, hogy önállóan tudjanak problémákat megoldani. Sok tanuló bizonytalan volt abban, hogyan kezdjen hozzá egy feladathoz. Ez arra utal, hogy az eszközök halmozása nem vezetett a megfelelő algoritmizáló képességek kialakításához.

A Python nyelv szintaktikájának elsajátítását előtérbe helyezése valójában korlátozhatja a diákok problémamegoldó képességeinek fejlődését. Bár fontos a nyelvi elemek megértése és alkalmazása, az igazi fejlődés a problémamegoldás terén általában akkor következik be, amikor a tanulók képesek a tanult eszközöket alkalmazni valós életbeli helyzetekben vagy komplex problémák megoldásában. A kérdőívünk eredményei alapján megállapítható, hogy sem a diákok, sem a tankönyv nem ismeri fel a fejezetek és témakörök közötti átjárhatóságot, annak ellenére, hogy a táblázatkezelőben felépített algoritmusok nem csak a programozás területén, hanem az adatbáziskezelésben is alkalmazhatóak. Az eredmények alapján úgy véljük, hogy változtatni kell a jelenlegi tankönyvek és oktatási módszereken. Fontos lenne az oktatás rugalmasabbá tétele és jól átgondolt, logikusan felépített tankönyvek bevezetése, amely segíthetne abban, hogy minden diák hatékonyan sajátítsa el az informatika (computer science, CS) alapjait.

További megállapításaink szerint a diákokat nem motiválják a tankönyvi feladatok. Ennek részben az az oka, hogy hiányzik a megfelelő ismeretük a feladatok megoldásához, valamint a tankönyv által felvetett problémák nem kapcsolódnak valós életbeli helyzetekhez. Ennek eredményeképpen a diákok nem érzik a feladatok fontosságát és relevanciáját.

Bebizonyosodott, hogy a Sprego használatával lényegesen hatékonyabban tanulható a táblázatkezelés, mint a felületorientált módszerekkel. Ezenkívül igazolást nyert, hogy a Sprego programozási megközelítésként és bevezető programozási nyelvként is sikeresen alkalmazható [9] [16] [17]. A Sprego segítségével könnyedén áttérhetünk az adatbáziskezelés SQL nyelvére, majd innen tovább haladhatunk a magasszintű programozási nyelvek irányába. Az alkalmazás tehát hatékony előkészítő lépést jelent az adatbáziskezelés és a programozás irányába, és lehetővé teszi a fokozatos fejlődést a kezdetektől a magasabb szintű programozási koncepciókig. A módszer segíthet a tanulóknak az algoritmizáló képességük fejlesztésében és a megfelelő

algoritmusok elsajátításában olyan környezetben, amely jobban alkalmazkodik a tanulók háttérismeretéhez, és a folyamatos ellenőrzés révén teljes képet kapnak az algoritmus működéséről. Sprego alkalmazásával, a tankönyvi 66 függvény tanítása helyett elegendő lenne 12 függvény elsajátítása. Az algoritmizáló készségek, képességek táblázatkezelőben történő fejlesztése, kevesebb függvény alkalmazásával, nemcsak időt spórolhatunk, de jobban fejleszthetjük a tanulók problémamegoldó gondolkodását egy olyan környezetben, amelyet a valós életben gyakrabban használnak. Ezáltal az óraszámokkal kapcsolatos problémák is megoldódhatnak.

7. Köszönetnyilvánítás

Szeretnénk kifejezni köszönetünket témavezetőnknek, Dr. Csernoch Mária tanárnőnek, aki a munkánk alatt nyújtott segítségével és szakmai iránymutatásával hozzájárult a Tudományos Diákköri Konferenciára való felkészülésünkhöz és a pályamunkánk elkészüléséhez. Külön köszönettel tartozunk a Debreceni Egyetem Kossuth Lajos Gyakorló Gimnáziuma és Általános Iskolájának, a Debreceni Egyetem Balásházy János Gyakorló Technikuma, Gimnáziuma és Kollégium intézményének, a Szent József Katolikus Elektronikai Technikum, Gimnázium és Kollégiumának, a Kisvárdai SZC Fehérgyarmati Petőfi Sándor Technikumának, valamint az iskolák tanulóinak, akik részt vettek a kutatásban és segítették munkánkat.

