

# A rekurzióról röviden

Dr. Szeghalmy Szilvia  
Debreceni Egyetem, Informatikai Kar

# Rekurzió

- ▶ Ha egy függvény önmagát hívja közvetlen vagy közvetett módon:
  - közvetlen: a függvény törzsén belül áll a saját magát meghívó függvényhívás
  - közvetett: pl. meghív egy másik függvényt, ami meghívja őt.
- ▶ Matematikából ismerősek lehetnek a rekurzív képletek, amikor az előző tag(ok) felhasználásával határozzuk meg a keresett értéket.

- ▶ Pl.: Első  $n$  nem negatív egész szám összege:

$$S_0 = 0 \quad \# \text{ különleges eset}$$

$$S_1 = 1 + S_0 \quad \# \text{ a többi esetben a problémát visszavezetjük egy korábbi esetre}$$

$$S_2 = 2 + S_1$$

...

$$S_n = n + S_{n-1}$$

$$S(n) = \begin{cases} 0, & \text{ha } n = 0 \\ n + S(n-1), & \text{ha } n > 0 \end{cases}$$

```
def sn(n):  
    if n <= 0: # különleges eset  
        return 0  
    return n + sn(n-1) # általános eset
```

# Rekurzió

## ▶ Faktoriális számítása

```
0! = 1          # 0 faktoriális definíció szerint 1
1! = 1 * 0!
2! = 2 * 1!
...
n! = n * (n-1)!
```

$$fact(n) = \begin{cases} 1, & \text{ha } n = 0 \\ n * fact(n - 1), & \text{ha } n > 0 \end{cases}$$

```
def fact(n):
    assert( n >= 0 ) # elvárás
    if n == 0: # különleges eset
        return 1
    return n * fact(n-1) #általános
```

Általánosságban:

- ▶ egy-vagy több könnyen számítható / különleges esetre az értékek meghatározása
- ▶ a többi probléma visszavezetése ezekre az esetekre

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ Minden függvényhíváskor információk kerülnek be a verembe
  - a hívó felelőssége:
    - visszatérési cím: hol kell a függvény befejezése után folytatódnia a programnak
    - paraméterek: milyen értékekre fut le a prog
    - visszatérési érték tárolására szolgáló hely
  - a hívott függvény:
    - a lokális változók számára helyfoglalás
    - visszatérési érték elhelyezése
- ▶ Amikor a függvény befejezi a működését az információkat a megfelelő fél ki is törli a veremből.  
verem => mindig a legfelső "információs doboz" kerül ki (hivatalos neve: **keret**)

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ Lássuk mi történik az alábbi kódrészlet futtatásakor (szemléltető jelleggel)
  - A visszatérési címet a baloldali sorszámmal jelezzük
  - Egy keretet szemléltetésére most egy sort használunk
- ▶ végrehajtódó sorok: 1. (definiálja a függvényt de nem hajtja végre), 7. (fv. hívás => új keret)

```
1.  def sn(n):  
2.      if n <= 0:  
3.          return 0  
4.      snm1 = sn(n-1)  
5.      return n + snm1  
6.  
7.  x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(3)` hívás
- ▶ végrehajtódó sorok a függvényen belül: 2., 4. (fv. hívás => új keret)

```
1.  def sn(n):  
2.      if n <= 0:  
3.          return 0  
4.      snm1 = sn(n-1)  
5.      return n + snm1  
6.  
7.  x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(3): 7.	?	3	?

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(2)` hívás
- ▶ végrehajtódó sorok a függvényen belül: 2., 4. (fv. hívás => új keret)

```
1.  def sn(n):  
2.      if n <= 0:  
3.          return 0  
4.      snm1 = sn(n-1)  
5.      return n + snm1  
6.  
7.  x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(2): 4.	?	2	?
sn(3): 7.	?	3	?

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(1)` hívás
- ▶ végrehajtódó sorok a függvényen belül: 2., 4. (fv. hívás => új keret)

```
1.  def sn(n):  
2.      if n <= 0:  
3.          return 0  
4.      snm1 = sn(n-1)  
5.      return n + snm1  
6.  
7.  x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(1): 4.	?	1	?
sn(2): 4.	?	2	?
sn(3): 7.	?	3	?

