Programozást tanító online felületek és kurzusok vizsgálata

Mahler-Lakó Viktória[1], Torma Hajnalka[2]

[1] viktoria.lako@gmail.com
ELTE IK
[2] hajnalka.torma@gmail.com
ELTE IK

Absztrakt. Az informatika illetve programozás tanításában nemzetközi szinten egyre elterjedtebb a különféle online felületek és kurzusok alkalmazása. Cikkünk célja, hogy ezek közül bemutassunk néhányat, amelyek a magyar közoktatásban is használhatóak lehetnének, megfelelő honosítás után. A vizsgálat során a következő szempontokat vettük figyelembe: Milyen a felület megjelenése, használhatósága? Milyen sorrendben és ütemben vezeti be az új fogalmakat? Milyen feladattípusok jelennek meg, ezekhez milyen általánosítható algoritmusminták kapcsolódnak? Milyen korosztály számára készült, milyen a feladatok nehézségi szintje? Vannak-e a felületben rejlő egyéb lehetőségek?

Kulcsszavak: programozás, e-learning, közoktatás

1. Bevezető

A cikkünk célja, hogy a külföldön már meghonosodott online tanulási környezetek közül bemutassunk néhányat, amelyek magyarra fordítás után (vagy akár anélkül) a hazai közoktatásban is alkalmazhatóak. Külföldön azért is van nagy szerepe ezeknek az online kurzusoknak, tananyagoknak, feladatbankoknak, mert képzett informatikatanár-hiánnyal küszködnek, ezért az is célja ezeknek az online kurzusoknak, hogy az informatikaképzettség nélküli tanárnak segítsenek felkészülni az órákra, feladatokat kapnak készen, így akár úgy is taníthatnak, hogy csak 1-2 órával vannak a diák előtt.[1]

Nálunk sokkal jobb a helyzet ebből a szempontból, hiszen vannak informatikatanári diplomával rendelkező pedagógusaink, viszont nekik is jól jön, ha van kész feladat, nem kell új programokat telepíteni a számítógépekre. Mindamellett a legtöbb iskolában kevés az informatika óra, tehát nagyon hasznos lehet egy olyan felület alkalmazása, amely tanár nélkül is működik, tehát a diák önállóan továbbképezheti magát, ha erre motivációt érez.

A vizsgálat tárgyául olyan kurzusokat, felületeket választottunk, amelyek ingyenesen elérhetőek és megfelelő nehézségi szintűek a közoktatásban: az általános iskola felső tagozatán vagy a középiskolában. Tehát az általunk választott oldalak kezdők számára készültek, olyanoknak, akik nem rendelkeznek előzetes programozási, kódolási ismeretekkel. A céljuk tehát nem a profi programozó kinevelése, hanem a programozás és algoritmizálás alapfogalmainak megtanítása, a számítógéppel való alkotás örömének megismertetése.

2. Vizsgálat

2.1. Gidget

A Gidget egy webes alkalmazás (debbugging game), amelyet Michael J. Lee kifejezetten programozási fogalmak tanítására tervezett. Az alkalmazás egy történet köré épül: egy gyárban ki-

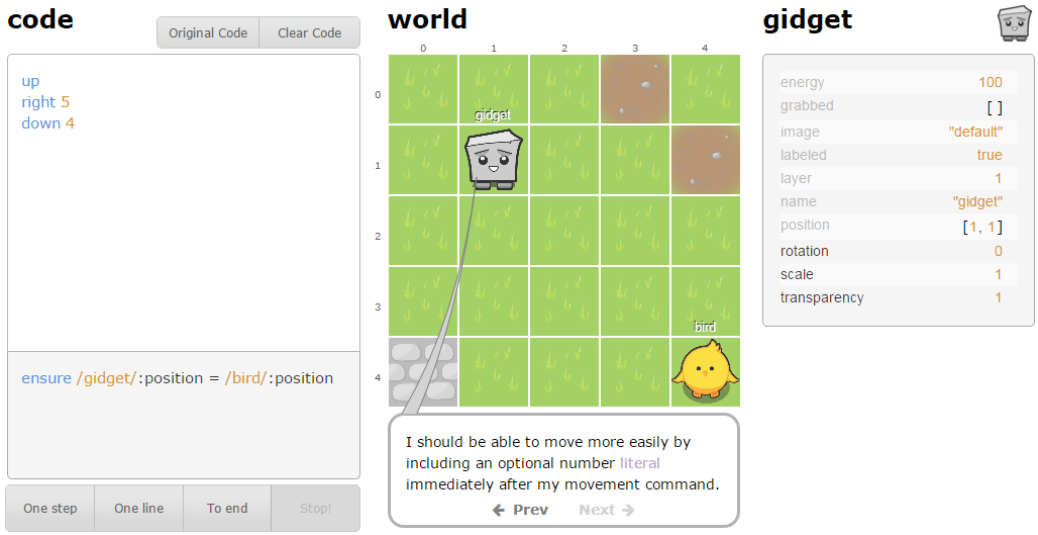
ömlik valamilyen vegyszer, és ennek a helyzetnek az orvoslása egy kis robot, Gidget feladata. A robot programok segítségével képes felismerni és megoldani különböző problémákat. Sajnos Gidget megsérült a szállítás közben, ezért csak olyan kódokat képes létrehozni, amelyek részben megoldják a feladatokban kitűzött célokat, de valamilyen szempontból hibásak. A játékos feladata, hogy segítse a robotot a kitűzött célok elérésében azáltal, hogy megkeresi a kódok hibáit, javítja a kódot, és futtatja a helyes programot.

Az alkalmazás egy imperatív Python-szerű programozási nyelvet használ, amely kifejezetten ehhez a játékhoz készült. A nyelv támogatja a dinamikus típuskezelést, logikai kifejezéseket és operátorokat, relációkat, matematikai műveleteket, objektumokat, függvényeket, valamint tartalmazza a játékbeli szereplők neveit, attribútumait és játékspecifikus utasításokat, amelyekkel a szereplőket irányíthatjuk.

A játék moduljainak befejezésével elérünk a Level Editor szintre, itt már magunk is készíthetünk Gidget feladatokat, játékokat.

2.1.1 Felület bemutatása

A Gidget alkalmazás felülete 3 egymás melletti részből áll (1. ábra): bal oldalon található a kódoló ablak, alatta olvashatóak a feladatban elérendő célok, amely alatt a futtatást és tesztelést indító gombok láthatók. Középen helyezkedik el a Gidget grafikus területe, a bejárandó pályával és különböző objektumokkal. A grafikus világ alatti szövegbuborékban pedig maga a Gidget figura részletezi a futtatási folyamat eseményeit. A jobb oldalon a játékban szereplő objektumok és azok paramétereinek aktuális állapotait követhetjük nyomon, illetve itt jelennek meg az ellenőrző kérdések a modulok végén.



1. ábra: A Gidget felülete

A Gidget figura a játék során segít nekünk a hibák keresésében és folyamatosan nyomon követni az utasítások végrehajtását. A jobb felső sarokban található könyv ikonra kattintva elérjük a nyelv összes utasítását.

2.1.2. Fogalmak bevezetése

A Gidget alkalmazásban egyre bonyolultabb problémák megoldásában kell segítenünk az elromlott chipet, így vezet be minket a programozási fogalmak világába az alábbi tematika szerint. A tematika bevezető programozási tankönyvek tematikájára, és a készítők tanítási tapasztalataira épül [2].

