**3.4. Sztring**

A sztring adattípus szöveges értékek tárolására szolgál. Pythonban a sztring nem más mint Unicode

szimbólumok (másnéven Unicode karakterek) nem módosítható sorozata.

Bemenet[20]: # A sztringállandót ' jelekkel határoljuk.

'alma'

4

Kimenet[20]: 'alma'

Bemenet[21]: # ...de lehet használni " jeleket is.

"körte"

Kimenet[21]: 'körte'

Bemenet[22]: # Megjegyzés: Az el®z® cellák kimenetében a ' nem a sztring része, csak az

# adattípust jelzi. Írjuk ki a sztring tartalmát, határoló jelek nélkül!

print('alma')

alma

Bemenet[23]: # A type függvény most is m¶ködik.

type('alma')

Kimenet[23]: str

Bemenet[24]: # A sztringben természetesen használhatunk Unicode szimbólumokat.

'*p* ¥ **N**'

Kimenet[24]: '*p* ¥ **N**'

Bemenet[25]: # A kétféle határoló értelme:

print("asd'fgh")

print('asd"fgh')

asd'fgh

asd"fgh

Bemenet[26]: # ...egyébként le kéne védeni az ' ill. ' karaktert.

print('asd\'fgh')

print("asd\"fgh")

asd'fgh

asd"fgh

Bemenet[27]: # Hozzunk létre egy s nev¶ sztringváltozót!

s = 'sör'

Bemenet[28]: # s karaktereinek kinyerése.

# Megjegyzés: Az indexelés 0-tól indul.

s[0]

Kimenet[28]: 's'

5

Bemenet[29]: # A kinyert karaktert egy 1 hosszú sztring formájában kapjuk vissza.

type(s[0])

Kimenet[29]: str

Bemenet[30]: # Túlindexelés esetén hibaüzenetet kapunk.

s[3]

---------------------------------------------------------------------------

IndexError Traceback (most recent call last)

<ipython-input-11-da58443a8b01> Bemenet<module>()

1 # Túlindexelés esetén hibaüzenetet kapunk.

----> 2 s[3]

IndexError: string index Kimenet of range

Bemenet[31]: # A sztring karaktereit nem lehet módosítani!

# (Majd kés®bb meglátjuk, hogy miért.)

s[0] = 'x'

---------------------------------------------------------------------------

TypeError Traceback (most recent call last)

<ipython-input-12-a489aa6d5eb9> Bemenet<module>()

1 # A sztring karaktereit nem lehet módosítani!

2 # (Majd kés®bb meglátjuk, hogy miért.)

----> 3 s[0] = 'x'

TypeError: 'str' object does not support item assignment

Bemenet[32]: # Természetesen s-nek adhatunk új értéket.

s = 'bor'

# Megjegyzés: Az értékadás megtörténik, de magának az értékadó kifejezésnek

# nincs eredménye. Emiatt a cellának nincsen kimenete.

Bemenet[33]: # Írjuk ki s tartalmát!

print(s)

6

bor

Bemenet[34]: # A sztring hossza (Unicode szimbólumok száma):

len('Béla¥')

Kimenet[34]: 5

Bemenet[35]: # Sztringek összef¶zése.

'sör' + 'bor'

Kimenet[35]: 'sörbor'

Bemenet[36]: # Tartalmazásvizsgálat.

'ab' Bemenet'abrakadabra'

Kimenet[36]: True

Bemenet[37]: # Sztringb®l a kódolás m¶veletével képezhetünk bájtsorozatot.

b = 'Géza'.encode('utf-8')

b

Kimenet[37]: b'G\xc3\xa9za'

Bemenet[38]: # Az eredmény típusa.

type(b)

Kimenet[38]: bytes

Bemenet[39]: # A bájtok száma nagyobb lehet, mint a Unicode szimbólumok száma!

len(b)

Kimenet[39]: 5

Bemenet[40]: # Feladat:

# Hány bájton tárolódnak a magyar ábécé ékezetes kisbet¶i UTF-8 kódolás esetén?

print(len('á'.encode('utf-8')))

print(len('é'.encode('utf-8')))

print(len('í'.encode('utf-8')))

print(len('ó'.encode('utf-8')))

print(len('ö'.encode('utf-8')))

print(len('®'.encode('utf-8')))

print(len('ú'.encode('utf-8')))

print(len('ü'.encode('utf-8')))

print(len('¶'.encode('utf-8')))

# Megjegyzés: A fenti kód tele van ismétléssel.

# Hamarosan megtanuljuk, hogy hogyan lehet elegánsabbá tenni.

7

2

2

2

2

2

2

2

2

2

Bemenet[41]: # Hány bájton tárolódik a *p* és a ¥ szimbólum?

print(len('*p*'.encode('utf-8')))

print(len('¥'.encode('utf-8')))

2

3

Bemenet[42]: # Bájtsorozatból a dekódolás m¶veletével képezhetünk sztringet.

b.decode('utf-8')

Kimenet[42]: 'Géza'

Bemenet[43]: # Üres sztring létrehozása.