verem



# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(0)` hívás
- ▶ végrehajtódó sorok a függvényen belül: 2., 3. (`return 0` => keret törlődik)

```
1.  def sn(n):  
2.      if n <= 0:  
3.          return 0  
4.      snm1 = sn(n-1)  
5.      return n + snm1  
6.  
7.  x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(0): 4.	0	0	?
sn(1): 4.	?	1	?
sn(2): 4.	?	2	?
sn(3): 7.	?	3	?

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(1)` folytatása
- ▶ végrehajtott sorok a függvényen belül: 2., 4 (fv hívás)
- ▶ folytatás: 4. (`snm1` megkapja a visszatérési értéket), 5. (`return 1` => keret törlődik)

```
1. def sn(n):  
2.     if n <= 0:  
3.         return 0  
4.     snm1 = sn(n-1)  
5.     return n + snm1  
6.  
7. x = sn(3)
```

0

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(0): 4.	0	0	?
sn(1): 4.	1+0 = 1	1	0
sn(2): 4.	?	2	?
sn(3): 7.	?	3	?

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(2)` folytatása
- ▶ végrehajtott sorok a függvényen belül: 2., 4 (fv hívás)
- ▶ folytatás: 4. (`snm1` megkapja a visszatérési értéket), 5. (`return 3` => keret törlődik)

```
1. def sn(n):  
2.     if n <= 0:  
3.         return 0  
4.     snm1 = sn(n-1)  
5.     return n + snm1  
6.  
7. x = sn(3)
```

függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(0): 4.	0	0	?
sn(1): 4.	1+0 = 1	1	0
sn(2): 4.	2+1 = 3	2	1
sn(3): 7.	?	3	?

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ `sn(3)` folytatása
- ▶ végrehajtott sorok a függvényen belül: 2., 4 (fv hívás)
- ▶ folytatás: 4. (`snm1` megkapja a visszatérési értéket), 5. (`return 6` => keret törlődik)

```
1. def sn(n):  
2.     if n <= 0:  
3.         return 0  
4.     snm1 = sn(n-1)  
5.     return n + snm1  
6.  
7. x = sn(3)
```

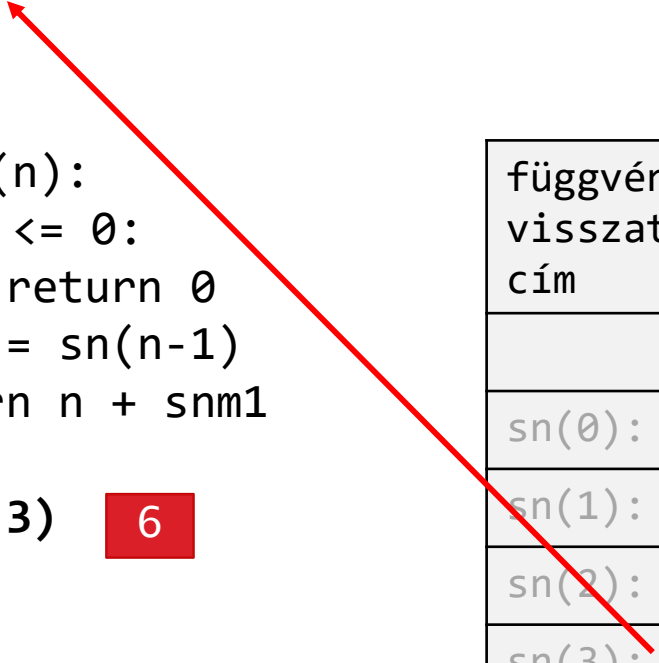
függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(0): 4.	0	0	?
sn(1): 4.	1+0 = 1	1	0
sn(2): 4.	2+1 = 3	2	1
sn(3): 7.	3+3 = 6	3	3

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ script futtatásának folytatása
- ▶ végrehajtott sorok: 1., 7 (fv hívás)
- ▶ folytatás: 7. (x megkapja a visszatérési értéket)