8. Melléklet

8.1. Kérdőív

8.1.1. Kérdőívben szereplő feladatok és megoldásuk eredeti formában

Lásd: https://bit.ly/kerdoiv_feladatai

8.1.2. Statisztikai elemzés a kérdőívben szereplő feladatokról

Lásd https://bit.ly/statisztika_algoritmizalas_feladatok

8.1.3. Középiskolában kitöltött kérdőív

Lásd: https://bit.ly/algoritmizalas_kerdoiv

8.2. Excel függvények megjelenése osztályonként

		8. o.	9. o.	10. o.	11. o.
1.	SZUM()	x	x	x	
2.	ÁTLAG()	x	x		
3.	MAX()	x	x		
4.	MIN()	x	x		
5.	NAGY()	x	x		
6.	KICSI()	x	x		
7.	DARAB()	x	x		
8.	DARAB2()	x	x		
9.	KEREKÍTÉS()		x		x
10.	SZÖVEG()		x		x
11.	HA()	x	x		x
12.	ÉS()		x		
13.	VAGY()		x		x
14.	MA()		x		
15.	BAL()		x		x
16.	JOBB()		x		x
17.	KÖZÉP()		x		x
18.	SZÖVEG.KERES()		x		x
19.	ÉV()		x		x
20.	HÓNAP()		x		x
21.	NAP()		x		x
22.	DÁTUM()		x		
23.	VÉL()		x		
24.	VÉLETLEN.KÖZÖTT()		x		
25.	MARADÉK()		x		
26.	HOL.VAN()		x	x	x
27.	INDEX()		x	x	x
28.	FKERES()		x		x
29.	DARABTELI()	x	x	x	
30.	DARABÜRES()		x		
31.	DARABHATÓBB()		x	x	x

		8. o.	9. o.	10. o.	11. o.
32.	SZUMHATÓBB()		x		x
33.	ÁTLAGHATÓBB()		x		x
34.	MAXHA()		x		x
35.	MINHA()		x		x
36.	GYÖK()			x	x
37.	HAHIBA()			x	
38.	HÉT.NAPJA()				x
39.	INT()				x
40.	ÓRA()				x
41.	PERCEK()				x
42.	MPERC()				x
43.	KISBETŰS()				x
44.	NAGYBETŰ()				x
45.	KIMETSZ()				x
46.	XKERES()				x
47.	HATVÁNY()				x
48.	LOG()				x
49.	SIN()				x
50.	COS()				x
51.	TAN()				x
52.	RADIÁN()				x
53.	ARCTAN()				x
54.	FOK()				x
55.	ARCSIN()				x
56.	ARCCOS()				x
57.	RÉSZLET()				x
58.	JBÉ()				x
59.	RÁTA()				x
60.	AB.DARAB()				x
61.	AB.DARAB2()				x
62.	AB.SZUM()				x

