Éveleji tanóratámogató segédanyag – IX. osztály

Informatika

9

magyar nyelven
a magyar tannyelvű iskolák/
osztályok számára,
9. osztály,
szakosított tanterv, elméleti
tagozat, reál szakirány,
természettudományok profil

SIGMA

1. Fejezet – A programozás alapjai
(1.1.; 1.3.; 1.5.; 2.1.; 2.3.; 2.5.; 3.1.; 3.3.; 3.5.; 4.1.; 4.3.; 4.5.; 5.1.; 5.3.; 5.5.; 6.1.; 6.5.)

1. Fejezet – A programozás alapjai

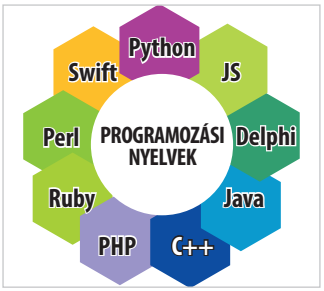
1. Lecke – Egy program elkészítésének szakaszai

*„Nem félek a számítógépektől. Nem félek a számítógépektől. A hiányuktól félek.” – **Isaac Asimov***

Informatika. Az adatok és az algoritmus

<def/> Az **informatika** olyan tudományterületek összessége, amelyek az információ számítógépes feldolgozásával foglalkozik, és alapvető szerepet játszik az *informatikai gondolkodás* fejlesztésében.

Megj. Az *informatikai* gondolkodás az ember azon képessége, amelynek segítségével utasítássorozatokkal és algoritmusokkal megvalósított problémamegoldási **módszereket** dolgoz ki. Fő jellemzői: ***felbontás***, ***általánosítás***, ***absztrakció***, ***algoritmizálás*** és ***értékelés***.



<def/> Az **algoritmus** véges számú, meghatározott sorrendben végrehajtott lépésből álló műveletsor, amely adatokat dolgoz fel. Az adatokat ***nevük***, ***típusuk*** és ***értékük*** azonosítja.

Megj. Az **adatokat** osztályozhatjuk: **típusuk** szerint (numerikus, logikai, karakterlánc típusú, illetve strukturált adatok), **felhasználásuk időpontja szerint** (bemeneti, közbenső/segéd és kimeneti adatok) és **értékük** szerint (változó és konstans adatok).

Kritérium	Kategóriák
típusuk szerint	numerikus, logikai, karakterlánc, strukturált
felhasználásuk időpontja szerint	bemeneti, közbenső/segéd, kimeneti
értékük szerint	változó és konstans

Jegyezd meg! Egy algoritmusnak végesnek, egyértelműnek és végrehajthatónak kell lennie, és az elfogadott bemeneti adatokra eredményt kell szolgáltatnia.

Egy program kidolgozásának szakaszai

Egy algoritmust megvalósító program elkészítéséhez a következő lépéseket kell végigjárni:

Lépés	Szerep
A probléma elemzése	bemeneti és kimeneti adatok, valamint a közöttük fennálló kapcsolatok azonosítása
Moduláris tervezés	a feladat alproblémákra bontása, a megoldás lépéseinek meghatározása és a szükséges adatok, s azok típusának kiválasztása
Implementálás	az algoritmus programozási nyelvre történő átültetése, annak szintaxisát betartva
Tesztelés és hibakeresés	a program futtatásával azonosítjuk és kijavítjuk a hibákat
Komplexitás elemzése/ optimalizálás	hatékonyság értékelése a felhasznált memória és a végrehajtási idő szempontjából

Támpontok ezen szakaszok alkalmazásához problémamegoldás során

Elemzés: a feladat követelményeinek megértése, a bemeneti/kimeneti adatok és korlátozások azonosítása.

Tervezés: vázlat pszeudokódban vagy folyamatábrában, a megfelelő adatszerkezetek kiválasztása és a probléma részproblémákra bontása.

Megvalósítás: a programkód világos és jól olvasható szerkesztése.

Tesztelés: a program ellenőrzése normál esetekkel, határesetekkel és hibás adatokkal.

Tesztek kidolgozásának szempontjai

A bemeneti adatok érvényesítéséhez: ellenőrizzük hogy a megadott adatok megfelelnek-e az elvárt típusnak, hogy az értékek a megengedett tartományba esnek-e és teszteljük határértékek és érvénytelen adatokkal.

Az algoritmus működésének ellenőrzéséhez: teszteljük az általános esetet, speciális eseteket, a kapott eredmények összehasonlítjuk a várt eredményekkel és ellenőrizzük az azonos adatokkal végzett ismételt futtatásokat.