''

Kimenet[43]: ''

Bemenet[44]: # Fehér karakterek (szóköz, tabulátor, sortörés) eltávolítása

# a sztring elejér®l és végér®l.

' \talma\n'.strip()

Kimenet[44]: 'alma'

Bemenet[45]: # Megadott karakterek eltávolítása a sztring elejér®l és végér®l.

'---alma+++'.strip('+-')

Kimenet[45]: 'alma'

**3.5. Logikai érték**

A logikai igaz értéket a True, a hamisat a False jelöli. A nagy kezd˝obet˝u fontos, a Python különböz˝onek

tekinti a kis- és nagybet˝ uket.

Bemenet[46]: # Hozzunk létre logikai típusú változót!

x = True

x

8

Kimenet[46]: True

Bemenet[47]: # Logikai ÉS m¶velet.

print(True and False)

print(True and True)

False

True

Bemenet[48]: # Logikai VAGY m¶velet.

print(False or False)

print(False or True)

False

True

Bemenet[49]: # Logikai tagadás.

not x

Kimenet[49]: False

Bemenet[50]: # Az összehasonlító m¶veletek eredménye logikai érték.

print(2 <= 3)

print(5 > 10)

True

False

Bemenet[51]: # Pythonban az egyenl®ségvizsgálat m¶veleti jele ==.

print('alma' == 'alma')

print('alma' == 'körte')

True

False

**3.6. None**

A szó jelentése semmi vagy egyik sem. A Pythonban a None értéknek helykitölt˝o szerepe van. Ezzel

jelölhetjük pl. a hiányzó vagy érvénytelen eredményt vagy az alapértelmezett beállítást.

Bemenet[52]: # A None érték típusa.

type(None)

Kimenet[52]: NoneType

Bemenet[53]: # Ha a cella utolsó kifejezése None érték¶, akkor nincs kimenet.

1 + 1

None

*Sztring*

Szöveg típus, melyet a Pythonban két formában is megadhatunk. ’alma’ vagy ”alma” mindkettő forma helyes. Több sorból álló szöveg ’’’ Három darab aposztróffal kezdődő és a végén hasonlóan három darab ’’’ jellel záródik.

Műveletek sztringekkel:

A szöveg típusú változókkal kapcsolatban több metódus áll rendelkezésre, melyekkel a sztringek kezelését elvégezhetjük. Legyenek a következő változók nevei txt1, txt2 stb.

len(txt1) # megadja a txt1 sztring karaktereinek számát

txt1 in txt2 # ellenőrzi, hogy txt1 benne van e txt2 -ben. True az értéke ha igen.

Szövegek „szeletelése”.

txt1[ 2:5 ] # txt1 2. pozíciótól 5-ig tartó rész

txt1[ :5 ] # txt1 elejétől 5-ös pozícióig

txt1[ 2: ] # txt1 2 pozíciótól a végéig tartó szakasz

Szövegek összefűzése. (concatenate)

txt1 + txt2 # eredménye a két sztring ”összege”

Szövegek „módosítását” végző függvények közül pár darab?

txt1.upper() # txt1 minden karakterét nagybetűssé alakítja

txt1.lower() # txt1 minden karakterét kisbetűssé alakítja

txt1.strip() # txt1 elejéről/végéről leszedi a szóközöket

txt1.split(’,’) # txt1-t széthasítja a paraméterként megadott szeparátoroknál

txt1.replace(txt2,txt3) # txt1 ben megtalált txt2 sztringet txt3-mal írja felül.

Néhány method mellyel egy sztring ellenőrzését végezhetjük

txt1.isalpha() # Ha txt1 alfabetikus karakterekből áll akkor True

txt1.isdigit() # True, ha minden karakter számjegy

txt1.isprintable() # True, ha minden karakter nyomtatható

txt1.isupper() # True, ha minden karakter nagybetű

txt1.isspace() # True, ha minden karakter „whitespace”

*Logikai*

A logikai típusú változók két értéket vehetnek fel: True vagy False

Több olyan függvény létezik a Python nyelvbe beépítve, melyek logikai értékeket adnak vissza. (pl. sztringek esetében a különböző isdigit(), isalpha() etc …) Logikai értékkel tér vissza minden olyan művelet, melyek során változók, kifejezések értékeit hasonlítjuk össze. Használjuk a következő változó neveket v1, v2, v3. Tegyük fel azonos típusúak, akkor a következő műveletek eredményei logikai jellegűek lesznek:

v1 == v2 (azonos), v1 < v2 (kisebb), v1 > v2 (nagyobb), v1 != v2 (nem egyenlő), v1 >= v2 (nagyobb egyenlő), v1 <= v2 (kisebb egyenlő) összehasonlítások eredményei True vagy False lesznek v1 és v2 értéktől függően.

Bitek szintjén értelmezett logikai műveletek:

**& and** 1 & 1 True 0 & 0 0 & 1 1 & 0 esetén False értékű a művelet

**| or** 0 | 0 False 0 | 1 1 | 1 1 | 1 esetén True értékű a művelet 8

**^ xor** 0 ^ 0 1 ^ 1 False 0 ^ 1 0 ^ 1 esetén True

**~ not** ha True volt akkor False ha True volt akkor False érték

**<< Bitenként balra léptet**

**>>** Bitenként jobbra léptet