Python

2. gyakorlat

Strohmayer Ádám

Agenda

- Követelmények
- Python alapok
- Változók
- Adattípusok
- Alap műveletek

Követelmények

- >= 50% : 2
- >= 64% : 3
- >= 78% : 4
- >= 90% : 5

2 beadandó : 20-20%

2 elméleti ZH : 20-20%

Óra eleji kvízek : 20%

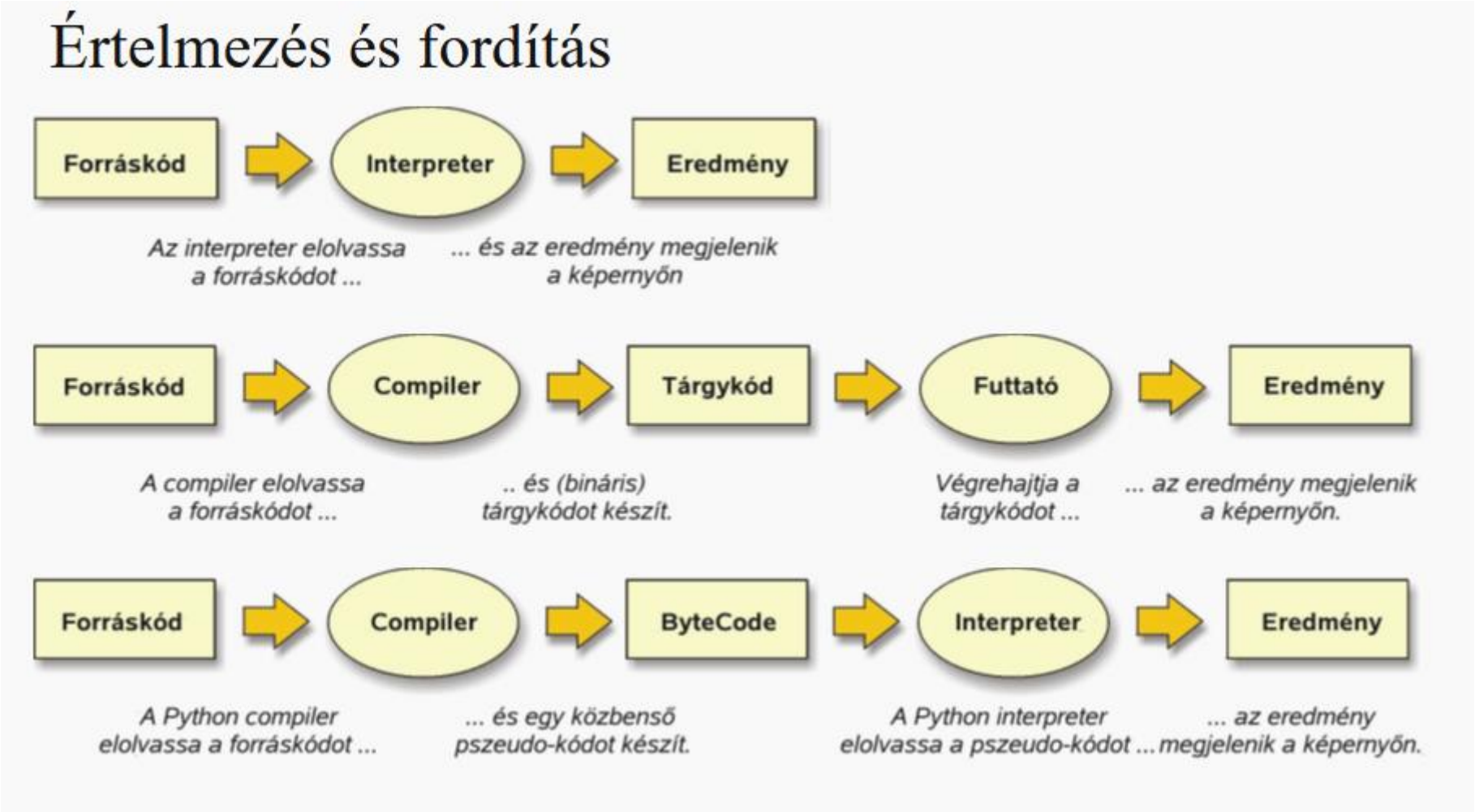
minimum 70%-os
kvízek darabszáma alapján!

Szóbeli beadandóvédés!