Gyakori hibatípusok

Hibatípus	Leírás
szintaktikai hiba	a kód nem felel meg a programozási nyelv szabályainak
logikai hiba	a program lefut, de helytelen eredményt ad
futási hiba	a program végrehajtás közben leáll (például: nullával való osztás)

Szemléltetési példa

Pl.

Mihálynak informatikából ***n*** darab jegye van. Határozzuk meg a tanuló jegyeinek átlagát, majd állapítsuk meg, hogy átment-e vagy sem informatikából.

```
olvas n
ossz <- 0
minden i<-1,n vegezd el
    olvas jegy
    ossz <- ossz + jegy
media <- ossz / n
ha media >= 5 akkor
    kiir "atment"
kulonben
    kiir "nem ment at"
```

Jegyezd meg! A tesztelés és a hibakeresés nem opcionális lépések. Egy programot különböző adatokkal kell ellenőrizni, nem csupán egy alkalmas példával.

Feladatlap 1

Hozzatok létre egy *Feladatlap1* nevű mappát. Oldjátok meg az alábbi feladatokat, és írjátok a helyes válaszokat egy *valaszok.docx* nevű fájlba.

1) ***Válaszd ki a helyes választ:***

1a. Az informatikai gondolkodásmód a következőket foglalja magában:

a) Felbontás, általánosítás, absztrakció, algoritmizálás, értékelés;

b) Programozás, fordítás, tesztelés, szállítás;

c) Elemzés, összefoglalás, rögzítés, visszaadás; **d)** Pszeudokód, folyamatábra, Python, C++.

1b. Közbenső/segédadatok a következőek:

a) A billentyűzetről beolvasott adatok; **b)** A végső eredményként megjelenített adatok;

c) A program futása során a számításokhoz felhasznált adatok; **d)** A fájlokban tárolt adatok.

1c. Milyen típusú hiba okozza a program hirtelen leállását futás közben?

a) Szintaxis hiba; **b)** Logikai hiba; **c)** Futási hiba; **d)** Fordítási hiba.

2) ***Töltse ki:***

a) A programkészítés 5 lépése sorrendben: ..... → ..... → ..... → ..... → ..... .

b) Egy algoritmusnak ....., ..., ... kell lennie, és egy ..... eredményt kell előállítania.

c) A tesztelés során három adatkategóriával ellenőrizzük a programot: ....., ... és ..... .

3) ***Feladat***: Meg kell mérni a beteg hőmérsékletét, és meg kell jeleníteni, hogy lázas-e (hőmérséklet > 37,0) vagy sem.

Lépés	Bemeneti adatok	Közbenső adatok	Kimeneti adatok	Ellenőrzött feltétel	Normál adatokkal végzett teszt
Válasz					

4) ***Határozza meg az alábbi részletekben szereplő hibák típusát, és írja le, mi a hiba:***

a)
```
n = int(input('n = ')
print(n)
```
Hibatípus: ..... .
Leírás: ..... .

b)
```
n = int(input('n = '))
if n > 0
  print('pozitiv')
```
Hibatípus: ..... .
Leírás: ..... .

5) ***Értékelje az alábbi állítást:*** *Egy diák azt állítja, hogy a tesztelés csak akkor szükséges, ha a program futás közben hibát jelez. Egyébként, ha lefordítja, akkor helyes.*

a) Egyetértesz ezzel? Indokold a hibatípusokra hivatkozva. ..... .

b) Adj meg egy konkrét ellenpéldát: egy olyan programot, amely lefordítható és futtatható, de hibás eredményt ad.

Önértékelés! Mit tanultam?

[ISMERET] [MEGÉRTÉS] [ALKALMAZÁS] [ELEMZÉS] [ÉRTÉKELÉS] [ALKOTÁS]

ISMERET	1) Sorold fel táblázatban a programkészítés 5 lépését és azok szerepét.
MEGÉRTÉS	2) Magyarázd el a logikai hiba és a futási hiba közötti különbséget. Adj mindkettőre egy-egy példát.
ALKALMAZÁS	3) Alkalmazza a megvalósítás lépéseit a következő feladatra: „Olvasson be 3 jegyet, és jelenítse meg az átlagot, valamint azt, hogy a diák átment-e.” Töltse ki egy táblázatot a bemeneti, kimeneti és közbenső adatokkal, és írjon 3 tesztet.
ELEMZÉS	4) Elemezze: két csapat oldja meg ugyanazt a feladatot. Az A csapat közvetlenül kódot ír, a B csapat először elvégzi az elemzést és a pszeudokódot. Hasonlítsa össze a megközelítéseket a minőség és a hibakeresési idő szempontjából.
ÉRTÉKELÉS	5) Értékelje a következő tesztkészletet az „olvassa be n-t, és jelenítsd meg, hogy páros vagy páratlan-e” programhoz: test1: n=4, test2: n=7. Elégségesek ezek? Milyen hiányzó eseteket tud azonosítani?
ALKOTÁS	6) Tervezzen meg papíron egy megoldást az „iskolai átlagot számító számítógéphez, amely n jegyet olvas be, és megjeleníti az átlagot, a legmagasabb jegyet, valamint azt, hogy a diák kitűnő tanuló-e (átlag >= 9,5)”. Járjon végig minden lépést, beleértve 3 tesztet is.