```
1. def sn(n):  
2.     if n <= 0:  
3.         return 0  
4.     snm1 = sn(n-1)  
5.     return n + snm1  
6.  
7. x = sn(3) 6
```



függvényhívás: visszatérési cím	visszatérési érték	paraméter (n)	lokális változó (snm1)
sn(0): 4.	0	0	?
sn(1): 4.	1+0 = 1	1	0
sn(2): 4.	2+1 = 3	2	1
sn(3): 7.	3+3 = 6	3	3

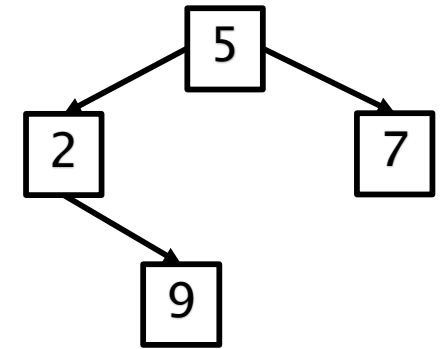
verem: a számunkra releváns része üres

*megj.: ténylegesen nem az, a modul futtatásánál is kerülnek bele infók: pl. az x a modul lokális változója.*

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ Tegyük fel, hogy a *gyoker* változó a jobb oldali ábrán látható fa gyökerét tartalmazza, amikor a 8-as sor végrehajtásra kerül.

- 8. fv. hívás => új keret



```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek

verem

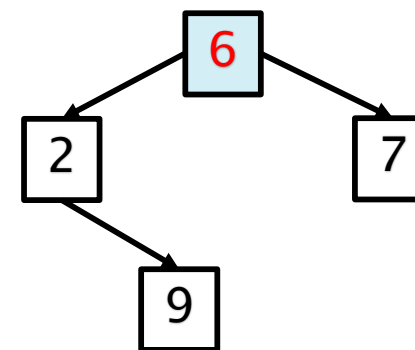
# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: / 5 → 6-os elem/

2. False

4. gy.adat módosul

5. fv. hívás => új keret



```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gyoker):8	adat:5 6   bal:(2)   jobb:(7)

verem

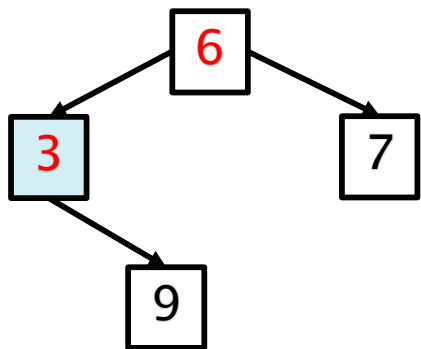
# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /2→3-as elem/

2. False

4. gy.adat módosul

5. fv. hívás => új keret



```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

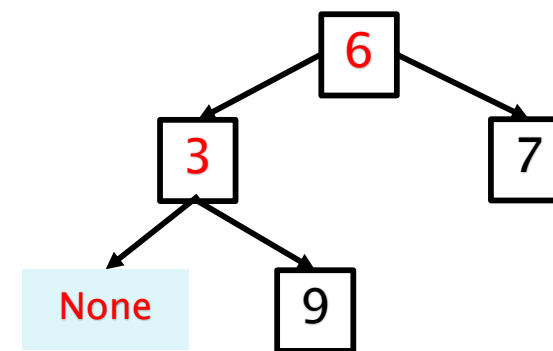
függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	adat:2 3   bal:None   jobb:(9)
novel(gyoker):8	adat: 6   bal:(3)   jobb:(7)

verem



# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /None: 3-as elem balja/
  - 2. True
  - 3. return => a függvényhívás vége, a keret törlődik



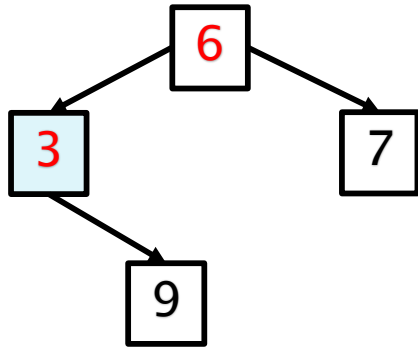
```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	None
novel(gy.bal):5	adat: 3   bal:None   jobb:(9)
novel(gyoker):8	adat: 6   bal:(3)   jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása /3-as elem/  
korábban végrehajtva: 2., 4., 5.  
visszatérés az 5. sorra és folytatás:  
6. fv. hívás => új keret