		8. o.	9. o.	10. o.	11. o.
63.	AB.ÁTLAG()				x
64.	AB.MIN()				x

		8. o.	9. o.	10. o.	11. o.
65.	AB.MAX()				x
66.	AB.MEZŐ()				x

9. Irodalomjegyzék

- [1] “ECDL / ICDL Word Processing Syllabus Version 5.0,” Testicdl, https://www.testicdl.com/images/syllabus-eecd/ECDL_ICDLWordProcessing1.pdf (accessed Jan. 27, 2024).
- [2] “Exam simulation ICDL module 3: Word processing,” Test ICDL ECDL & IT Certifications, <https://www.testicdl.com/module-3-word-processing.html> (accessed Jan. 27, 2024).
- [3] “Future of jobs report 2023 - World Economic Forum,” WEF_Future_of_Jobs_2023.pdf, https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf (utolsó megtekintés: 2024.01.30.)).
- [4] “Horror stories,” EuSpRIG2 (n.d.). European Spreadsheet Risk Interest Group. [Online]. Available: <https://eusprig.org/research-info/horror-stories/>. (accessed Jan. 3, 2024)
- [5] “Korábbi érettségi időszakok Feladatai és Javítási-értékelési útmutatói vizsgatárgyaként”, <https://dari.oktatas.hu/erettsegi.utmutato.index> (accessed Jan. 27, 2024). <https://doi.org/10.1016/j.sbspro.2010.12.256>
- [6] A. Abonyi-Tóth et al Digitális Kultúra 11. Budapest, Hungary: Oktatási Hivatal, 2022.
- [7] A. Abonyi-Tóth, C. Farkas, and P. Varga, Digitális Kultúra 7. Budapest, Hungary: Oktatási Hivatal, Budapest, 2022.
- [8] A. Abonyi-Tóth, C. Farkas, and P. Varga, Digitális Kultúra 8. Budapest, Hungary: Oktatási Hivatal, 2023.
- [9] A. Abonyi-Tóth, C. Farkas, N. Turzó-Sovák, and P. Varga, Digitális Kultúra 6. Budapest, Hungary: Oktatási Hivatal, Budapest, 2021.
- [10] A. Abonyi-Tóth, C. Farkas, Z. Reményi, and K. Jeneiné Horváth, Digitális Kultúra 10. Budapest, Hungary: Oktatási Hivatal, 2021.
- [11] A. Lénárd, A. Abonyi-Tóth, N. Turzó-Sovák, and P. Varga, Digitális Kultúra 5. Budapest, Hungary: Oktatási Hivatal, Budapest, 2020.
- [12] A. Lénárd, G. Sarbó, M. Tarné Éder, and N. Turzó-Sovák, Digital Culture 3. In Hungarian: Digitális Kultúra 3. Budapest, Hungary: Oktatási Hivatal, Budapest, 2022.

- [13] A. Lénárd, N. Turzó-Sovák, M. Tarné Éder, and G. Sarbó, Digitális Kultúra Tankönyv 4. Budapest, Hungary: Oktatási Hivatal, Budapest, 2023.
- [14] Benkő, L.; Benkő, Lné.; Tóth. B.; (1995). *Programozzunk C nyelven*. ComputerBooks Kiadói Kft , Budapest <http://fizweb.elte.hu/download/Fizika-BSc/C-programozas/Programozzunk-C-nyelven.pdf> (utolsó megtekintés: 2024.01.30.)
- [15] Biró, P., Csernoch, M.: The mathability of computer problem solving approaches. In: Peter Baranyi (szerk.) Proceedings of 6th IEEE Conference on Cognitive Infocommunications. Konferencia helye, ideje: Győr, 2015.10.19-2015.10.21. Győr: IEEE (2015) 111–114
- [16] Biró, P., Csernoch, M.: The mathability of spreadsheet tools. In: Peter Baranyi (szerk.) Proceedings of 6th IEEE Conference on Cognitive Infocommunications. Konferencia helye, ideje: Győr, 2015.10.19-2015.10.21. Győr: IEEE (2015) 105–110
- [17] Booth, S.: Learning to program: A phenomenographic perspective. Gothenburg, Sweden: Acta Universitatis Gothoburgensis (1992)
- [18] C. Angeli and N. Valanides, “Technological Pedagogical Content Knowledge - Exploring, developing, and assessing TPACK,” Springer US, 2015.
- [19] C. Angeli, “Teaching spreadsheets: A TPACK perspective,” *Improving Computer Science Education, Routledge, Taylor & Francis Group*, pp. 140–153, Feb. 2013. doi:10.4324/9780203078723-20
- [20] C. Angeli, N. Valanides, Technological Pedagogical Content Knowledge: Exploring, Developing, and Assessing TPACK, 2015th Edition. Springer, 2015.
- [21] C. Redecker, *European Framework for the Digital Competence of Educators. DigCompEdu*, Y. Punie, Ed. Luxembourg: Publications Office of the European Union, 2017.
- [22] C. Wolfram, “Evidence: Let’s promote not stifle innovation in Education,” Conrad Wolfram, May 22, 2015. [Online]. Available: <https://www.conradwolfram.com/writings/2015/5/21/role-of-evidence-in-education-innovation>
- [23] C. Wolfram, “The math(s) FIX: An education blueprint for the AI age,” Wolfram Media, Inc., 2020.
- [24] Core Python Programming, by Wesley J. Chun, Prentice Hall (December 15, 2000), 816 pages, ISBN: 0130260363
- [25] Csapó, G.; Csernoch, M.; Abari, K. Sprego: Case study on the effectiveness of teaching spreadsheet management with schema construction. *Educ. Inf. Technol.* 2020, 25, 1585-1605.
- [26] Csernoch, M. (2014). Programozás táblázatkezelő függvényekkel – Sprego, Műszaki Könyvkiadó, Budapest.