Javítókulcs: 1p (hivatalból) + 1p (1) + 1p (2) + 1p (3) + 1p (4) + 2.5p (5) + 2.5p (6)

	A lecke végén tudom...	***Igen***	***Nem***	***Nem vagyok biztos***
1.	felsorolni és elmagyarázni a programkészítés 5 lépésének szerepét.			
2.	meghatározni egy feladat bemeneti, közbenső és kimeneti adatait.			
3.	osztályozni a hibákat (szintaktikai, logikai, futási) és mindegyikre adni egy példát.			
4.	kidolgozni a tesztelési kritériumokat: normál esetekkel, határesetekkel és hibás adatokkal.			
5.	alkalmazni a megvalósítás lépéseit egy egyszerű feladat esetében.			

6. Nehézségeim adódtak a következőkben: …

Ne felejtsétek el értékelni a viselkedéseteket és a tevékenységeteket az óra során!

A lecke végén így érzem magam: 

2. Lecke – Pszeudokód és programozási nyelvek

2.1. Az algoritmusok ábrázolása

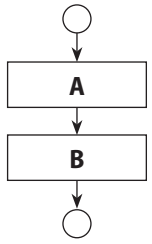
Minden ábrázolási módnak vannak előnyei és korlátai. Egy **algoritmus** leírható grafikus blokkokkal, pszeudokóddal vagy programozási nyelvvel.

Ábrázolási mód	Jellemzők	Felhasználás
folyamatábra	grafikus leírás blokkokkal és nyilakkal	az algoritmus lépéseinek szemléltetése
pszeudokód	a természetes nyelvhez közel álló szöveges leírás	az algoritmus megtervezése a kódolás előtt
programozási nyelv	szigorú szintaxisú, a számítógép által végrehajtott utasítások	az algoritmus tényleges megvalósítása

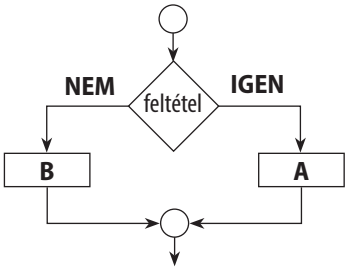
2.2. Folyamatábrák

<def/>

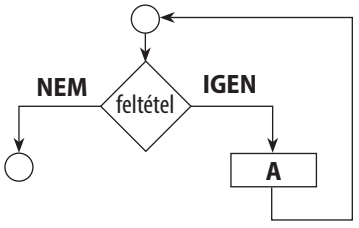
A **folyamatábra** az algoritmusok grafikus ábrázolással történő leírásának módszere. A vezérlési szerkezeteket blokkok írják le, alkalmazásuk sorrendjét pedig nyilak jelölik.



lineáris szerkezet



alternatív szerkezet



ismétlődő szerkezet

2.3. A pszeudokód nyelve

<def/>

A pszeudokód az algoritmusok köztes ábrázolási módja.A számítógép közvetlenül nem hajtja végre, de lehetővé teszi a lépések világos leírását egy programozási nyelven történő megvalósítás előtt.

Vezérlési szerkezetek pszeudokódban

Szerkezet	Forma pszeudokódban
lineáris	adatok beolvasása utasítás 1 utasítás 2 adatok kiírása
alternatív	ha <feltétel> akkor <utasítások 1> különben <utasítások 2> ha vége
ismétlődő	amíg <feltétel> teljesül, végezd <utasítások> amíg-ciklus vége
	i <- kezdeti_érték, végső_érték, lépésköz esetén végezd <utasítások> ciklus vége

Példa – Egy kód számjegyeinek összege

Mirunának van egy doboza, amelyet egy 9 számjegyből álló kóddal működő lakat zár. A doboz kinyitásához Mirunának meg kell határoznia a kód számjegyeinek összegét.

```
olvasd be kod-ot
osszeg <- 0
amig kod > 0 vegezd
    szamjegy <- kod % 10
    osszeg <- osszeg + szamjegy
    kod <- kod // 10
amig-ciklus vege
ird ki osszeg-et
```

Jegyezd meg! A pszeudokód nyelve lehetővé teszi az algoritmus magyarázatát anélkül, hogy egy adott programozási környezettől függene.