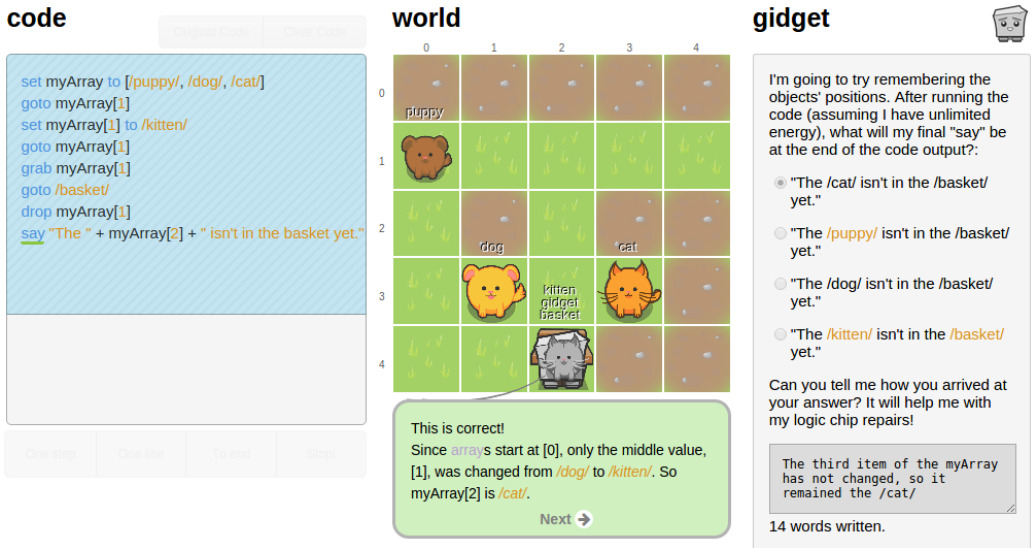
- Felület megismerése (programindítás, kódírás, hibakeresés), pálya bejárására alkalmas utasítások (right, left, up, down), objektumra hivatkozás (pl. /kitten/), objektumok mozgatása (grab, drop).
- Kételemű lista, lista első, illetve utolsó eleme, szintaktikai hibák keresése.
- Változó, tömb, értékadás, objektumtulajdonság.
- Objektum létrehozása és törlése, függvény megadása, függvényhívás, paraméter.
- Elágazás, logikai típus, logikai műveletek (és, vagy, nem).
- Ciklusok: feltételes, számlálós, lista elemeken végig lépkedő.
- Műveletek száma, hatékonyság.

2.1.3. Feladattípusok, algoritmus minták

A Gidget alkalmazásban a programozási fogalmakat 7 modulra osztották, az első hat modulon belül modulonként négy hibakereső-javító feladat és két ellenőrző kérdés található. Minden modul néhány egymással összefüggő programozási kulcsfogalomra, koncepcióra fókuszál (lásd előző pont).

A tesztkérdéseknek két fajtája van: feleletválasztós, pozíció megjelölés. Az első példában egy feleletválasztós kérdést látunk, minden esetben 4 válaszlehetőség közül kell az 1 helyes választ kiválasztanunk (2. ábra).

A másik kérdéstípus esetén a játéktér megfelelő celláját kell megjelölnünk. Mindkét kérdéstípus be van ágyazva a Gidget játék környezetébe, így nem zökkent ki a játék menetéből. A tesztkérdések megoldása közben a felhasználó nem fér hozzá az utasításszótárhoz, és nem tudja



2. ábra: Feleletválasztós kérdés a Gidgetben

futtatni a feladatban szereplő kódot. A kérdések tehát rákényszerítik a tesztalanyt, hogy fejben játssza le a kód utasításait, felidézze a leckében tanult fogalmakat. A zárt kérdések mellett, mindkét kérdéstípus esetén egy szövegmezőbe kell leírni a feladatmegoldás gondolatmenetét. Akár helyes, akár helytelen választ adtunk meg, a Gidget elvégzi a kitűzött utasításokat, valamint a helyes választ és a helyes gondolatmenetet is megmutatja [3].

A teszt előtti utolsó feladatban az összes korábbi fogalmat együtt kell alkalmazni a problémamegoldás során.

A 37 lecke tehát a legalapvetőbb programszerkezetekkel, programozási fogalmakkal foglalkozik, bonyolultabb algoritmusminták nem jelennek meg.

2.1.4. Korosztály, nehézség

A Gidget kiválóan alkalmas arra, hogy felkeltse a tanulók érdeklődését a kódolás iránt, tartalmilag és formailag megfelelő nehézségű az általános iskola felső tagozatába és a középiskolai korosztály számára is tanári segítséggel, útmutatással.

2.1.5. Extra szolgáltatások

A Level Editor szinten betekintést nyerünk a modulok feladatainak forráskódjába, és továbbfejleszthetjük azokat. Továbbá készíthetünk saját feladatokat a fogalmak gyakoroltatásához, új fogalmak bevezetéséhez, algoritmusminták bemutatásához.

2.1.6. Hol használják, tapasztalatok

Michael J. Lee több kutatás keretében vizsgálta a Gidget programozástanítási hatékonyságát.

Az egyik vizsgálat során megállapították, hogy a Gidget játékba épített értékelési eszközök segítik a fogalmak megértését és elsajátítását, pozitív hatással vannak a feladatmegoldás sebességére, valamint a felhasználókat ösztönzi a játék folytatására [4]. Egy másik kutatás során programozni nem tudó felnőttekkel próbáltatták ki a Gidget játékot, és azt tapasztalták, hogy jelentősen pozitív irányba változott a résztvevők kódoláshoz való hozzáállása [4].

A Gidgetet középiskolások körében is kipróbálták egy laborgyakorlat és két nyári tábor keretében. A tanulók 5 órában foglalkoztak a Gidget játék moduljaival (ha elakadtak, akkor segítséget kaptak), ezután a tanulók képessé váltak a megtanult fogalmak segítségével saját Gidget feladatok készítésére, azaz önállóan tudták alkalmazni a tanult ismereteket, programszerkezeteket [5].

2.1.7. Hiányosságok

Az alkalmazásban rengeteg lehetőség rejlik már most is, de a következő funkciók még profeszszionálisabbá tehetnék az alkalmazást:

- Tanárfelügyelet, felhasználók osztályokba szervezésének lehetőségének biztosítása.
- A szerkesztőfelület kiegészítése tesztkérdések készítésének lehetőségével.
- Lehetőség az általunk készített feladatok megosztására.

2.2.A CodeHS

CodeHS egy interaktív online tanulási környezet, amely 7 kurzust hirdet programozás témakörben. A felületet 2012-ben hozta létre a Stanford Egyetem két munkatársa Jeremy Keeshin és Zach Galant. Kurzusaik alapjául az egyetem bevezető programozási tárgyai szolgáltak. A felület alkalmas az önálló tanulás támogatására, illetve iskolai használatra is. Célja, hogy tananyagaival széles körben hozzáférhetővé tegye a számítógép-tudományos ismereteket.

A bevezető modulban Karel a kutya segítségével ismerkednek meg a programozási alapfogalmakkal. A későbbi modulok már magasabb szintű programozási ismereteket tanítanak JavaScript, Java, illetve HTML nyelveken grafikus alkalmazások, animációk, játékok készítését, illetve adatbázis-kezelési ismereteket sajátíthatnak el a felhasználók.

A kurzusok leckékre tagolódnak. A leckék videókból, tesztkérdésekből, példaprogramokból és megoldandó feladatokból állnak [6]. Mivel csak a bevezető kurzus férhető hozzá ingyenesen, illetve ennek szintje megfelel a célkorosztálynak, ezért a továbbiakban csak ezzel foglalkozunk. Az előfizetési díjtól függően, nem csak további tananyagokhoz férhetünk hozzá, hanem egy személyes tutort is kaphatunk, aki értékeli a feladatainkat és segítséget nyújt a tananyagok elsajátításában. A felületre iskolák is előfizethetnek 2500 dollárért évente, ekkor lehetőség van a diákok csoportba szervezésére, és a meglévő tananyagokat saját kurzusba is szervezhetik a tanárok [7]. Az előfizető iskolák tanárai ezen felül nem csak a kurzusok tematikáit kapják meg, hanem egy komplett módszertani anyagot is, amely segít a tananyagok hatékony osztálytermi alkalmazásában [8].

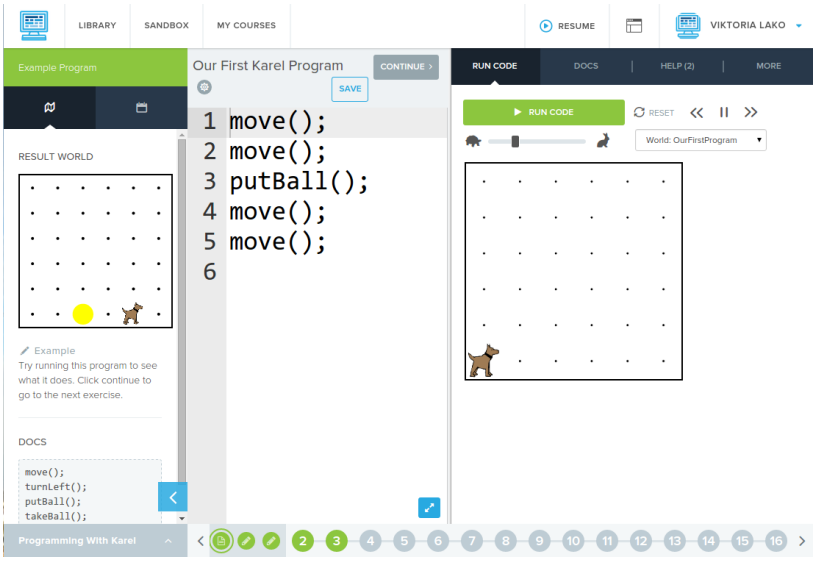
2.2.B Programming with Karel

A Programming with Karel a CodeHS felület bevezető kurzusa, amely 16 leckéből épül fel. A kurzus a Karel [9] programozási nyelvre épül, amelyet kifejezetten oktatási célra Richard E. Pattis fejlesztett ki, és alkalmazta is a programozás kurzusain a kaliforniai Stanford Egyetemen. A Karel programozási nyelvnek sok változata létezik, a CodeHS az eredeti változatot alkalmazta a bevezető tananyagban.