Python

- Multiparadigmás
- Szkriptnyelv
- Interpretált
- Dinamikusan típusos
- Multiplatform
- PyPI (Python Package Index)

Python kód értelmezése



PVM

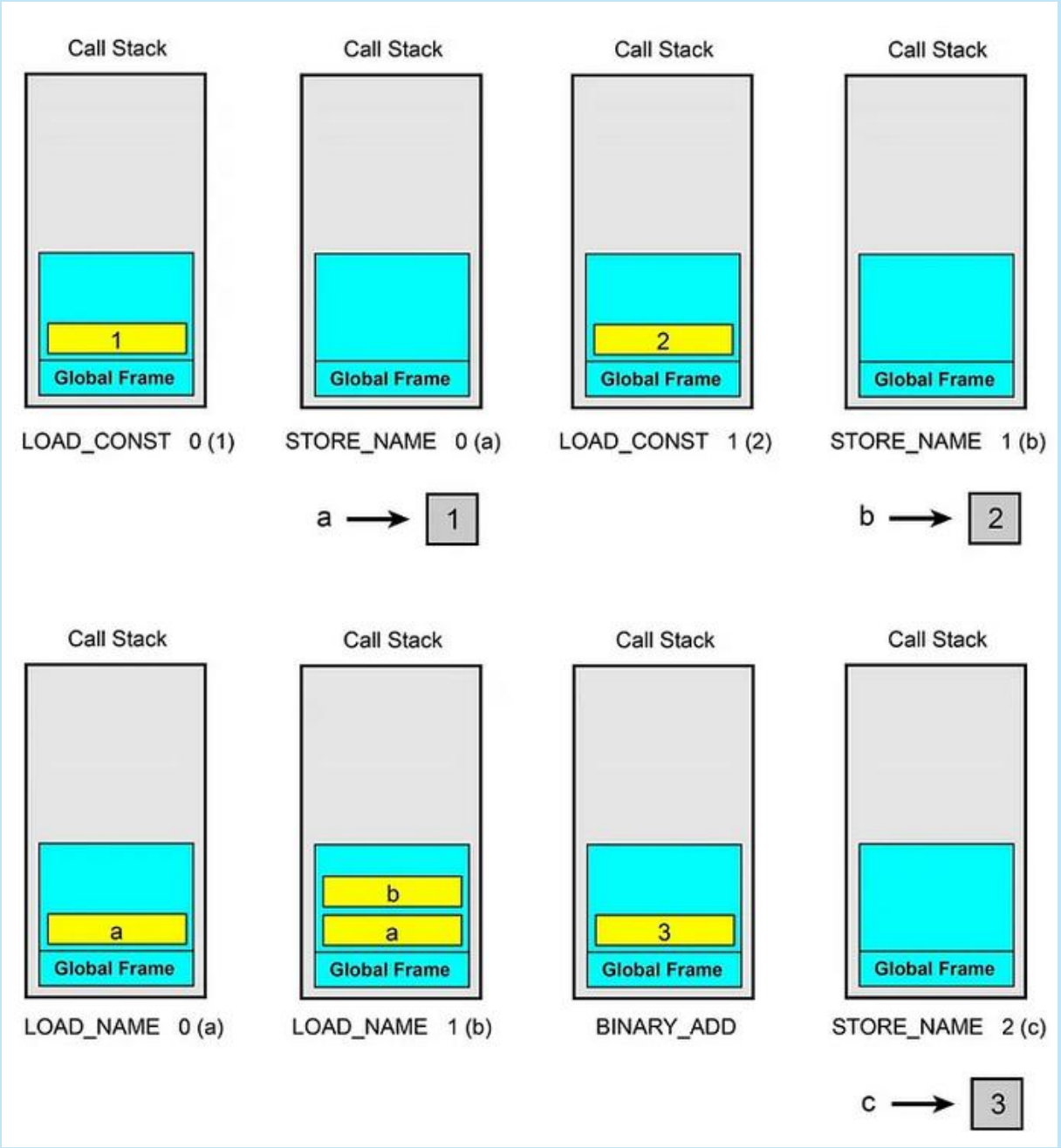
- Verem alapú virtuális gép (LIFO)
- 3 féle verem
  - Hívási (függvényhívások)
  - Kiértékelési (utasítások)
  - Blokk (vezérlési szerkezetek)

dis modul (disassembly)

```
>>> dis.dis("[]")
...
          0           0 RESUME                   0

          1           2 BUILD_LIST               0
                      4 RETURN_VALUE
>>> dis.dis("list()")
...
          0           0 RESUME                   0

          1           2 PUSH_NULL
                      4 LOAD_NAME                0 (list)
                      6 CALL                     0
                     14 RETURN_VALUE
```