2.4. Programozási nyelvek

A programozási nyelv a programozó és a számítógép közötti kommunikáció eszköze. A programozási nyelvek segítségével az algoritmusok programokban valósíthatók meg, amelyekkel hatékonyan és világos módon oldható meg egy problémacsoport.

Típus	Jellemzők
értelmezővel működő nyelv	az utasítások sorban hajtódnak végre; a Python értelmezett nyelv
fordítóprogrammal működő nyelv	a programot végrehajtás előtt végrehajtható formává alakítják

Átírás pszeudokódból Pythonba

Pszeudokód	Python

`olvasd be n-t`
`osszeg ← 0`
`i ← 1, tol n-ig végezd`
`    osszeg ← osszeg + i`
`ciklus vege`
`írd ki osszeg-et`
	`n = int(input("n="))`
	`osszeg = 0`
	`for i in range(1, n + 1):`
	`    osszeg += i`
	`print(osszeg)`

Előnyök és korlátok

- A pszeudokód intuitív és könnyen követhető, de közvetlenül nem tesztelhető.
- A Python lehetővé teszi a program futtatását, de megköveteli a szintaxis és a behúzás szigorú betartását.

Jegyezd meg! Kódírás előtt hasznos az algoritmus lépéseit pszeudokódban megfogalmazni. Így a megvalósítás világosabbá és könnyebben tesztelhetővé válik.

Feladatlap 2

Hozzatok létre egy Feladatlap2 nevű mappát. Oldjátok meg az alábbi feladatokat, és írjátok a helyes válaszokat egy *valaszok.docx* nevű fájlba.

1) Írjatok pszeudokódban egy algoritmust, amely beolvas két számot, és kiírja a nagyobbikat.

2) Készítsétek el a folyamatábrát a következő algoritmushoz: ha x páros, akkor írja ki, hogy „páros", különben írja ki, hogy „páratlan".

3) Írjátok át Pythonba az alábbi pszeudokód-szerkezetet:

Pszeudokód	Python

`olvasd be n-t`
`osszeg <- 0`
`i <- 1-tol n-ig végezd`
`  olvasd be x-et`
`  osszeg <- osszeg + x`
`ciklus vege`
`írd ki osszeg-et`
	`n = int(input())`
	`osszeg  = 0`
	`for i in range(___, ___):`
	`  x = int(input())`
	`  _____________`
	`print(___)`

Önértékelés! Mit tanultam?

[ISMERET] [MEGÉRTÉS] [ALKALMAZÁS] [ELEMZÉS] [ÉRTÉKELÉS] [ALKOTÁS]

ISMERET	1) Soroljátok fel az algoritmus ábrázolásának 3 módját, és mindegyikhez adjatok meg egy előnyt és egy korlátot.
MEGÉRTÉS	2) Magyarázzátok el a különbséget az értelmezett és a fordított nyelv között. Soroljátok be a Python nyelvet a megfelelő kategóriába, és indokoljátok meg.
ALKALMAZÁS	3) Írjatok pszeudokódban, majd Pythonban egy algoritmust, amely beolvas n számot, és kiírja, hány közülük negatív.
ELEMZÉS	4) Elemezzétek: mikor célszerűbb for ciklust használni while ciklus helyett? Adjatok példát mindkét esetre.
ÉRTÉKELÉS	5) Egy osztálytárs azt állítja, hogy a pszeudokódnak semmi haszna, mert úgyis Pythonban kell megírni a kódot. Értékeljétek ezt az állítást, és érveljetek mellette vagy ellene.
ALKOTÁS	6) Tervezzetek megoldást egy „iskolai átlagkiszámítóhoz", amely beolvas n jegyet, és kiírja az átlagot, a legnagyobb jegyet, valamint azt, hogy a tanuló díjazott-e (átlag >= 9.5). Járjátok végig az összes lépést, beleértve 3 tesztet is.

Javítókulcs: 1p (hivatalból) + 1p (1) + 1p (2) + 2p (3) + 1p (4) + 2p (5) + 2p (6)

	A lecke végén tudom...	*Igen*	*Nem*	*Nem vagyok biztos*
1.	fel tudjam sorolni az algoritmus ábrázolási módjait, azok előnyeivel és korlátaival együtt.			
2.	át tudjak írni egy algoritmust pszeudokódból Pythonba.			
3.	meg tudjak írni és lépésről lépésre tudjak követni egy ismétlődő szerkezetet.			
4.	el tudjam magyarázni a különbséget az értelmezett és a fordított nyelv között.			