A környezet Karel a kutya köré épül, őt kell egyszerű utasításokkal (turnLeft, move, putBall, takeBall) irányítanunk egy négyzethálón. A feladatok során egy adott kiinduló állapotból kell eljuttatnunk Karelt a kijelölt célállapotba, miközben teniszlabdákat helyez el a négyzetrácson.

2.2.1. Felület bemutatása

A leckék mindegyike egy-egy videóval indul, amelyben a leckében elsajátítandó új fogalmat mutatják be egy példaprogram segítségével. Ezt követi egy-két tesztkérdés, majd megkapjuk a



3. ábra: A Programming with Karel környezet

videóban bemutatott kész példaprogramot, amelyet kipróbálhatunk és módosíthatunk. Ezután pedig a fogalomhoz kapcsolódó kódolási feladatot kapunk, ekkor önállóan kell elkészítenünk Karol számára azt a programot, amely a kiinduló állapotból a feladatban kitűzött célállapotba juttatja.

A Karel programozási felülete három függőleges területre tagolódik, a jobb oldalon láthatjuk a játékteret, ahol Karelt irányíthatjuk a négyzetrácson. A játéktér minden feladat esetében más-más kiindulási állapotban van, és itt követhetjük nyomon az utasítások végrehajtását. A játéktér felett található a programfuttatást indító gomb, az utasításokat itt is, mint a Gidget esetében végrehajthatjuk lépésenként, illetve állíthatjuk a programfutás sebességét, ezen kívül folyamatosan látjuk, hogy melyik kódrészt hajtja végre éppen a program. A középső terület a kódszerkesztő. A játéktér feletti gombokra kattintva a Karol nyelv utasításkészletét találjuk, kérdést tehetünk fel a tutoroknak, illetve megtekinthetjük a leckéhez kapcsolódó prezentációt, amely a videóban is szerepel.

A bal oldalon láthatjuk a feladat célállapotát, amelyet a kód segítségével kell kialakítanunk a játéktéren. A célállapot alatt pedig a feladatban használható utasításokat, illetve a feladathoz kapcsolódó elméleti összefoglalót olvashatjuk.

Az alsó sávon pedig azt követhetjük nyomon, hogy hányadik feladatnál tartunk a modulon belül, illetve visszaléphetünk egy-egy korábbi feladatra, videóra.

2.2.2. Fogalmak bevezetése

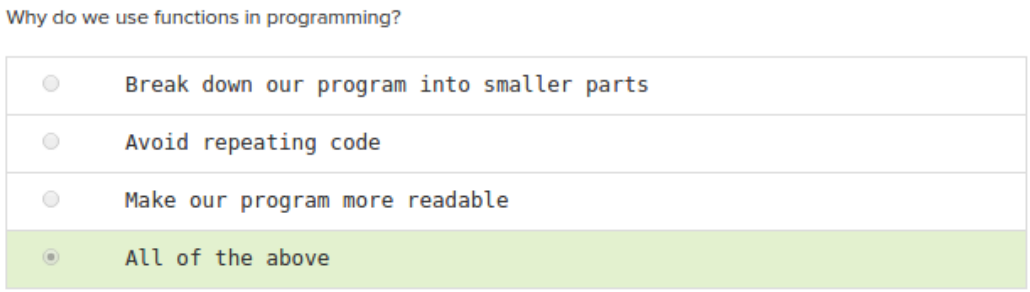
A Programming with Karel kurzus leckéi a következő tematika szerint épülnek fel [10]:

- A Karel programozási nyelv alap utasítás-készletének és a környezetének bemutatása. (move, turnLeft, putBall, takeBall)
- Tájékozódás Karel játékterében irányok, koordináták.
- Függvény fogalmának bevezetése, miért van szükség függvényekre, turnRight függvény készítése és használata.
- Függvény fogalmának elmélyítése, további függvények készítése.
- A start() függvény fogalma, miért van rá szükség.
- Felülről lefelé tervezés és a feladat részekre bontása.
- Kommentelés használata, funkciója, függvény előfeltétel és utófeltétel.
- SuperKarol (tud jobbra és hátra fordulni).
- Számlálós ciklus
- If utasítás és logikai kifejezések, logikai függvények.
- Elágazás
- Feltételes ciklus
- Utasítás szerkezetek (ciklus. elágazás) rendszerezése, egymásba ágyazása.
- Tesztelés
- Kód tagolása
- További gyakorló feladatokat és egy összefoglaló tesztet tartalmaz, ezek közül csak a teszthez férünk hozzá előfizetés nélkül.

2.2.3. Feladattípusok, algoritmus minták

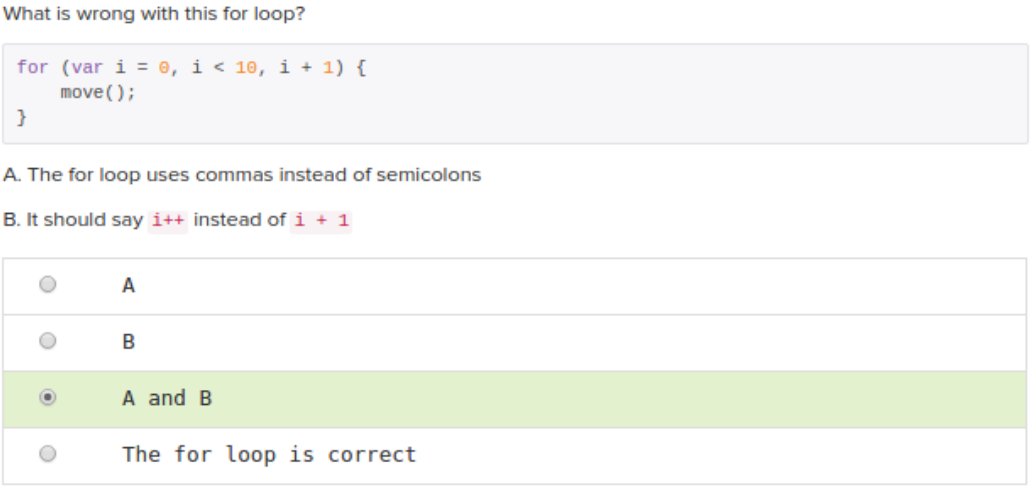
A leckékben tehát videókat, tesztkérdéseket és programozási feladatokat kínál a felhasználó számára a kurzus. A tesztek általában 4-5 kérdésből álló feleletválasztós kérdések, ahol az egyetlen helyes lehetőséget kell bejelölni.

A kérdések egy része valamilyen programozási fogalomra vonatkozik (4. ábra).



4. ábra: Feleletválasztó kérdés a Programming with Karol környezetben

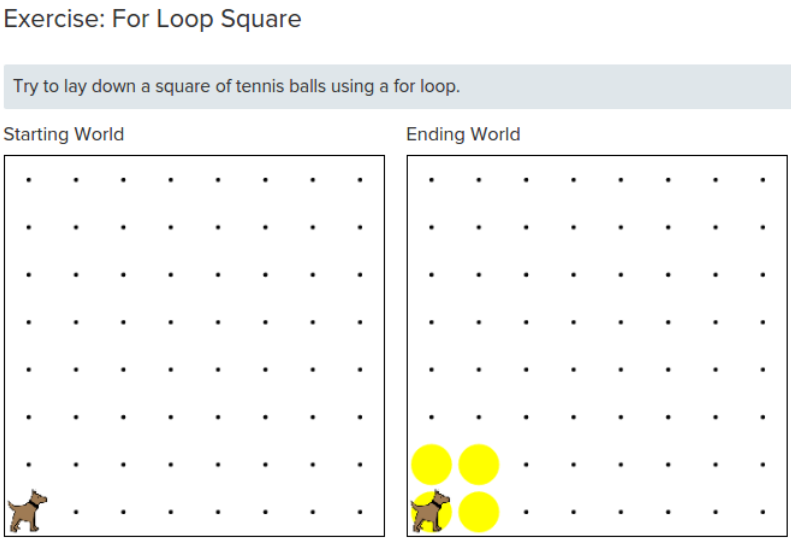
A másik részük pedig a nyelv szintaktikájának ismeretét kéri számon (5. ábra).