```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	None
novel(gy.bal):5	adat: 3   bal:None   jobb:(9)
novel(gyoker):8	adat: 6   bal:(3)   jobb:(7)

verem

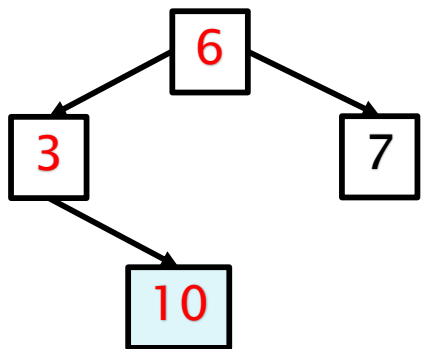
# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /9→10-es elem/

2. False

4. gy.adat módosul

5. fv. hívás => új keret



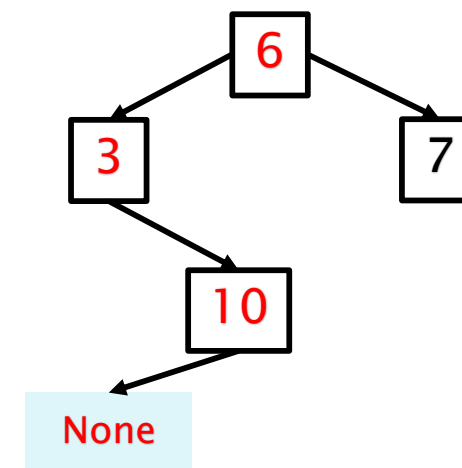
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	adat:9 10   bal:None   jobb:None
novel(gy.bal):5	adat: 3   bal:None   jobb:(10)
novel(gyoker):8	adat: 6   bal:(3)   jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /None, 10-es elem balja/
  - 2. True
  - 3. return => függvény hívás vége, keret törlődik



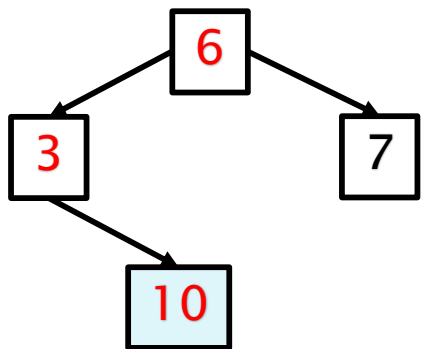
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	None
novel(gy.jobb):6	adat: 10   bal:None  jobb:None
novel(gy.bal):5	adat: 3   bal:None  jobb:(10)
novel(gyoker):8	adat: 6   bal:(3)  jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 10-es elem/  
korábban végrehajtva: 2., 4., 5.  
visszatérés az 5. sorra és folytatás:  
6. fv. hívás => új keret



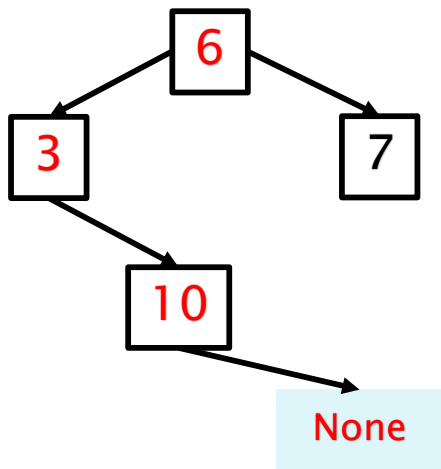
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
<code>novel(gy.bal):5</code>	None
<code>novel(gy.jobb):6</code>	adat: 10   bal:None  jobb:None
<code>novel(gy.bal):5</code>	adat: 3   bal:None  jobb:(10)
<code>novel(gyoker):8</code>	adat: 6   bal:(3)  jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /None: 10-es elem jobbja/
  - 2. True
  - 3. return => a függvényhívás vége, a keret törlődik