5. Nehézségeim voltak a következőkben: …

Ne felejtsétek el értékelni a viselkedéseteket és a tevékenységeteket az óra során!

A lecke végén így érzem magam:

3. Lecke - Változók, adattípusok és vezérlési szerkezetek

3.1. Egy Python-program felépítése

<def/> A **Python** értelmezett nyelv, amelyben az utasításokat az értelmező sorban egymás után hajtja végre. A legegyszerűbb program a **print()** függvény segítségével ír ki egy karakterláncot.

```
print("Jó napot")
print("Pythont tanulok")
```

A Python nyelv szókészlete

- a programok írásához használt ***karakterkészlet;***
- ***iazonosítók,*** amelyeket változók, konstansok, függvények vagy modulok elnevezésére használunk;
- különleges jelentésű ***kulcsszavak***: if, else, elif, for, while, continue, break, return;
- ***elválasztójelek:*** szóköz, behúzás, zárójelek, :, , vagy ;
- ***megjegyzések,*** amelyeket a # karakter segítségével vezetünk be.

⚠ **Figyelem!** Pythonban a behúzás nagyon fontos, mert elhatárolja az utasításblokkokat, és a szintaxis része.

3.2. Változók, beolvasás és kiírás

Egy változó létrehozásához Pythonban kiválasztunk egy azonosítót, majd értéket rendelünk hozzá.TA változó típusa automatikusan az érték alapján határozódik meg.

```
nev = input("Név: ")
kor = int(input("Életkorod: "))
print(f"eletkor, {nev}! (kor) éves vagy.")
```

3.3. Adattípusok

<def/> A Python nyelvben a következő adattípusok léteznek: **numerikus, logikai, karakterlánc, lista, szótár és tuple**. Ebben az egységben főként a numerikus értékek, a karakterláncok és a listák érdekelnek bennünket.

Adattípus	Példa	Megjegyzések
int	x = 7	egész szám
float	atlag = 8.5	valós szám
str	nev = "Anna"	karakterlánc
list	jegyek = [10, 8, 9]	rendezett gyűjtemény, indexelése 0-tól kezdődik

Az input() függvényről

Megj. Az input() függvénnyel beolvasott adatok alapértelmezés szerint karakterláncként tárolódnaSzámításokhoz olyan átalakításokat használunk, mint az `int()` és a `float()`.

Kifejezések és operátorok

A kifejezés operandusokból álló sorozat, amelyeket operátorok kapcsolnak össze. *Példa: 12 - 3 * 2.* Egy kifejezés kiértékelése az operátorok elsőbbségének figyelembevételével történik.

Kategória	Operátorok
aritmetikai	+, -, *, **, /, //, %
relációs	>, >=, <, <=, ==, !=
logikai	not, and, or
értékadás	=, +=, -=, *=, /=, //=, %=

Jegyezd meg! A % és // operátorok hasznosak egy szám számjegyeinek feldolgozásakor: az **n % 10** kinyeri az utolsó számjegyet, az **n // 10** pedig eltávolítja az utolsó számjegyet.

3.4. Vezérlési szerkezetek Pythonban

A vezérlési szerkezetek meghatározzák, hogy milyen sorrendben hajtódnak végre egy program utasításai: lineárisan, alternatív módon vagy ismétlődően.

Alternatív szerkezet

```
a = int(input("a = "))
b = int(input("b = "))
if a > b:
    print("az első gyerek nagyobb")
else:
    print(" a második gyerek nagyobb")
```

Ismétlődő szerkezet ismeretlen számú lépéssel

```
n = int(input("n = "))
cnt = 0
while n > 0:
    cnt += 1
    n //= 10
print(cnt)
```

Ismétlődő szerkezet ismert számú lépéssel

```
n = int(input("n = "))
szorzat  = 1
for i in range(1, n + 1):
    szorzat *= i
print(szorzat)
```

⚠ **Figyelem!** Az alternatív szerkezethez hasonlóan az ismétlődő szerkezetben lévő utasítások behúzása is nagyon fontos.

Feladatlap 3

Hozzatok létre egy *Feladatlap3* nevű mappát. Oldjátok meg az alábbi feladatokat, és írjátok a helyes válaszokat egy *valaszok.docx* nevű fájlba.

1) Mit ír ki a következő kód: x = "10"; y = 5; print(int(x) + y)??

2) Python nyelvet használva olvassátok be billentyűzetről egy téglalap hosszát és szélességét, majd megfelelő változók használatával írjátok ki a területét.

3) Írjatok Python-programot, amely beolvas egy számot, és kiírja, hogy pozitív, negatív vagy nulla.