5. ábra: Szintaktikára vonatkozó kérdés a Programming with Karol környezetben

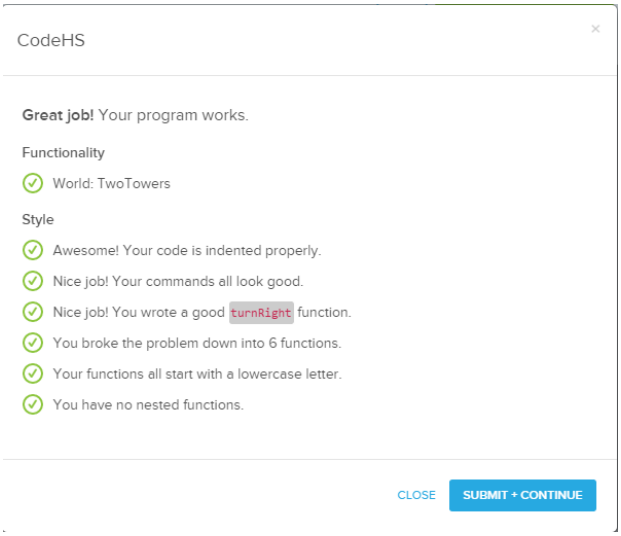
A leckék végén pedig programozási feladatok találhatóak, erre látunk példát a 6. ábrán. A feladathoz tehát kapunk egy rövid utasítást, illetve a feladat kiindulási és célállapotát.

Az elkészített kódot az első leckék esetében a feladatok a rendszer önmagában képes kiértékelni a 6. leckétől kezdve azonban már csak a kód tulajdonságainak egy részét képes ellenőrizni (ezekről visszajelzést is ad: lásd 7. ábra), ezért a kódot részletes egy tutor értékeli (amennyiben van előfizetésünk).



6. ábra: Kódolási feladat a Programming with Karol környezetben

A 16 lecke tehát az alapvető programszerkezetekkel, programozási fogalmakkal, és programozási paradigmákkal foglalkozik, bonyolultabb algoritmus minták nem jelennek meg a feladatokban.



7. ábra: Kód értékelő visszajelzés a Programming with Karol környezetben

2.2.4. Korosztály, nehézség

Nem igényel programozási, kódolási előismeretet [11]. Játékos keretben ismerteti a programozási fogalmakat, a magyarázatok megértéséhez absztrakt gondolkodás szükséges, ezért középiskolai bevezető kurzusnak ideális inkább, nem pedig az általános iskolás korosztálynak [12].

2.2.5. Extra szolgáltatások

A CodeHS környezet rengeteg lehetőséget kínál, de ezek nagy része csak előfizetés esetén érhető el.

2.2.6. Hol használják, tapasztalatok

Már 2012-ben kipróbálták a kurzust több középiskolában, többek között az oaklandi College Track-ben és East Palo Alto's Phoenix akadémián.

A CodeHS-t két másik oktatástechnológiai céggel együtt kiválasztották, hogy vegyen részt a 2013 Innovációs kihívásában, ahol élő műsorban versenyeztették egymással a szervezeteket, a kihívást a CodeHS nyerte, amelynek díjaként 75000 dollárt kaptak a Robin Hood alapítványtól [26].

2013 óta a Kódolás órája kihíváshoz kapcsolódóan több ingyenes kurzust is elérhetővé tettek.

2.2.7. Hiányosságok

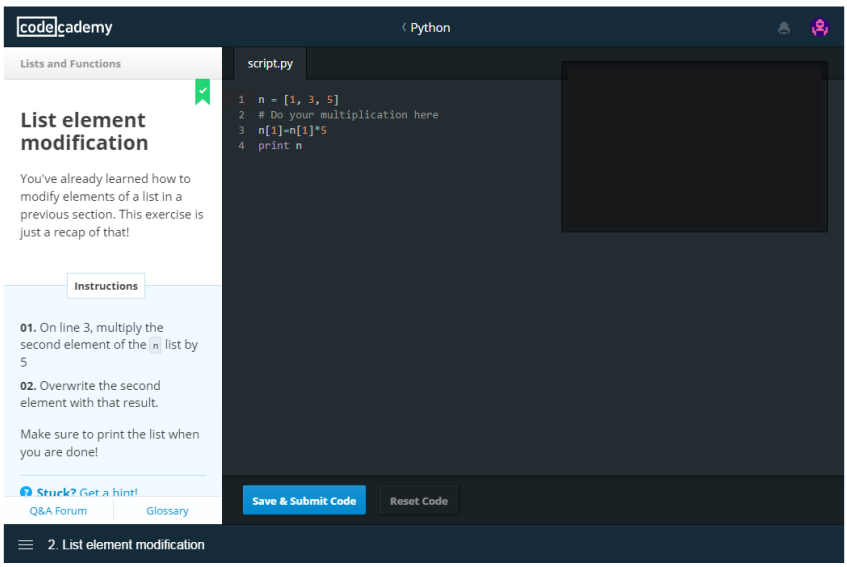
A CodeHS legnagyobb hátránya, hogy korlátozottak az ingyenes lehetőségek, valamint gyakran lassan töltődnek be egymás után a tananyagok.

2.3. Codecademy

Codecademy egy népszerű interaktív online környezet, amely több programozási nyelven biztosít programozás kurzusokat. Több tízmillió felhasználója, akik több 100 millió feladatot oldottak már meg a felületen. Jelen cikkünkben a Python nyelvhez kapcsolódó kezdő kurzussal foglalkozunk.

2.3.1 Felület bemutatása

A kurzus felülete két függőleges elválasztott területből áll: a keskenyebb bal oldali rész tetején láthatjuk a lecke címét, új nyelvi elem bevezetése esetén ezalatt található egy példakód. A példát követően, pedig a megoldandó feladathoz kapcsolódó instrukciókat találjuk, illetve itt jutunk segítséghez, ha elakadtunk. A szélesebb ablakrész a kódszerkesztő, amelynek jobb felső sarká-



8. ábra: A Codeacademy felülete

ban található a futtatási terület, tehát itt jelenik meg a program eredménye. A Save & Submit gombra kattintva a program végrehajtódik, itt nincs lehetőség a programfutás lassítására, lépésenkénti végrehajtására. Hiba esetén a futási ablakban kapunk hibaüzenetet.

2.3.2. Fogalmak bevezetése

A Codeacademy kurzus 21 modulra tagolódik, amelyek a következő tematikát követik [13]:

- A Codecademy felület megismerése, A Python nyelv alapvető terminológiájának megismerése: kiírás, változók (egész, valós, logikai), műveletek (+, -, *, /, maradék, hatvány), matematikai függvények, program egységek (behúzás), kommentelés.
- Éttermi számla kiszámítása – összetett feladat.
- Szöveg típus fogalma, megadása és műveletei, függvényei. Szöveg kiírása, szöveg konkatenáció, egyéb típusú érték szöveggé konvertálása, mikor van szükség szövegkonverzióra, változó értékek beágyazása a kiírandó szövegbe (% operátor).
- Dátum és idő függvénykönyvtár használata és tartalma, dátum és idő típus műveletei (év, hónap, nap, óra, perc, másodperc értékek kinyerése), változók értékeinek kiírása szövegbe ágyazással különböző formátumokban. Aktuális dátum meghatározása.
- Beolvasás, elágazás, logikai műveletek (és, vagy, nem), relációk (<, >, <=, >=, !=, ==), részszöveg (szemantikus ellenőrzés: szöveg-e?).
- PygLatin szójáték (beolvasás, elágazások, szövegműveletek, relációk) – összetett feladat.
- Függvények fogalma, paraméteres és paraméter nélküli függvények készítése, függvények meghívása. Speciális függvények használata a függvénykönyvtárból (min, max, abs).
- Nyaralási költségek kiszámítása függvénnyel – összetett feladat.
- Listák készítése és módosítása, lista elem hozzáadása (append függvény), listák szétvágása (részlisták), lista elemeken való végiglépkedés for ciklussal. Szótárak készítése és módosítása, mi a különbség a lista és a szótár között, szótár értékének módosítása, hozzáadása és törlése.
- Egy nap a bevásárló központban (raktárkészlet adatainak összegzése, fizetendő összeg kiszámítása bevásárló lista és raktárkészlet alapján) – összetett feladat.
- Tanár lesz a diákból (pontszámok rendszerezése, súlyozott átlag számítás, tanulónként, osztályonként) – összetett feladat.
- Listák mint függvényparaméterek, számlálós ciklus, lista generálás, listaelemek indexelése.
- Torpedó (korábbi fogalmak összekapcsolása, program tesztelés, tesztesetek), játék továbbfejlesztési lehetőségek – összetett feladat.
- Ciklusok
- Matematikai és szövegfüggvények - összetett feladat?
- Vizsga statisztika – összetett feladat.
- Magasabb szintű Python nyelvi elemek (pl. lambda kifejezések használata listák megadásához)
- Bitwise operatorok
- Osztályok
- Osztályok – összetett feladat.
- Fájlkezelés, bemeneti és kimeneti állományok használata.