PVM

- Bájtkód utasítás/változó bekerül a verembe
- Adatok ezek szerint értékelődnek ki!

https://medium.com/towards-data-science/understanding-python-bytecode-e7edaae8734d

Compiler-Interpreter

COMPILATION

- Felhasználónak nem kell a fordító, csak a programozónak
- Lefordított kód végrehajtása általában gyorsabb
- Időigényes, hibakeresés, javítás kellhet
- Nem platformfüggetlen
- Több futásbeli hibát észre tud venni

INTERPRETATION

- Felhasználónak kell interpreter
- Kész kódot egyből futtatja, fordítási fázis nélkül
- Kód nem gépi nyelven van tárolva – multiplatform
- Számítógép osztozik az erőforrásain az interpreterrel

Első program!

print("Hello World!")

```
>>> print(
        (*args, sep=' ', end='\n', file=None, flush=False)
        Prints the values to a stream, or to sys.stdout by default.
```

print("Hello", "hello", "hello", "world!", sep="-", end="\n\n")

```
>>> print("Hello", "hello", "hello", "world!", sep="-", end="\n\n")
Hello-hello-hello-world!


>>>
```

Bemenet kezelése

```
name = input("Hogy hívnak? ")
print("Szia ", name, "!")
print("Szia {n}!".format(n = name))
print(f"Szia {name}!")
print("Szia "+ name + "!")
```

- Adatokat **változókban** tároljuk el, ha dolgozni akarunk velük!
  - Ha nem, használjunk **alulvonást**!

```
_ = input("Hogy hívnak? ")
print("Szia Reginald!")
```

Változók Pythonban

Mutable

- Lista
- Dictionary
- Set
- Bytearray

Immutable

- Int
- Float
- Complex
- Bool
- Str
- Tuple
- Frozenset
- Bytes
- NoneType

Változók a háttérben

- PyObject
  - **Referenciák számát** eltárolja, **típussal** együtt
- PyVarObject
  - **Adatszerkezetek** számára – pointertömböt, alap méretet is tárol

Immutable

```
>>> name1 = "Reginald"
>>> name2 = "Reginald"
>>> id(name1) == id(name2)
True
>>>
```

Ugyanarra a címre fognak mutatni!

Immutable adattagjait nem tudjuk módosítani!

Kérdések

Ugyanazon a memóriacímen lesznek-e a változóink?

- list1, list2 = [1, 2, "piton"], [1, 2, "piton"]
  - Ez a **szimultán értékadás!**
    - Példa rá még: literature = grammar = chemistry = 1
- list1[-1], list2[-1]
  - Ez a **hátulról indexelés!**
- set([1,2,3]), frozenset([1,2,2])
  - a, b = set([1,2,3]), frozenset([1,2,3])
- a, b = ([1,2,3],) , ([1,2,3],)
  - Ez egy **egyelemű tuple!**
  - a, b = (2,2), (2,2)
- a, b = 256, 256
  - a, b = 257, 257

Alap adattípusok

Elemi adattípusok

- Egész szám (int)
- Valós szám (float)
- Komplex szám (complex)
- Logikai értékek (bool)

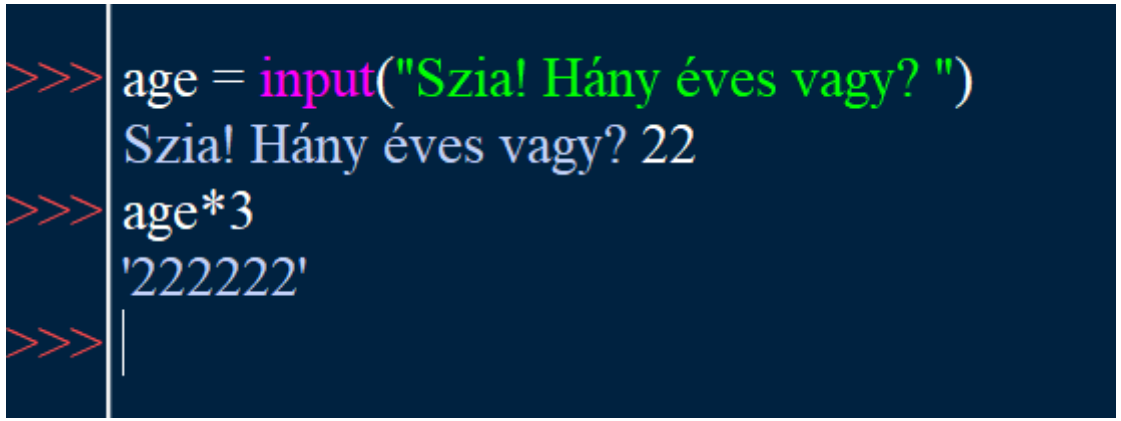
Összetett adattípusok

- Szöveg (string)
- Tuple
- Lista (list)
- Szótár (dictionary)
- Halmaz (set)

Külön karakter típus nincs!