4) Írjatok Python-programot, amely beolvassa **n** értékét, és for ciklus, valamint a range() függvény segítségével kiírja az **1-től n-ig** terjedő számok összegét..

Önértékelés! Mit tanultam?

[ISMERET] [MEGÉRTÉS] [ALKALMAZÁS] [ELEMZÉS] [ÉRTÉKELÉS] [ALKOTÁS]

ISMERET	1) Soroljátok fel a Python 4 alapvető adattípusát, és adjatok mindegyikre egy-egy példát.
MEGÉRTÉS	2) Magyarázzátok el a / és // operátorok közötti különbséget Pythonban. Számítsátok ki a 17 / 3 és a 17 // 3 értékét, és magyarázzátok meg az eredményeket.
ALKALMAZÁS	3) Írjatok Python-programot, amely beolvassa egy tanuló jegyét valós számként, és kiírja a minősítést: „Kiváló” (>= 9), „Jó” (>= 7), „Elégséges” (>= 5) vagy „Elégtelen”.
ELEMZÉS	4) Elemezzétek: miért okoz hibát az int('3.14'), de miért működik az int(3.14)? Magyarázzátok el a karakterláncból történő átalakítás és a float típusú érték csonkítása közötti különbséget.
ÉRTÉKELÉS	5) Értékeljétek: egy tanuló 3 jegy átlagát az (a + b + c) / 3 kifejezéssel számítja ki, int értékeket használva. Felmerülhet probléma? Teszteljétek a = 7, b = 8, c = 9 esetén.
ALKOTÁS	6) Tervezzetek egy „Számlakalkulátor” programot, amely beolvassa az egységárat, a mennyiséget és az áfakulcsot (%), majd kiszámítja és kiírja a részösszeget, az áfa értékét és a fizetendő végösszeget.

Javítókulcs: 1p (hivatalból) + 1p (1) + 1p (2) + 2p (3) + 1p (4) + 2p (5) + 2p (6)

	A lecke végén tudom...	*Igen*	*Nem*	*Nem vagyok biztos*
1.	fel tudjam ismerni egy Python-változó típusát.			
2.	tudjam használni az aritmetikai operátorokat, beleértve a //, % és ** operátorokat is.			
3.	tudjak adatokat átalakítani típusok között az int(), float(), str() függvényekkel.			
4.	helyesen behúzott alternatív szerkezeteket tudjak írni (if / elif / else).			
5.	különböző követelményekhez ismétlődő szerkezeteket tudjak írni.			

6. Nehézségeim voltak a következőkben: …

Ne felejtsétek el értékelni a viselkedéseteket és a tevékenységeteket az óra során!

A lecke végén így érzem magam:

4. Lecke - Alprogramok: definíció, paraméterek, lokális/globális változók

4.1. Mi az alprogram?

<def/>

Az alprogram (Pythonban függvény) egy névvel azonosított utasításblokk, amely egy konkrét feladatot old meg, és a program több részében is újra felhasználható.

Az alprogramok szerepe a kód moduláris szervezésének lehetővé tétele: egy összetett program kisebb alprogramokra bontható, amelyek mindegyike a probléma egy részét oldja meg.

Előny	Magyarázat
A kód újrafelhasználása	egy alprogramot csak egyszer írunk meg, és akárhányszor meghívható
Olvashatóság	a program könnyebben olvashatóvá és érthetővé válik
Tesztelés és hibakeresés	minden alprogram külön tesztelhető
Együttműködés	a csapattagok egymástól függetlenül dolgozhatnak különböző alprogramokon

4.2. Egy alprogram szerkezete Pythonban

Összetevő	Leírás
Fejrész (header)	az első sor, tartalmazza a def kulcsszót, a függvény nevét és a paramétereket
Törzs (body)	a fejrész alatt behúzással írt sorok; azokat az utasításokat tartalmazza, amelyek a hívás-kor végrehajtódnak
return	opcionális utasítás, amely visszaküldi az eredményt; return nélkül a függvény None értéket ad vissza

```
def fuggveny_neve(parameter1, parameter2):
    utasitas1
    utasitas2
    return eredmeny # opcionális
```

4.3. Paraméterek, argumentumok és az eredmények visszaadása

<def/>

A paraméterek olyan változók, amelyeken keresztül a függvény kívülről adatokat kap. A híváskor átadott értékeket argumentumoknak nevezzük. Ezek lehetővé teszik ugyanannak a függvénynek a használatát különböző értékekre.

Pl.

