## Egyszer?bb rekurzív függvények

1. Írj egy Sage függvényt *palindroma* néven, amely bemenetként kap egy stringet (*s*) és eldönti, hogy a string egy palindróma-e! (Azt nevezzük palindrómának ami visszafelé olvasva is ugyanaz, pl 'almaamla' vagy 'abcdcba').
2. Írj egy függvényt *bst_beszur* néven, amely bemenetként kap egy bináris keres?fát (*bst*) és egy elemet (*elem*) és ha az elem már a fában van akkor False értékkel tér vissza, egyébként beszúrja az elemet a megfelel? helyre a fába és True-t ad vissza.

## Gráfok létrehozása

- Hozz létre egy irányítatlan gráfot a szomszédsági mátrixának megadásával! Legyen 5 csúcsa, és 7 éle! Rajzold is ki a *show()* függvénnyel!

(segítség: négyzetes és szimmetrikus mátrixot adj meg! Például egy 3 csúcsú gráf így adható meg márixxal:

```
M = matrix([[0,1,0],[1,0,1],[0,1,0]])
g = Graph(M)
show(g)
```

)

- Hozz létre egy irányítatlan, 3 mélység? bináris fát *G* névvel:

```
G = graphs.BalancedTree(2,3)
```

Adj hozzá egy élet hogy kör is legyen benne:

```
G.add_edge(6,4)
```

Nézd meg, milyen lett!

- Hozz létre egy véletlenszer? irányított gráfot a *digraphs* csomag *RandomDirectedGNP()* függvényének használatával! Példa:

```
d1 = digraphs.RandomDirectedGNP(9,0.2)
```

Itt egy 9 csúcsú gráfot hozunk létre, és minden pontpár között 0.2 valószín?ségegl húzunk be élet.

## Gráfok bejárása

1. Írj egy függvényt ami egy gráfban **mélységi** keresés szerint keres meg egy elemet, egy adott, másik elemb?l kiindulva! Segítség, pszeudo-kód: DFS-wikipedia
2. Írj egy függvényt ami egy gráfban **szélességi** keresés szerint keres meg egy elemet, egy adott, másik elemb?l kiindulva! Segítség, pszeudo-kód: BFS-wikipedia
3. Próbáld ki a kétféle keresést a *g* és a *G* gráfokon!