- [27] Csernoch, M. Biró P. Táblázatkezelés algoritmikus megközelítése https://www.researchgate.net/publication/284550673_TABLAZATKEZELES_ALGORITMI-KUS_MEGKOZELITESE (2013). (utolsó megtekintés: 2024.01.30.)
- [28] Csernoch, M., Biró, P., Máth, J., Abari, K.: *Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments*. Informatics in Education, Vilnius University, Vol. 14, No. 2. (2015)
- [29] Csernoch, M., Biró, P., Sprego helye az informatika tantervekben, (2015), DE IK
- [30] D. Kahneman, “Thinking, fast and slow,” Farrar, Straus and Giroux, 2011.
- [31] Dienes N., Gulácsi Á. Játékosított -Sprego programozás TDK dolgozat (2020)
- [32] E. Soloway, “Should we teach students to program?,” *Communications of the ACM*, vol. 36, no. 10, pp. 21–24, 1993. doi:10.1145/163430.164061
- [33] G. Csapó, K. Sebestyén, M. Csernoch, and K. Abari, “Case study: Developing long-term knowledge with Sprego,” *Education and Information Technologies*, vol. 26, no. 1, pp. 965–982, 2020. doi:10.1007/s10639-020-10295-0
- [34] G. Polya, *How to Solve It*. Princeton University Press, 1945.
- [35] Gál, T.:(2011). *Tehetséggondozás és informatika*. MFPI, Budapest. https://informatika.gtportal.eu/letoltes/doc/ModszertaniKiadvanyok_1_Gal_Digital_Small.pdf (utolsó megtekintés dátuma:2023.08.31)
- [36] GeoGebra. GeoGebra tools and resources. Teach and learn math in a smarter way. <https://www.geogebra.org/calculator>. (utolsó megtekintés: 2024.01.30.)
- [37] Gérard Swinnen Tanuljunk meg programozni Python nyelven 2005 (link: chrome-extension://efaidnbmnnnibpcajpcglclefind-mkaj/https://mek.oszk.hu/08400/08435/08435.pdf, —utolsó megtekintés ideje: 2024.01.31.)
- [38] Hack F; Fülöp F; Kugler S; Radnai Gy.; Urbán J.; Szabados L; Nemerkenyi A.; Balázs L.; Büki A. Négyjegyű függvény táblázatok, összefüggések és adatok, Nemzeti Tankönyvkiadó, Budapest (2004) <http://jerikogimnazium.hu/wp-content/uploads/2020/03/fuggvenytablazat.pdf>
- [39] J. Chen, D. Morris, and N. Mansour, “Science Teachers’ Beliefs: Perceptions of Efficacy and the Nature of Scientific Knowledge and Knowing,” in *International Handbook of Research on Teachers’ Beliefs*, 2014.
- [40] J. Hattie, “Visible learning for teachers: Maximizing Impact on learning,” Routledge, 2012.
- [41] J. M. Carroll and M. B. Rosson, *Paradox of the Active User*. San Jose etc.: IBM Thomas J. Watson Research Division, 1986.