Típusok

- Dinamikusan kezeltek – de bele lehet nyúlni!



- Típusellenőrzés: **type(age), isinstance(age, int)**
- Típuskonvertálás: **age = int(age)**
  - Vagy ezesetben: int(input("Szia! Hány éves vagy? ") is működik.

Típusok

- Az interpreter **nem biztosítja a típushelyességet sima típusannotációval!**

```
>>> dogs: int = input("Hány kutyád van? ")
Hány kutyád van? 101
>>> dogs*10
'1011011011011011011011011011011'
>>>
```

- Fontos, hogy megbizonyosodjunk a típushelyességről futási időben is!
- **Annotáció** általában csak IDE környezeteknek szól.

Típusok

- Bizonyos típusok között van átjárás
  - list(set("adam"))
  - int(3.1415) → 3, int(2.718) → 2
  - tuple("szia")

```
>>> list(set("adam"))
['m', 'a', 'd']
>>> tuple("szia")
('s', 'z', 'i', 'a')
>>>
```

- Bizonyos típusok között nincs/nem mindig van átjárás
  - int(str(3.)), int(str(10))
  - dict(("a1","b2")), dict(("a",1),("b",2))
  - **tuple(dict(("a1","b2")))**
  - int(3 + 0j)

```
>>> dictionary = dict(("a1","b2"))
>>> dictionary
{'a': '1', 'b': '2'}
>>> tuple(dictionary)
('a', 'b')
>>> dict(("a",1))
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    dict(("a",1))
ValueError: dictionary update sequence element #0 has length 1; 2 is required
>>>
```

Adatszerkezetek típusai

- Pythonban az **alap adatszerkezetek heterogének!**
  - azaz több típusú változót tudnak tárolni egyszerre.

```
>>> data = list([1, "2025", "hello", 3.1415, 1 + 34j, [27], {1, 2, 3, "text"}, (1,), 13])
>>> type(data[0])
<class 'int'>
>>> type(data[4])
<class 'complex'>
>>> type(data[-2])
<class 'tuple'>
>>>
```

- Nagy **overheaddel** jár – léteznek homogén adatszerkezetek is

Műveletek

- Alap aritmetikai műveletek
  - +, -, /, %, *, //, **
  - Mindegyik használható értékadással!
    - pl. x = 10; x **= 3
- Összehasonlító operátorok
  - ==, !=, <, >, <=, >=
- Logikai operátorok
    - or
    - and
    - not

Műveletek

Műveleti sorrend van!

Hatványozás jobbról kiértékelendő!

Zárójelezés élvezi a legnagyobb prioritást

Prioritás	Operátor	neve	
1	+, -	előjel	egyváltozós
2	**	hatványozás	kétváltozósak
3	*, /, //, %	szorzás, …	kétváltozósak
4	+, -	összeadás, kivonás	kétváltozósak

Kérdések

Mik lesznek a változók típusai?

- type(0xab)
- type(0o70)
- type(print("2"))
- type(True)
- type(3.)
- type("2000")
- type(None)
- type(object)
- type({1,2})
- type(1.2 + 3.1j)
- type((1,))
- type([])
- type((1 + 4j) – 4j)
- type(1+False)
- type(False + True)
- type(2. ** 3)
- type(2 ** 3.)
- type(6. // 2)
- type(12 / 3)
- type(121 // 11)

Műveletek

Pár fontosabb művelet alap adatstruktúrákra:

Halmazra:

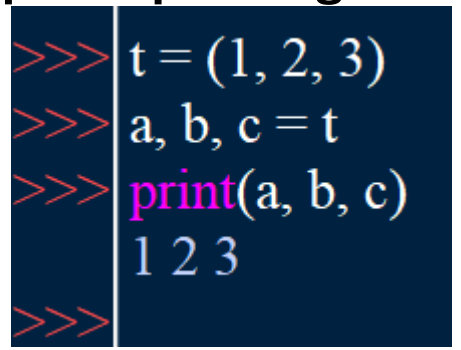
- set.add() : hozzáadás
- set.remove() : törlés
- x in set: tartalmazásvizsgálat