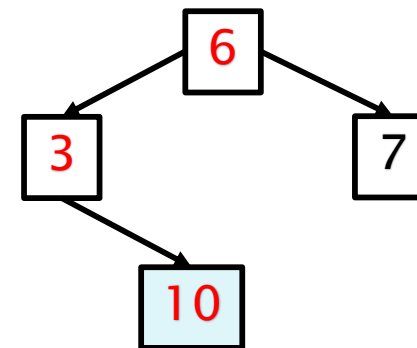
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	None
novel(gy.jobb):6	adat: 10   bal:None  jobb:None
novel(gy.bal):5	adat: 3   bal:None  jobb:(10)
novel(gyoker):8	adat: 6   bal:(3)  jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 10-es elem/  
korábban végrehajtva: 2., 4., 5., 6.  
visszatérés az 6. sorra, de nincs több sor  
=> a keret törlődik



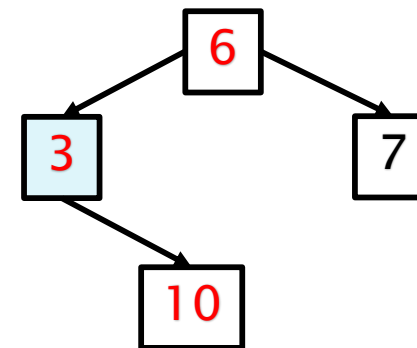
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
<code>novel(gy.jobb):6</code>	None
<code>novel(gy.jobb):6</code>	adat: 10   bal:None  jobb:None
<code>novel(gy.bal):5</code>	adat: 3   bal:None  jobb:(10)
<code>novel(gyoker):8</code>	adat: 6   bal:(3)  jobb:(7)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 3-as elem/  
korábban végrehajtva: 2., 4., 5., 6.  
visszatérés az 6. sorra, de nincs több sor  
=> a keret törlődik



```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

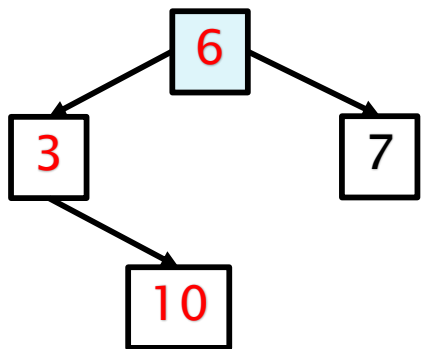
függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	adat: 10   bal:None  jobb:None
novel(gy.bal):5	adat: 3   bal:None  jobb:(10)
novel(gyoker):8	adat: 6   bal:(3)  jobb:(7)

verem



# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 6-os elem/  
korábban végrehajtva: 2., 4., 5.  
visszatérés az 5. sorra és folytatás  
6. fv. hívás => új keret



```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	adat: 3   bal:None   jobb:(9)
novel(gyoker):8	adat: 6   bal:(3)   jobb:(7)

verem

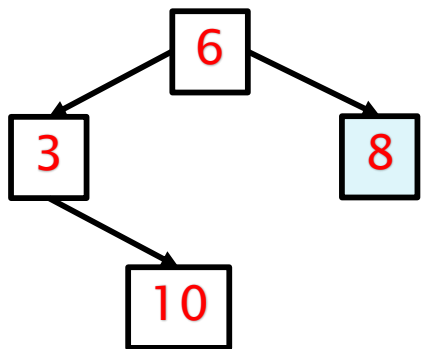
# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /8-as elem/

2. False

4. gy.adat módosul

5. fv. hívás => új keret



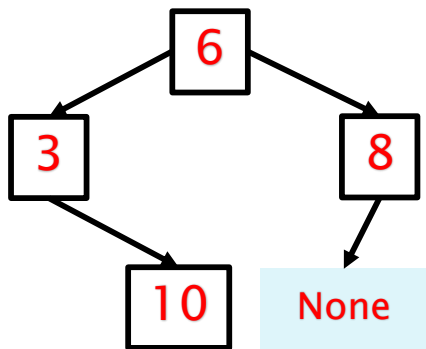
```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	adat:7 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)   jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /None, 8-as elem balja/
  - 2. True
  - 4. return => a függvényhívás vége, a keret törlődik



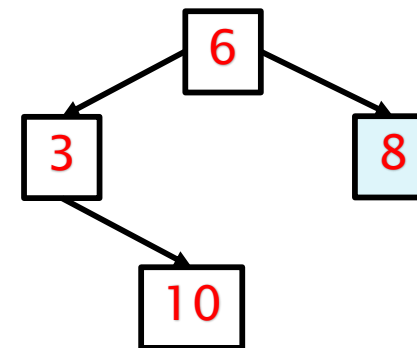
```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	None
novel(gy.jobb):6	adat: 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)  jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 8-as elem/  
korábban végrehajtva: 2., 4., 5.  
visszatérés az 5. sorra és folytatás  
6. fv. hívás => új keret