2.3.3Feladattípusok, algoritmus minták

Az egyes leckék új programozási fogalmakat vezetnek be, a modulok végén viszont egy-egy összetett probléma megoldásán vezet minket végig a tananyag, amelyeket apró részekre bont.

	Feladat és példa	Instrukciók	Kód

1	**Your Own Store!**
	Okay—on to the core of our project.
	Congratulations! You are now the proud owner of your very own Codecademy brand supermarket.
	`animal_counts = {`
	`"ant": 3,`
	`"bear": 6,`
	`"crow": 2}`
	In the example above, we create a new dictionary called animal_counts with three entries. One of the entries has the key "ant" and the value 3.
		1. Create a new dictionary called prices using {} format like the example above.
		2. Put these values in your prices dictionary, in between the {}:
		`"banana": 4,`
		`"apple": 2,`
		`"orange": 1.5,`
		`"pear": 3`
		Yeah, this place is really expensive. (Your supermarket subsidizes the zoo from the last course.)
			`prices={`
			`"banana": 4,`
			`"apple": 2,`
			`"orange": 1.5,`
			`"pear": 3}`

2	**Investing in Stock**
	Good work! As a store manager, you're also in charge of keeping track of your stock/inventory.
		Create a stock dictionary with the values below.
		`"banana": 6`
		`"apple": 0`
		`"orange": 32`
		`"pear": 15`
			`prices={`
			`"banana": 4,`
			`"apple": 2,`
			`"orange": 1.5,`
			`"pear": 3}`
			`stock={`
			`"banana": 6,`
			`"apple": 0,`
			`"orange": 32,`
			`"pear": 15}`

3	**Keeping Track of the Produce**
	Now that you have all of your product info, you should print out all of your inventory information.
		Loop through each key in prices.
		Like the example above, for each key, print out the key along with its price and stock information. Print the answer in the following
			`prices={`
			`"banana": 4,`
			`"apple": 2,`
			`"orange": 1.5,`
			`"pear": 3}`

	once = {'a': 1, 'b': 2}
	twice = {'a': 2, 'b': 4}
	for key in once:
	print "Once: %s" % once[key]
	print "Twice: %s" % twice[key]
	In the above example, we create two dictionaries, once and twice, that have the same keys.
	Because we know that they have the same keys, we can loop through one dictionary and print values from both once and twice.
		format:
		apple
		price: 2
		stock: 0
		Like the example above, because you know that the prices and stock dictionary have the same keys, you can access the stock dictionary while you are looping through prices.
		When you're printing, you can use the syntax from the example above.
			stock={
			"banana": 6,
			"apple": 0,
			"orange": 32,
			"pear": 15}
			for key in prices:
			print key
			print "price: %s" % (prices[key])
			print "stock: %s" % (stock[key])

4	**Something of Value**
	For paperwork and accounting purposes, let's record the total value of your inventory. It's nice to know what we're worth!
		Let's determine how much money you would make if you sold all of your food.
		Create a variable called total and set it to zero.
		Loop through the prices dictionaries.
		For each key in prices, multiply the number in prices by the number in stock. Print that value into the console and then add it to total.
		Finally, outside your loop, print total.
			prices = {
			"banana" : 4,
			"apple" : 2,
			"orange" : 1.5,
			"pear" : 3}
			stock = {
			"banana" : 6,
			"apple" : 0,
			"orange" : 32,
			"pear" : 15}
			for key in prices:
			print key
			print "price: %s" % prices[key]
			print "stock: %s" % stock[key]
			total = 0
			for key in prices:
			total += prices[key]*stock[key]
			print total

2.3.4. Korosztály, nehézség

Középiskolástól bármilyen korosztály számára.

2.3.5. Extra szolgáltatások

A felületen hozzáférhető a tananyagokhoz kapcsolódó részletes tanmenet [14], valamint egy módszertani segédlet [15]. Rendelkezik tanárfelügyeleti rendszerrel, tehát a tanárként regisztrált felhasználók készíthetnek maguknak osztályt, ahol tananyagokat oszthatnak meg a diákokkal és nyomon követhetik az előrehaladásukat. Motivációként a feladatokért kitűzőket lehet gyűjteni.

2.3.6. Hol használják, tapasztalatok

Amerikában, az Egyesült Királyságban, valamint Spanyolországban és Argentínában is alkalmazzák a középiskolai informatika oktatásban [27].

2.3.7. Hiányosságok

Combéfis (2012) azt írja, hogy a Codecademy és az ehhez hasonló online alkalmazások fő céljaként egy-egy programozási nyelv megtanítása a cél, nem pedig az általános programozási tudásé és az algoritmusalkotási készségeké. A tanulónak kis kódrészleteket kell végrehajtania, és ebből felfogni a nyelv működését [16].

Friedman (2012) szerint a Codecademy nem tesz lehetővé semmilyen intelligens beszélgetést az oldalon található problémákról; nem kérdezi meg a felhasználót, mit gondol az adott utasításról, hogy hogyan kellene válaszolnia az utasításra, és nem ad lehetőséget a kétirányú kommunikációra arról, hogy miért pont úgy kellett megoldani a feladatot, ahogy a felhasználó megcsinálta. Ezzel együtt nem ösztönöz arra, hogy kérdéseket tegyen fel a felhasználó azzal kapcsolatban, hogy éppen mit is próbál megtanítani neki az adott lecke [17].

Wortham (2012) a Codecademy készítőivel készített interjút, melyben a készítői is elismerik, hogy nagy különbség van azon két dolog között, hogy valaki képes a kódokat „elolvasni”, és hogy jó programozó váljék belőle. Egy Codecademy-vel kódolni tanuló megkérdezett alany pedig úgy fogalmazott, hogy az oldalon tanulni olyan, mint amikor azt ígérik, hogy megtanulják megírni a nagy amerikai regényt egy kurzuson, de azzal kezdik, hogy mi az a főnév [18]. Ezzel szintén arra utalva, hogy a programozási nyelvnek csak az egységeit tanítja meg a Codecademy.

Lee (2013) kutatásaiból kiindulva úgy gondolja, hogy amennyiben a Codecademy kurzusok tartalmaznának olyan felmérő teszteket, amelyek beleillenek az oldal stílusába, akkor nemcsak a résztvevők tanulását tudnák fejleszteni, hanem lehetőségük lenne a tanulók valódi előrehaladásának vizsgálatára, és ezen információk birtokában fejleszthetik a tanterveiket, leckéiket [3].

Továbbá, bár a szolgáltatások között szerepel a tanári felügyelet, a tanár csak azt látja, hogy hányadik leckénél tart a diák, de azt nem, hogy miket rontott el, így nehezebb neki segítséget nyújtani.

2.4.A EDX

Az edX-et 2012-ben alapította a Harvard University és az MIT. Az edX kiváló minőségű kurzusokat kínál a világ legjobb egyetemeitől a világon bárki számára. Az edX kurzusok felületéül az Open edX nyílt forráskódú és ingyenes platform szolgál [19]. Az edX több mint ötven olyan kurzust kínál, amelyeket kifejezetten a középiskolás korosztálynak szánnak, mintegy előkészítve a felsőoktatási tanulmányaikat, elősegítve a tanulók pályaorientációját, több esetben egyetemi kreditszerzési lehetőséggel.

Az edX oldalán lévő kurzusokról általánosságban elmondható, hogy azok videókból, leírásokból, gyakorlatokból, ellenőrző kérdésekből és projektekből álló fejezetekből tevődnek össze. Az egyes fejezetek alfejezetekre és lapokra tagolódnak. A programozás tanulása során külső

eszközöket kell igénybe venni a kódok elkészítéséhez, jellemzően valamilyen felhő alapú, webböngészőben elérhető eszközt ajánlanak a kurzus instruktorai. A tanulók a saját kurzuson belüli előrehaladásukat követhetik nyomon, míg a kurzus létrehozója az összes tanuló elért, illetve folyamatban lévő eredményét láthatja a kurzushoz tartozó adminisztrációs felületen. A kitűzött feladatok és tesztek megoldásáért pontok járnak, amelyek a kurzus vezetője által megadott százalékban járulnak hozzá a kurzus végeredményéhez.