A minimum függvény

```
def minim(a, b):
    if a < b:
        return a
    else:
        return b

x = minim(3, 4)
y = minim(10, 7)
print(x, y)
```

1. A függvény paraméterei (bemeneti adatok)

A def minimum(a, b): sorban az a és b változók a függvény paramétereit jelentik. Ideiglenes lokális változókként viselkednek, amelyek a hívás pillanatában átadott értékeket veszik fel.

2. A függvény törzse és a döntési logika

A függvényben egy alternatív szerkezet hajtódik végre a paraméterekben tárolt értékek összehasonlítására.

3. A return utasítás:

A return kulcsszónak kettős szerepe van:

- Azonnal megszakítja a függvény végrehajtását.
- Visszaküldi a kapott értéket a program azon pontjára, ahol a függvényt meghívták..

A paraméterek alapértelmezett értékei

Vannak helyzetek, amikor elvárjuk, hogy egy paraméternek általában legyen egy megszokott értéke, de különleges esetekben módosítani szeretnénk azt.

```
def jegy_kiszamitasa(jegy, alappont = 1, maxim = 10):
    return min(jegy + alappont, maxim)

print(jegy_kiszamitasa(3))          # 4
print(jegy_kiszamitasa(10))         # 10
print(jegy_kiszamitasa(3, 2, 5))    # 5
```

Jegyezd meg! A return utasítás visszaküldi a kiszámított eredményt a program azon pontjára, ahonnan az alprogramot meghívták. A **return** végrehajtása után az alprogram azonnal befejeződik.

4.4. A végrehajtás mechanizmusa és a lokális/globális változók

Amikor a program egy függvényhíváshoz ér, megáll, átadja az argumentumok értékeit a függvény paramétereinek, lefuttatja az alprogram törzsét a kapott értékekkel, visszaadja az eredményt annak a programrésznek, ahol meghívták, majd a végrehajtás onnan folytatódik, ahol megszakadt.

Lépés	Mi történik
1	a főprogram végrehajtása megáll a hívás elérésekor
2	új memóriaterület jön létre az alprogram számára
3	az argumentumok értékei átadódnak a függvény paramétereinek
4	végrehajtódik a függvény törzse, és visszatér az eredmény
5	a főprogram végrehajtása onnan folytatódik, ahol megállt

4.5. Lokális változók vs. globális változók

Típus	Jellemzők
lokális	egy függvényen belül vannak definiálva; csak a függvény végrehajtása alatt léteznek; a paraméterek lokális változók
globális	minden függvényen kívül vannak definiálva; a függvényekből olvashatók, de a global kulcsszó nélkül nem módosíthatók

```
afa = 0.19  # globális változó
def ar_kiszamitasa(ar):
   vegosszeg = ar + ar * afa # a vegosszeg lokális változó
    return vegosszeg
print(ar_kiszamitasa(100))
```

⚠ **Figyelem!** A globális változók túlzott használata megnehezítheti a program hibakeresését. Ezért célszerűbb, ha a függvények paramétereken keresztül kapják meg az adatokat, és eredményeket adnak vissza.

Feladatlap 4

Hozzatok létre egy *Feladatlap4* nevű mappát. Oldjátok meg az alábbi feladatokat, és írjátok a helyes válaszokat egy *valaszok.docx* nevű fájlba.

1) Írjatok egy legnagyobb(a, b, c) függvényt, amely visszaadja három szám közül a legnagyobbat.

2) Az f(3, 5) hívásban a 3 és 5 értékek: **a)** Formális paraméterek. **b)** Globális változók. **c)** Argumentumok. **d)** Konstansok.

3) Határozzátok meg, mely változók lokálisak és melyek globálisak a mellékelt kódrészletben.

```
afa = 0.19
kedvezmeny = 0.10
def vegosszeg_kiszamitasa(ar, mennyiseg):
    reszosszeg = ar * mennyiseg
    vegosszeg = reszosszeg * (1 - kedvezmeny) * (1 + afa)
    return vegosszeg
eredmeny = vegosszeg_kiszamitasa(100, 3)
print(eredmeny)
```

Önértékelés! Mit tanultam?

[ISMERET] [MEGÉRTÉS] [ALKALMAZÁS] [ELEMZÉS] [ÉRTÉKELÉS] [ALKOTÁS]