- [42] J. M. Wing, “Computational thinking,” *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006. doi:10.1145/1118178.1118215
- [43] J. Sweller, P. Ayres, and S. Kalyuga, “Cognitive load theory,” Springer New York, 2011.
- [44] J. Vahrenhold, E. Nardelli, C. Pereira, G. Berry, M. E. Caspersen, J. Gal-Ezer, M. Kölling, A. McGettrick, and M. Westermeier, “Informatics Education in Europe: Are We All In The Same Boat? Report by The Committee on European Computing Education (CECE),” Association for Computing Machinery, 2017.
- [45] Király, R.: Funkcionális nyelvek. <https://gyires.inf.unideb.hu/KMITT/d02/index.html>
- [46] Learn to program using Python, by Alan Gauld, AddisonWesley Professional;, January 15 2001, 288 pages, ISBN: 0201709384
- [47] Learning Python, by Mark Lutz, David Ascher, 2nd Edition, O'Reilly, December 2003, 620 pages, ISBN : 0596002815
- [48] M. Ben-Ari and T. Yeshno, “Conceptual models of software artifacts,” *Interacting with Computers*, vol. 18, no. 6, pp. 1336–1350, 2006. doi:10.1016/j.intcom.2006.03.005
- [49] M. Ben-Ari, “Bricolage Forever!,” in *Psychology of Programming Interest Group*, 1999
- [50] M. Caspersen et al., “Informatics Reference Framework for School,” Retrieved from: <https://www.informaticsforall.org/wp-content/uploads/2022/03/Informatics-Reference-Framework-for-School-release-February-2022.pdf>, 2022.
- [51] M. Csernoch, J. Máth, and T. Nagy, “The interpretation of graphical information in word processing,” *Entropy*, vol. 24, no. 10, p. 1492, 2022. <https://doi.org/10.3390/e24101492>
- [52] M. Csernoch, K. Nagy, and T. Nagy, “The entropy of digital texts—the mathematical background of correctness,” *Entropy*, vol. 25, no. 2, p. 302, 2023. <https://doi.org/10.3390/e25020302>
- [53] M. Lodi and S. Martini, “Computational Thinking, Between Papert and Wing,” *Science & Education*, vol. 30, no. 4, pp. 883–908, Apr. 2021. doi:10.1007/s11191-021-00202-5
- [54] Magyar Közlöny, A Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 110/2012. (VI. 4.) Korm. rendelet módosításáról 290–446 (2020). Budapest. <https://magyarkozlony.hu/dokumentumok/3288b6548a740b9c8daf918a399a0bed1985db0f/letoltes>. (2024.03.11.)
- [55] Megabyte. C programozás kezdőknek. <https://megabyte.hu/kurzus/cprogkezd/3>. (utolsó megtekintés: 2024.01.30.)

- [56] Message, R.: *Programming for humans: a new paradigm for domain-specific languages*. Technical Report. UCAM-CL-TR-843. University of Cambridge, Computer Laboratory (2013)
- [57] Microsoft. Dynamic array formulas and spilled array behavior. <https://support.microsoft.com/en-us/office/dynamic-array-formulas-and-spilled-array-behavior-205c6b06-03ba-4151-89a1-87a7eb36e531>. (utolsó megtekintés: 2024.03.30.)
- [58] NAT2020. A Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 110/2012. (VI. 4.) Korm. rendelet módosításáról. 5/2020. (I. 31.) Korm. rendelet. (utolsó megtekintés: 2024.01.30.)
- [59] OFI. Kerettanterv a gimnáziumok 9–12. évfolyamára. (Utolsó megtekintés: 2023. 08. 31.) https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx
- [60] Osztatlan tanárképzés programja (2019). [https://mad-hatter.it.unideb.hu/portal/displayDocument/Szervezeti%20%C3%A1rak/Egy%C3%A9b%20szervezeti%20egys%C3%A9gek/Tan%C3%A1rk%C3%A9pz%C3%A9si%20K%C3%B6zpont/Dokumentum%C3%A1r/K%C3%A9pz%C3%A9sek/Osztatlan/K%C3%A9pz%C3%A9si%20programok/Az%20osztatlan%20tan%C3%A1rk%C3%A9pz%C3%A9s%20k%C3%A9pz%C3%A9si%20programja%20\(2019\).pdf](https://mad-hatter.it.unideb.hu/portal/displayDocument/Szervezeti%20%C3%A1rak/Egy%C3%A9b%20szervezeti%20egys%C3%A9gek/Tan%C3%A1rk%C3%A9pz%C3%A9si%20K%C3%B6zpont/Dokumentum%C3%A1r/K%C3%A9pz%C3%A9sek/Osztatlan/K%C3%A9pz%C3%A9si%20programok/Az%20osztatlan%20tan%C3%A1rk%C3%A9pz%C3%A9s%20k%C3%A9pz%C3%A9si%20programja%20(2019).pdf) (Letöltés dátuma 2023. 07. 29)
- [61] P. A. Kirschner, J. Sweller, and R. E. Clark, “Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching,” *Educational Psychologist*, vol. 41, no. 2, pp. 75–86, 2006. doi:10.1207/s15326985ep4102_1
- [62] P. Sestoft, *Spreadsheet Technology*. Copenhagen: IT University of Copenhagen, 2011.
- [63] P. Varga, K. H. Jeneiné, Z. Reményi, Cs. Farka, and I. Takács, *Digitális Kultúra 9*. Budapest, Hungary: Oktatási Hivatal, Budapest, 2020.
- [64] P. Warren, “Learning to program: spreadsheets, scripting and HCI,” *Proceedings of the Sixth Australasian Conference on Computing Education*, vol. 30, pp. 327–333, 2004.
- [65] Pedagógus képzés: B típusú gyakorlat. <https://tanarkepzes.unideb.hu/b-tipusu-gyakorlat>, (Legutóbbi frissítés: 2023. 02. 16. 13:
- [66] Pedagógus képzés: Egyéves összefüggő egyéni iskolai gyakorlat. <https://tanarkepzes.unideb.hu/egyveves-osszefuggo-egyeni-iskolai-gyakorlat>, (Legutóbbi frissítés: 2023. 05. 17. 12:53)
- [67] Pedagógus képzés: Közösségi pedagógiai gyakorlat. <https://tanarkepzes.unideb.hu/kozossegi-pedagogia-gyakorlat>, (Legutóbbi frissítés: 2023. 02. 16. 13:02)
- [68] *Programming Python*, by Mark Lutz, Second Edition, O'Reilly, March 2001, 1296 pages, ISBN : 0596000855

