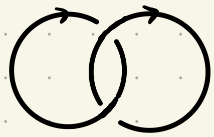
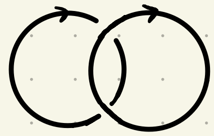
Week 4

5. Knot theory for computers

1. Write down the PD code for the following link diagrams:

a) 

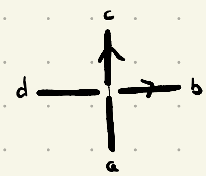
b) 

Hint: recall the procedure was:

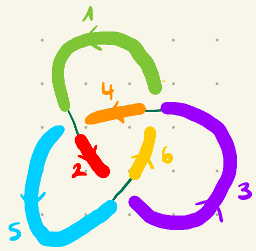
• Step 1: Break into pieces:

• Step 2: Decode using the rule:

RULE:



$\longleftrightarrow$ [a,b,c,d]



$\rightsquigarrow$ [(1,5,2,4),(5,3,6,2),(3,1,4,6)]

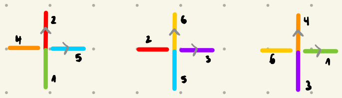
2. Draw the link diagrams associated to the following PD codes:

a) [(1,2,2,1)] b) [(1,1,2,2)]

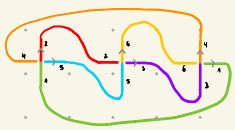
c) [(4,2,3,1), (1,3,2,4)] d) [(3,1,4,2), (4,1,3,2)]

Recall:
• Step 1: draw the crossings you need, anywhere you like:

[(1,5,2,4),(5,3,6,2),(3,1,4,6)]



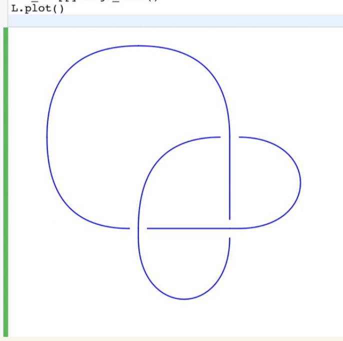
• Step 2: match the edges accordingly:

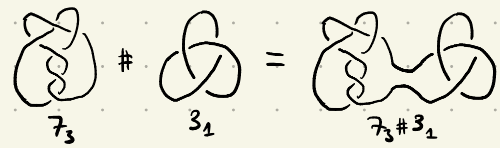


3. Plot the links you obtained in 1 and 2 in Sage.

Recall the example:

```
import snappy
PD= [(1, 5, 2, 4), (5, 3, 6, 2), (3, 1, 4, 6)]
L_snappy = snappy.Link(PD)
L=L_snappy.sage_link()
L.plot()
```



4. (Optional) Recall the connected sum of two knots: 

In Sage, this is implemented as:

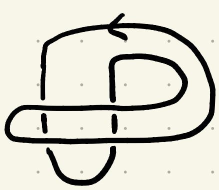
```
L1_snappy = snappy.Link('7_3')
L2_snappy = snappy.Link('3_1')
L1 = L1_snappy.sage_link()
L2 = L2_snappy.sage_link()
L_sum = L1.connected_sum(L2)
```

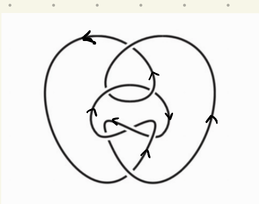
Choose any two knots you like (e.g. from the table of prime knots) and record the Jones polynomial (`L1.jones_polynomial()`) of each as `p_1` and `p_2` and verify that `L_sum.jones_polynomial()` and `expand(p_1*p_2)` are equal.

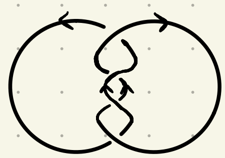
We are thus experimentally verifying that $J(K_1\#K_2) = J(K_1)\cdot J(K_2)$.

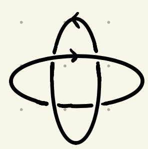
5. Obtain PD codes for the following link diagrams using SnapPy. Then verify using Sage that their Jones polynomials are equal:

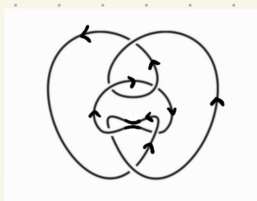
a) 





b) 

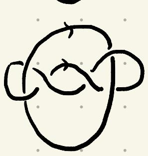




6. Obtain the Jones polynomials of the following links and their mirror images.

Do you see a pattern? Conjecture it mathematically. Help: in SnapPy's editor, go to Tools → Reflect to get the mirror image.

a) 



b)