Dictionaryre:

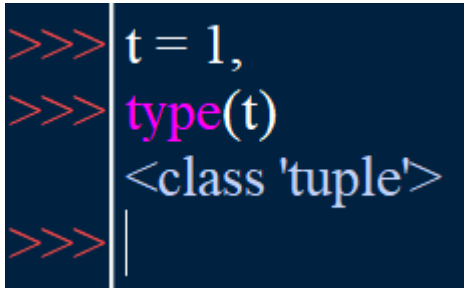
- dict["Name"] : értékre hivatkozás kulcs alapján
- dict.items() : tupleban a kulcs-érték párok
- dict.keys(), dict.values() : kulcsok/értékek listában

Tuplere:

- **Tuple unpacking**



- Egyelemű tuple (vessző miatt!)



Elemszám lekérése: len(x)

Törlés: del x

Listaműveletek – szeletelés, másolás

Slicing

- Megadható **lépésköz**, [x, y) formában működik (intervallumként tekintve)
- Lista[innen : idáig : ennyi lépésközzel]
- Ha a **lépésköz -1 → hátulról megy**

```
>>> nums = [0, 1, 2, 3, 4]
>>> nums[:]
[0, 1, 2, 3, 4]
>>> nums[0:1]
[0]
>>> nums[1:1]
[]
>>> nums[-1:-3]
[]
>>> nums[-3:-1]
[2, 3]
>>> nums[::2]
[0, 2, 4]
>>>
```

nums_copy **listát másol**

nums_alias **referenciát!**

referenciamásolás: **copying**

```
>>> nums_copy = nums[:]
>>> nums_alias = nums
>>> nums[0] = "OUT!"
>>> nums_copy
[0, 1, 2, 3, 4]
>>> nums_alias
['OUT!', 1, 2, 3, 4]
>>>
```

Listaműveletek, string műveletek

Fontosabb listaműveletek

lst.append(x), lst.count(x), lst.pop(x), lst.sort(), lst.remove(x), lst.insert(i, x)

Bővebben: https://www.w3schools.com/python/python_ref_list.asp

Fontosabb string műveletek

s.count(x), " ".join(**iterable**), s.lower(), s.upper(), s.split(x), s.startswith(x)

Bővebben: https://www.w3schools.com/python/python_ref_string.asp

Kérdések

Mit fognak csinálni a következő műveletek?

- (3,4) * 3
  - [1,2] * 2
  - [3,4] + [2,1]
  - {1,2} * 2
- "-".join(["hello", "world!"])
  - ".".join(("hello","world?"))
  - "".join({"hello", "world??"})
  - " ".join({"h":1, "ello":2})
- a, b = {1,2}, {2,4}
  - a & b, a | b
  - 2 in (a or b)
  - 1 in (b or a)
- a = {"Name" : "Mario"}
  - a["Name"] = "Luigi"
  - a["Age"] = 40

Kérdések

Mit fognak csinálni a következő műveletek?

- s = "hallo welt!"
  - s[::2]
  - s[-5:]
- s = "indula gör"
  - s += s[-2::-1]

- lst = [0, 1, 2, 3, 4, 5, 6]
  - a = lst[:]
  - b = lst
  - lst = lst[0::2]
- lst = [1, 10, 100, 1000]
  - lst[1:3] = lst[1:3][::-1]
  - lst[3] = lst
  - del lst[3]

Névkonvenciók

snake_case használatos!

Változónév nem kezdődhet számmal, csak alfanumerikus karaktereket és aláhúzást tartalmazhat!

Különbséget teszünk kis és nagybetűk között is.

Használjunk beszédes változóneveket! (pl. x helyett: client_number)

A kulcsszavak tiltottak:

and	**assert**	**break**	**class**	**continue**	**def**
del	**elif**	**else**	**except**	**exec**	**finally**
for	**from**	**global**	**if**	**import**	**in**
is	**lambda**	**not**	**or**	**pass**	**print**
raise	**return**	**try**	**with**	**while**	**yield**
False	**None**	**True**			

Mi történt eddig?

- Beszéltünk a Pythonról.
- Megírtuk az első programunkat.
- Alap típusokat beszéltünk meg.
- Alap műveleteket beszéltünk meg.
- Néztünk példákat típusokra, műveletekre.
- Megismerkedtünk az alap adatszerkezetekkel.
- Megismerkedtünk az alap adatszerkezetek fontosabb műveleteivel.
- Volt szó a névkonvenciókról, kulcsszavakról.

Feladatok Canvasben!

Köszönöm a figyelmet!