ISMERET	1) Soroljátok fel egy Python-alprogram összetevőit (fejrész, törzs, return), és magyarázzátok el mindegyik szerepét.
MEGÉRTÉS	2) Magyarázzátok el a paraméter és az argumentum közötti különbséget. Miért hasznosak a paraméterek alapértelmezett értékei?
ALKALMAZÁS	3) Írjatok egy haromszog_terulet_kiszamitasa(alap, magassag) függvényt és egy teglalap_kerulet_kiszamitasa(a, b) függvényt. Hívjátok meg őket, és írjátok ki az eredményeket.
ELEMZÉS	4) Elemezzétek a globális változók használatának előnyeit és hátrányait a paramétereken keresztüli adatátadással szemben. Mikor elfogadható egy globális változó használata?
ÉRTÉKELÉS	5) Egy atlag_kiszamitasa(lista) függvényt üres listával, []-val hívunk meg. Mi történik? Értékeljétek a következő kód helyességét, és végezzetek javításokat, ha szükséges: if len(lista) == 0: return 0.
ALKOTÁS	6) Tervezzetek modulárisan egy „Számlakalkulátor kedvezménnyel” alkalmazást. Definiáljátok a következőket: reszosszeg_kiszamitasa(ar, mennyiseg) függvény, kedvezmeny_alkalmazasa(reszosszeg, szazalek) függvény, afa_kiszamitasa(osszeg, kulcs=19) függvény, vegosszeg(reszosszeg, kedvezmeny, afa) függvény.

Javítókulcs: 1p (hivatalból) + 1p (1) + 1p (2) + 1p (3) + 1p (4) + 1p (5) + 4p (6)

	A lecke végén tudom...	*Igen*	*Nem*	*Nem vagyok biztos*
1.	tudjak paraméterekkel és visszaadott értékkel rendelkező Python-függvényt definiálni és meghívni.			
2.	fel tudjam ismerni a lokális és globális változókat egy programban.			
3.	el tudjam magyarázni egy függvényhívás végrehajtási mechanizmusát.			
4.	tudjak alapértelmezett paraméterértékeket használni.			
5.	fel tudjam ismerni a különbséget a return és a print között egy függvényben.			

6. Nehézségeim voltak a következőkben: …

Ne felejtsétek el értékelni a viselkedéseteket és a tevékenységeteket az óra során!

A lecke végén így érzem magam:

ÚTMUTATÓK ÉS VÁLASZOK

1. Fejezet

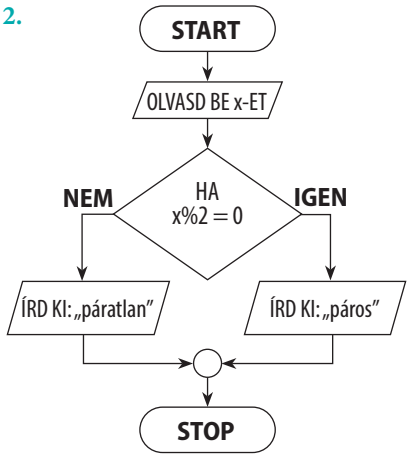
Feladatlap 1: 1. ***1a***-a); ***1b***-c); ***1c***-c), **2.** ***a*** – A probléma elemzése → Az algoritmus megtervezése → Megvalósítás (Kódolás / A program megírása) → Tesztelés és hibakeresés → Karbantartás (Dokumentálás és átadás); ***b*** – világos, véges, általános, eredmény; ***c*** – normál, határértéki, hibás. **3.** hőmérséklet, üzenet, nem azok, hőmérséklet > 37, t = 38 => „A beteg lázas". **4.** ***a*** – logikai hiba – a kiírt eredmény hibás lesz; ***b*** – szintaktikai hiba – hiányzik az akkor, illetve a ha vége a döntési szerkezetből. **5.** ***a*** – Nem, nem értek egyet. Az, hogy egy program lefordul és fut, csak azt bizonyítja, hogy nincsenek benne **szintaktikai** hibák. A tesztelés azonban feltétlenül szükséges, mert a program tartalmazhat **logikai** hibákat; ***b*** – példa a 4a pontban szereplő program hibásan számítja ki a számtani átlagot.

Feladatlap 2:

1.

```
algoritmus ket_szam_maximuma
   egesz a, b, max
   olvasd_be a, b
   ha a > b akkor
      max ← a
   kulonben
      max ← b
   ha_vege
   ird_ki max
algoritmus_vege
```

2.



3. 1, n, osszeg += x

Feladatlap 3: 1. 15

2.

```
hossz = float(input("Add meg a hosszúságot: "))
szelesseg = float(input("Add meg a szélességet: "))
terulet = hossz * szelesseg
print("A terület:", terulet)
```

3.

```
n = int(input("Add meg n értékét: "))
osszeg = 0
for i in range(1, n + 1):
    osszeg = osszeg + i
print("Az 1-től", n, "-ig terjedő számok
összege:", osszeg)
```

Feladatlap 4:

1.

```
def legnagyobb(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c
```

2. c). **3.** **globálisak**: afa, kedvezmeny, eredmeny; **lokálisak**: ar, mennyiseg, reszosszeg, vegosszeg.