Kutatásunkban két ilyen középiskolásokat megcélzó programozást tanító kurzust vizsgáltunk meg, ezek a következők: Programming in Scratch és The Beauty and Joy of Computing.

2.4.B The Beauty and Joy of Computing

Az AP Computer Science Principles tantervéhez kapcsolódóan a számítógépes gondolkodást tűzi ki célul a Scratch alapú, könnyen elsajátítható Snap! programozási nyelv használatával, olyan témákat érintve, mint például a rekurzió, magasabb rendű függvények és kiszámíthatóság [20]. Tanulási célul a következőket tűzi ki: egy vagy több mobilalkalmazás készítése; eljárások tervezése (rajzolás, zeneszerkesztés, animálás, interakció a felhasználóval); ciklusok használata; eljárások az eljárásokban és ciklusok a ciklusokban; a számítógép-tudomány matematikájának felfedezése (véletlen, modulo, logikai operátorok); a technológia kultúrára és társadalomra gyakorolt hatásának elemzése; a programok és adatok szerkezetéről való gondolkodás elindítása; a feltételes utasítások ismerete; lokális változók használata; számítógépes innovációk elemzése és az adatvédelmi szempontok megbeszélése. A kurzus négy külön részből tevődik össze, melyek elvégzésére összesen 28 hét áll rendelkezésre, heti 4-5 munkával töltött órával számolva. A kurzus címéből adódóan pedig legfőbb céljuk, hogy mindenki jól érezze magát tanulás közben. A kurzus első része 2015. szeptemberében indult először, a második rész 2015. október végén, míg a kurzus második két része 2016-ban kerül először meghirdetésre.

2.4.1. Fogalmak bevezetése

A kurzus 4 külön egységből tevődik össze, és mivel a cikk írásának pillanatában még csak az első rész érhető el teljes terjedelmében, ezért csak erre korlátozódik a vizsgálatunk. Az első rész egyes fejezetei a következő számítógépes gondolkodási és programozási fogalmakat vezetik be:

- Absztrakció és a részletek elhagyása, általánosítás

  Gyakorlat: Egyszerű utasítások a Snap! nyelvben, vezérlő utasítások: ismétlés (repeat blokk), egymásba ágyazott ismétlések, végtelen ciklus (forever blokk), számlálós ciklus; különbség a repeat, forever és for blokkok között
- Függvények, adattípusok, értékhalmaz, értéktartomány, miért használjunk függvényeket

  Gyakorlat: Bemenet, blokkok (függvények) készítése, hibakeresés

  Logikai kifejezések, elágazások, matematikai operátorok, modulo

  Változók
- Számok (decimális, bináris, hexadecimális), konvertálás a különböző alapok között, digitális adatábrázolás, korlátok (túlcsordulás, kerekítési hiba)

  Gyakorlat: Feltételes kifejezések/elágazás, predikátumok, listák, könyvtár betöltése a programba

  Lokális változó, hibakeresés
- Adatvédelem, kreativitás

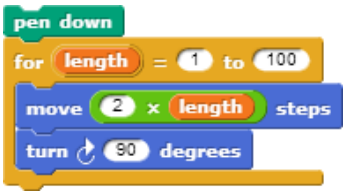
  Gyakorlat: Összetettebb programok írása, bemenet típusa, függvények összeillesztése
- Programozási megközelítések (functional, imperative, object-oriented, declarative)

2.4.2. Feladattípusok, algoritmus minták

A kurzus minden fejezetéhez tartozik laborgyakorlat, ahol a Snap! nyelv elemeit ismerheti meg a tanuló, miközben különböző feladatokat hajt végre. A feladatok megoldása során alkalmazásra kerülnek a fejezethez tartozó olvasmányban és videókban bevezetett programozási fogalmak is. A feladatoknak két szintje van: a laborgyakorlatban megadott lépések követése, próba útján történő gyakorlás, irányított feladatmegoldás (adott blokk kipróbálása, partnernek a blokk elmagyarázása, blokk módosítása, különböző megoldások vizsgálata/értékelése, hibakeresés), illetve az irányított feladatból kiindulva feladatok egyéni/párban történő megoldása. A következőkben minden fejezet esetén példát mutatunk a kétfajta feladattípusra (a és b pontok)

1. fejezet

a. Készítsd el a következő szkriptet (9. ábra), majd próbáld ki.



9. ábra

Kísérletezz! Módosítsd a forgatási szöget más számokra, mint például 92, 126 stb.

b. Készíts egy szkriptet, amely 12 szabályos sokszöget rajzol, úgy, hogy mindig eggyel növekszik az oldalak száma.

2. fejezet/ a.

i. Értelmezd a következő blokkot (10. ábra), és döntsd el, hogy a következők közül melyik nem lehetséges kimenet:

10. ábra

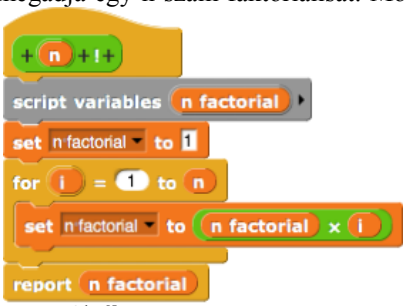
Válaszlehetőségek: 16, 8, 3, 12

ii. Készíts egy paritás-ellenőrző programot, amely választ egy számot 1 és 10 között, majd egy szereplő mondja meg, hogy a szám páros vagy páratlan (modulo használata).

b. Készíts egy programot, amely egy adott évről eldönti, hogy szökőév-e, ha akkor tekintünk egy évet szökőévnek, ha az évszám osztható 4-gyel.

3. fejezet /a.

i. A következő program (11.ábra) megadja egy n szám faktoriálisát. Módosítsd úgy a prog-



11. ábra

ramot, hogy működjön n=0 esetén is. Módosítsd úgy a programot, hogy ellenőrizze, hogy a megadott bemenet szám-e.

ii. Bemenetként adott egy angol szó, adjuk meg a többesszámát. (Mivel az angol nyelvben a szó végződése alapján többféle többesszám is lehet, ezért az egyes eseteket kezelni kell.)

b.

i. A többes számos program továbbfejlesztése arra az esetre, ha a felhasználó egy/bármennyi szóközt írt a bemenet végére.

ii. Készíts egy blokkot (függvényt), amely értékül azt adja meg, hogy egy adott egész számnak mennyi osztója van.

iii. Snap!-ben van “nagyobb mint” (<) operátor, “egyenlő” (=) operátor és “kisebb mint” (<) operátor, de nincs beépített “nagyobb vagy egyenlő” (>=) operátor (12. ábra). Készíts egy operátort, és ellenőrizd, hogy jól működik-e.

12. ábra

4. fejezet/ a.

i. Adott sorból álló téglafalat rajzoló eljárás írása: absztrakció, feladat lebontása lépésekre.

ii. Elemezz 3 függvényt, amelyek mindegyike azt vizsgálja, hogy egy adott év szökőév-e: Melyiket a legkönnyebb olvasni? Melyiket lenne a legkönnyebb újraírni fejből? Melyikben lenne legkönnyebb megtalálni egy hibát? Melyik a “legszebb”?

b.

i. Módosítsd a >= blokkot úgy, hogy csak számokat fogadjon el bemenetként.

ii. Készíts olyan blokkot (függvényt), amely 3 számot fogad bemenetül, és akkor ad igazat vissza, ha a 3 szám közül bármelyik kettő egyenlő.

5. fejezet: Szókitaláló játék készítése: egy megadott alapkódhoz kapcsolódó 4 függvény megírása a feladat, ahol adott a bemenet és kimenet elvárása. Szintén adott a program elvárt működése és egy példafutás.

6. Szabadon választott projektmunka egy másik résztvevővel együttműködve.

A laborfeladatok megoldására szolgáló szerkesztőfelület beágyazásra került a felületbe, és minden egyes feladat esetén egy külön ablakot jelöltek ki a szerzők a kód megírására. Továbbá lehetővé tették azt is, hogy a tanuló az általa írt kódról azonnali visszajelzést kapjon a Get Feedback gombra kattintva.

2.4.3. Korosztály, nehézség

A középiskolástól felfele bármilyen korosztály kezdő programozói számára. A kurzus nem tekinthető nehéznek, tempója megfelelő.