- [69] Python : How to program, by Harvey M. Deitel, et al, Prentice Hall; February 4, 2002, 1292 pages, ISBN: 0130923613
- [70] Python and Tkinter Programming, by John E. Grayson, Manning Publications; 1st edition (January, 2000), 688 pages, ISBN: 1884777813
- [71] Python cookbook, by Alex Martelli, Anna Martelli Ravenscroft, David Ascher, 2nd Edition, O'Reilly, 03/2005, 807 pages, ISBN: 0596007973
- [72] Python in a Nutshell, by Alex Martelli, 1st Edition, O'Reilly, March 2003, 654 pages, ISBN : 0596001886
- [73] Python Programming On Win32, by Mark Hammond, Andy Robinson, First Edition January 2000, 669 pages, ISBN: 1565926218
- [74] Python Scripting for Computational Science, by Hans P. Langtangen, Springer; 1st edition (September 20, 2004), 726 pages, ISBN: 3540435085
- [75] Python Standard Library, by Fredrik Lundh, O'Reilly, 05/2001, 281 pages, ISBN: 0596000960
- [76] R. Lister, “After the gold rush: toward sustainable scholarship in computing,” in Proceedings of the tenth conference on australasian computing education, 2008, vol. 78, pp. 3–17.
- [77] R. R. Panko, “The Cognitive Science of spreadsheet errors: Why thinking is bad,” in 2013 46th Hawaii International Conference on System Sciences. <https://doi.org/10.1109/hicss.2013.513>
- [78] S. F. Gibbs, G. Steel, and A. Kuiper, “Expectations of competency: The mismatch between employers’ and graduates’ views of end-user computing skills requirements in the Workplace,” Journal of Information Technology Education: Research, vol. 10, pp. 371–382, 2011. doi:10.28945/1531
- [79] S. Gibbs, G. Steel, and A. McKinnon, “A content validity approach to creating an end-user computer skill assessment tool,” Journal of Applied Computing and Information Technology, vol. 19, 2015.
- [80] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. New York, New York: New York: Basic Books, 1980.
- [81] Szlávi P.; Zsakó L. Módszeres programozás: Programozási tételek. ELTE Informatikai Kar, 2008.
- [82] T. Nagy, M. Csernoch, and P. Biró, “The comparison of students’ self-assessment, gender, and programming-oriented spreadsheet skills,” Education Sciences, vol. 11, no. 10, 2021. <https://doi.org/10.3390/educsci11100590>, Article number: 590
- [83] Vizsgaeredmények: Előző évi érettségi eredmények statisztikái <https://www.ket-szintu.hu/publicstat.php> (utolsó megtekintés: 2024.05.10.)

- [84] Vizsga tájékoztató: 203-2024-ben megrendezésre kerülő érettségi vizsgák és jelentkezők száma https://www.oktatas.hu/kozneveles/erettsegi/2024tavaszi_vizsgaidoszak/erettsegi_info_2024tavasz?itemNo=12&printMode=true (utolsó megtekintés: 2024.05.10.)
- [85] W. Gander, “Informatics in Schools? - Urgently Needed!” ECSS 2012, Barcelona, Nov. 21, 2012. [Online]. Available: <https://people.inf.ethz.ch/gander/talks/GanderECSS2012.pdf>. (utolsó megtekintés: 2024.01.30.)