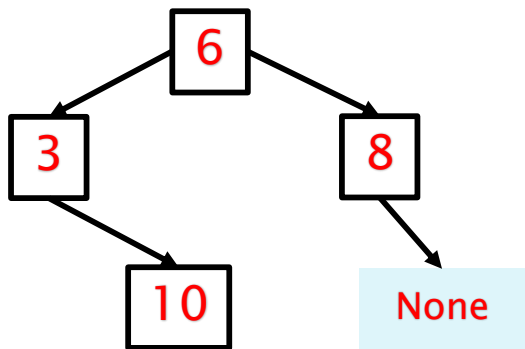
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.bal):5	None
novel(gy.jobb):6	adat: 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)  jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ végrehajtódó sorok a függvényen belül: /None, 8-as elem jobbja/
  - 2. True
  - 4. return => a függvényhívás vége, a keret törlődik



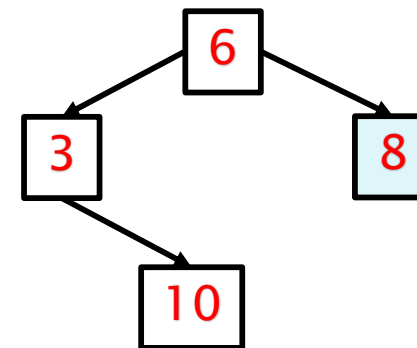
```
1. def novel(gy):
2.     if not gy:
3.         return
4.     gy.adat += 1
5.     novel(gy.bal)
6.     novel(gy.jobb)
7.
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	None
novel(gy.jobb):6	adat: 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)  jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 8-as elem/  
korábban végrehajtva: 2., 4., 5., 6.  
visszatérés az 6. sorra, de nincs több sor  
=> a keret törlődik



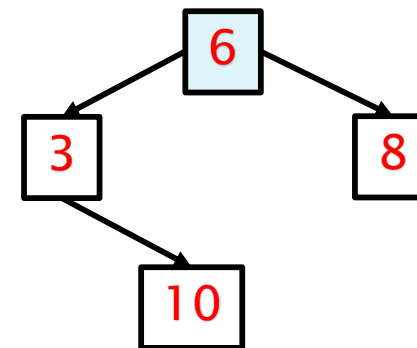
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	None
novel(gy.jobb):6	adat: 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)  jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ folytatódik az előző függvényhívás végrehajtása / 8-as elem/  
korábban végrehajtva: 2., 4., 5., 6.  
visszatérés az 6. sorra, de nincs több sor  
=> a keret törlődik



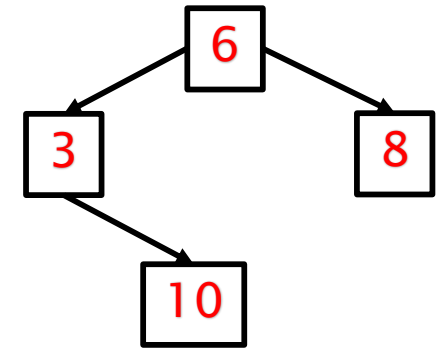
```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gy.jobb):6	adat: 8   bal:None   jobb:None
novel(gyoker):8	adat: 6   bal:(3)   jobb:(8)

verem

# Rekurzió: hogyan működik egy ilyen kód?

- ▶ a script futtatása a 8. sor után folytatódik



```
1. def novel(gy):  
2.     if not gy:  
3.         return  
4.     gy.adat += 1  
5.     novel(gy.bal)  
6.     novel(gy.jobb)  
7.  
8. novel(gyoker)  
9. ...
```

függvényhívás: visszatérési cím	paraméter (gy) adat, bal és jobb gyerek
novel(gyoker):8	adat: 6   bal:(3)   jobb:(8)

verem