2.4.4. Extra szolgáltatások

A kurzus végén bizonyítványt kaphatunk 75% eredmény felett, amennyiben a meghirdetett időszakban végezzük a kurzust. Jövőbeli terv, hogy a kurzus SPOC (Small Private Onlince Course) formában is elérhető legyen az edX-en, ahol a tanárok a saját tanulóik előrehaladását nyomon követhetik [21].

2.4.5. Hol használják? Tapasztalatok

A Beauty and Joy of Computing kurzus a University of California, Berkeley (UCB) kurzusából alakult ki, amelyet először 2009-ben indított az egyetem, majd az AP Computer Science Principles tanterv kifejlesztésének egyik pilot helyszíneként folytatta az ezt követő években. A kurzus sikeressége [22] miatt ezt követően ez a kurzus lett a nem informatika szakosok számára az ajánlott informatikai kurzus az egyetemen [23], illetve elnyerte a National Science Foundation elismerését és pénzügyi támogatását is [24]. Az AP Computer Science Principles tanterve alapján a kurzus továbbfejlesztésre került az Education Development Centerrel való együttműködésben, a fejlesztés végét 2015 novemberére tűzték ki. Jelenleg New York középiskoláiban tesztelik, amelyet megelőzött több tanároknak tartott, egyenként 6 hetes műhelymunka. 2015-2018 között 100 tanár képzését tűzték ki célul [25]. A jelenleg edX-en futó kurzusról a tapasztalatok a jövőben lesznek elérhetőek, illetve az első hivatalos AP Computer Science Principles vizsgák után.

2.4.7. Hiányosságok

A Beauty and Joy of Computing kurzus magyar iskolákban történő felhasználása lényegében csak az angol nyelvi korlátok miatt kérdéses. A felépítése megfelel a jelen kor elvárásainak, mivel a számítógéptudomány kérdéseit, a határokat, a jövőt, úgy taglalja, hogy miközben igyekszik hangsúlyt fektetni a számítógépes gondolkodás fejlesztésére, az releváns legyen a tanuló és a társadalom számára is.

2.4.C EDX - Programming in Scratch

2.4.8. Fogalmak bevezetése

- Blokkok beszúrása, mozgatása, szkriptek készítése és ismétlés blokk
- Változók, változók módosítása a repeat ciklusban
- Koordináták és elágazás, valamint ciklus amíg
- Véletlenszám
- Iteratív fejlesztés (lebontás kisebb lépésekre), változó beolvasása, szöveg-összefűzés
- Eljárások, függvények (saját blokk készítése)

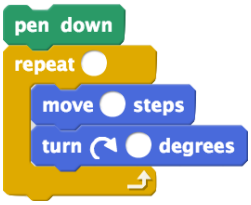
2.4.9. Feladattípusok, algoritmus minták

A kurzus minden fejezetének végén házi feladat és egy ellenőrző kérdéssor található. A házi feladat általában egy összetettebb feladat megoldásának lépésein vagy egy algoritmuson vezet végig kérdések segítségével. Ezt kiegészíti 4 projektmunka, amelyek önálló programozást kívánnak meg. Így a videókban és leírásokban lévő egyszerű kódok összetettebb programokká válhatnak, ahol a tanuló dönti el, hogy mennyi idő- és energia-befektetés mellett mennyire sajátítja el az elhangzottakat. Például a 6. fejezet házi feladata a klasszikus számkitalálós játék algoritmusát mutatja be, míg a projekt egy játék készítését kéri, amelyben csak annyi megkötést tesz, hogy a “broadcast” blokkot használja a tanuló az egyes szereplők “beszédének” elindítására.

Az ellenőrző feladatok a tanuló előrehaladását mérik: az aktuális fejezethez tartozó kérdések mellett mindig megjelennek az előző fejezetekben taglalt tudáselemek nehezített formában. Néhány példa kérdés az ellenőrző feladatok közül a fogalmak megjelenése szerint:

1. Ismétlés

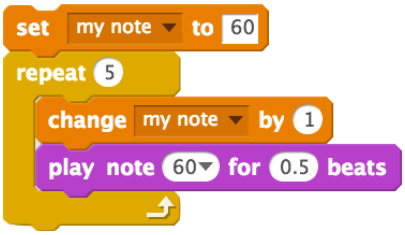
Példa: Miket írjunk az üres helyekre egy 9 oldalú alakzat rajzolásához? (13. ábra)



13. ábra

2. Változók, változók módosítása a repeat ciklusban

Példa: Milyen hangokat játszik le a szkript? (14. ábra)



14. ábra

3. Koordináták és elágazás, valamint ciklus amíg

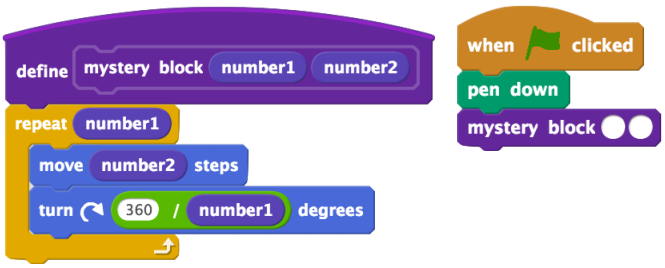
Példa: Ugyanazt teszi a három kódrészlet a zöld zászlóra kattintás után? (15. ábra)

15. ábra

4. Függvények

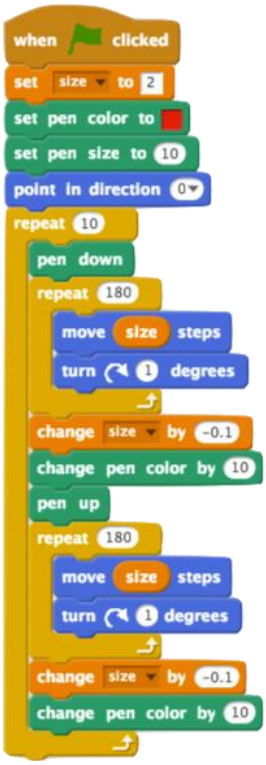
A mystery block blokkot szeretnénk használni egy 50 hosszú négyzet rajzolására, mit írjunk a “mystery block” első üres mezőjébe? (16. ábra)

5. Iteratív fejlesztés (lebontás kisebb lépésekre), változó beolvasása, szöveg-összefűzés



16. ábra

Mi lesz a size változó értéke a program futása után? (17. ábra)



17. ábra

2.4.10. Korosztály, nehézség

Leginkább az általános iskolás/kezdő programozó középiskolás korosztály számára.

2.4.11. Extra szolgáltatások

A kurzus végén bizonyítványt kaphatunk 60% eredmény felett

Fórumokban lehet beszélgetni a többi tanulóval

2.4.12. Hiányosságok

Akkor is megszerezhető a kurzus elvégzéséért járó bizonyítvány, ha a tanuló a végső nagy projektet nem csinálja meg, csak eltervezi, és a tervre pontot kap a többi tanulótárstól.

3. Összegzés

A nemzetközi kutatásokat figyelve azt látjuk, hogy egyre inkább van létjogosultsága az online tanulási környezeteknek az osztályteremben és azon kívül is. Segítségükkel a tanárok facilitátorokká válnak, könnyebben megvalósulhat a differenciált oktatás. Már korábban is készült vizsgálat hazánkban az online felületek programozás tanításában való bevezetésére, hiszen a jelenleg az oktatásban lévő diákok az Y és Z generáció tagjai, akiknek megváltoztak a tanulással kapcsolatos szokásaik, amelyekhez a gyakorlati pedagógia csak lassú ütemben képes igazodni. Ezzel szemben az online kurzusok jobban lépést tudnak tartani az új igényekkel. A Menyhárt László és Horváth Győző tavalyi cikkében a programozás tanításának komponenseire, ezek viszonyára és az online kurzusok felsőoktatásban való használatának létjogosultságára helyeződött a hangsúly [], a jelenlegi vizsgálat azonban kifejezetten a középiskolában bevezethető felületek elemzésére fókuszált, azon belül is arra, hogy milyen fogalmakat, milyen módszerekkel és feladattípusokat alkalmaznak az egyes kurzusokban.

Az általunk kiválasztott kurzusok közül a Gidget és a Programming with Karol egy-egy szereplő irányítására épül, ezt a szereplőt kell segítenünk különböző problémák megoldásában, ez közelebb hozza a matematika, illetve programozás iránt kevésbé érdeklődő diákok számára is a számítástudomány alapjait. Ezáltal alkalmas arra, hogy vonzóvá tegyük a programozást a tanulók számára.

A Gidget, a Programming with Karol, az edX Beauty and Joy of Computing kurzusa és a Codacademy Python bevezető kurzusa esetén is a felületbe építve kapunk egy programírási és futtatási környezetet, amely ráadásul ellenőrzi a futtatott kódot, és visszajelzést is ad annak helyességéről. Ez segíti a diákokat az önálló kódolásban, a tanárnak pedig a differenciálásban. Az első három kurzus akár az általános iskolai, akár középiskolai felhasználásra is alkalmas lehet megfelelő tanári módszerekkel támogatva, ezek lényege, hogy a programozási fogalmakat tanítsák meg. Ezzel szemben a Codeacademy Python modulja kódolni tanít, ezért ez utóbbi esetén szükség van a programozási alapfogalmak, és a probléma-megoldási módszerek más formában való megtanulására, ennek eszköze lehet a pedagógus, vagy akár a többi vizsgált felület, vagy ezek kombinációja.

Az edX-en található középiskolásoknak szólóként megjelölt kurzusok szintén alkalmasak lehetnek a közoktatásban történő felhasználásra, amennyiben ezeket a megfelelő tanulási módszert választva használjuk. A The Beauty and Joy of Computing kurzus elméletre és gyakorlatra osztott fejezetei különösen alkalmassá teszik a kurzust vegyes tanulási (blended learning) vagy tükrözött tanulási (flipped learning) módszerrel történő feldolgozásra.

A Programming in Scratch kurzus alapvetően egy bevezető kurzus azok számára, akik szeretnek önállóan próbálkozni, mivel a kurzus anyaga kitér ugyan az alap programozási fogalmakra, de a gyakorlást teljes egészében a tanulóra bízza, a megoldásokat külön nem ellenőrzi. Az egyes fejezetekben található tesztek és házi feladatok azonban fokozatosan nehezednek, így ezeket gyakorlás nélkül egy programozásban és algoritmizálásban járatlan tanuló nehezen oldja meg. A kurzus tanári "felügyelettel" kiegészítve viszont ideálisan működhet a közoktatásban.

Az eDX két kurzuásban az a hasonlóság, hogy a Scratch, illetve a Snap! nyelvet használva a játékos programozásra helyezik a hangsúlyt, szemben az olyan online programozási felületekkel, amelyek a programozást a kódolással, egy adott nyelv elsajátításával teszik egyenlővé.

A felületeket megvizsgálva azt a hipotézist fogalmaztuk meg, hogy a kiválasztott kurzusoknak lehet helye a magyar közoktatásban, például szakköri keretek között, a kiemelkedő tanulók differenciálásának segítésére. Továbbá nagy szerepük lehet akár az informatikatanár képzésben is, hiszen rengeteg feladat-ötlet, módszer található meg ezekben a kurzusokban, ráadásul folyamatosan frissülő és jól hozzáférhető formában. A felületek oktatásban való gyakorlati megvalósításának vizsgálatára egy következő kutatásban kerül majd sor.

Irodalom

1. M. Klawe: MOOCs Aim To Strengthen Computer Science And Physics Teaching In Middle And High Schools. Forbes (2015) http://www.forbes.com/sites/mariaklawe/2015/01/13/moocs-aim-to-strengthen-computer-science-and-physics-teaching-in-middle-and-high-schools/ (utoljára megtekintve: 2015.11.05.)
2. Aleven, V., Myers, E., Easterday, M., & Ogan, A. (2010). Toward a framework for the analysis and design of educational games. IEEE DIGITEL, 69-76.
3. Lee, M.J., Ko, A.J., and Kwan, I. (2013). In-Game Assessments Increase Novice Programmers' Engagement and Level Completion Speed. ACM International Computing Education Research Conference (ICER), San Diego, California, 153-160.
4. Charters, P., Lee, M.J., Ko, A.J., and Loksa, D. (2013). Challenging Stereotypes and Changing Attitudes: The Effect of a Brief Programming Encounter on Adults' Attitudes Toward Programming. ACM Technical Symposium on Computer Science Education (SIGCSE), Atlanta, Georgia, 653-658.
5. Lee, M.J., Bahmani, F., Kwan, I., Laferte, J., Charters, P., Horvath, A., Luor, F., Cao, J., Law, C., Beswetherick, M., Long, S., Burnett, M., and Ko, A.J. (2014). Principles of a Debugging-First Puzzle Game for Computing Education. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Melbourne, Australia, 57-64,
6. CodeHS weboldala: https://codehs.com/
7. J. Minor: CodeHS Review. PC Mag. (2015) http://www.pcmag.com/article2/0,2817,2493694,00.asp (utoljára megtekintve: 2015.11.05.)
8. CodeHS Professional Development Course. https://codehs.com/library/course/10/module/36 (utoljára megtekintve: 2015.11.05.)
9. Richard E. Pattis. Karel The Robot: A Gentle Introduction to the Art of Programming. John Wiley & Sons, 1981. ISBN 0-471-59725-2
10. CodeHS Student Learning Outcomes for Introduction to Computer Science. https://d14to6y4nub5k1.cloudfront.net/docs/CodeHSStudentLearningOutcomes.pdf (utoljára megtekintve: 2015.11.05.)
11. CodeHS FAQ. https://codehs.com/faq/ (utoljára megtekintve: 2015.11.05.)
12. CodeHS Website Review. Common Sense Media. https://www.commonsensemedia.org/website-reviews/codehs# (utoljára megtekintve: 2015.11.05.)
13. Codecademy Unit Overview. https://codecademy-school.s3.amazonaws.com/uk-curriculum/python.pdf (utoljára megtekintve: 2015.11.05.)
14. Codecademy Teaching Resources. https://www.codecademy.com/schools/curriculum
15. How to effectively use Codecademy in the classroom. Codecademy. https://codecademy-school.s3.amazonaws.com/Teacher_Training.pptx
16. S. Combéfis: Teaching Programming and Algorithm Design with Pythia, a Web-Based Learning Platform. Olympiads in Informatics, Vilnius (2012) Vol. 6, 31–43
17. J. Wortham: A Surge in Learning the Language of the Internet. The New York Times.(2012) http://cs.nyu.edu/courses/spring12/CSCI-UA.0002-001/NYTIMES_2012_0327.pdf (utoljára megtekintve: 2015.11.05.)

18. About Open edX. https://open.edx.org/about-open-edx
19. The Beauty and Joy of Computing. Syllabus: Course Information and Policies (2015) https://courses.edx.org/c4x/BerkeleyX/BJC.1x/asset/BJC.1xSyllabus.pdf (utoljára megtekintve: 2015.11.05.)
20. D. Garcia: Transforming K-12 CS. The Beauty and Joy of Computing (2014). http://www.cs.berkeley.edu/~ddgarcia/papers/2014-07-17-Garcia-BJC-CSTA.pdf (utoljára megtekintve: 2015.11.05.)
21. R. Shafer: Oh! The Beauty and Joy of Computing. Berkeley Engineering. (2009) http://engineering.berkeley.edu/2009/12/oh-beauty-and-joy-computing (utoljára megtekintve: 2015.11.05.)
22. A Berkeley Egyetem CS10 kurzusának éppen aktuális kurzusának honlapja: https://inst.eecs.berkeley.edu/~cs10/ (utoljára megtekintve: 2015.11.05.)
23. National Foundation Award a BJC részére: http://www.nsf.gov/awardsearch/showAward?AWD_ID=1138596 (utoljára megtekintve: 2015.11.05.)
24. Announcing Beauty and Joy of Computing: An Exciting New AP Computer Science Principles Course for NYC High Schools. Software Engineering Pilot. New York City Department of Education. http://sepnyc.org/apcs/#overview (utoljára megtekintve: 2015.11.05.)
25. L. Bradford: Codehs: bringing computer science to the classroom. Learn to code with me.(2014) http://learntocodewith.me/reviews/codehs (utoljára megtekintve: 2015.11.05.)
26. Az NBC közvetítése 2013 Innovációs kihívásáról: http://www.nbcnews.com/video/nbc-learn/53243821#53243821 (utoljára megtekintve: 2015.11.05.)
27. Issie Lapowsky: The Startup That's Bringing Coding to the World's Classrooms (2014.05.22) http://www.wired.com/2014/05/codecademy/ (utoljára megtekintve: 2015.11.05.)
28. Menyhárt László, Horváth Győző: Oktatási környezetek vizsgálata a programozás tanításához (2014), InfoDidact 2014 Konferencia, Zamárdi, 2014. november 20-21. http://people.inf.elte.hu/szlavi/InfoDidact14/Manuscripts/HGy_ML.pdf