Chapter 1: First steps in Python and Sage

Author: Vincent Delecroix
Author: Nadia Lafrenière
Author: Thierry Monteil

The three golden rules

Recall from the introduction the three most import things

First, to execute a cell on which you have the focus press Shift-Enter.

Second, to access the documentation add a question mark ? at the end and press enter

```
sage: is_prime?
```

Third, to obtain the list of methods that starts with a given prefix, press the tab key:

```
sage: is_pr
```

Objects, functions, methods

Most names in Sage are either in *snake case* like `bernoulli`, `is_prime`, `cos`, ... or *camel case* like `VectorSpace`, `PolynomialRing`, `Matrix`, ... The snake case notation is mostly reserved to functions

```
sage: bernoulli(13)
0

sage: is_prime(73)
True

sage: cos(pi/3)
1/2
```

While camel case are often objects, that is to say after calling it you obtain an object that you can query further:

```
sage: K = GF(5)           # the field with 5 elements
....: K.cardinality()     # size of K
5

sage: V = VectorSpace(K, 3)   # vector space K^3
....: V.cardinality()         # size of V
125
```

Note that in the above example we used `K.cardinality()` that actually meant `cardinality(K)`. Though in many programming languages like Python functions are bind to objects. In this situation they are named *methods*. Most method names are in lower case. Tab-completion also applies to methods. For example if you write `V.` in a cell and press the tab key you will get all methods appliable to `V`.

```
sage: V.
```

While if you write `V.card` and press tab, you will magically obtain `V.cardinality` because this is the unique method of `V` starting with `card`

```
sage: V.cardinality
```

Exercise 1.1

A *partition* of a nonnegative integer n is a non-increasing list of positive integers with total sum n. Does Sage have a command for defining a partition ?

```
sage: Partition
```

(*hint*: type `Part` and then hit the `<TAB>` key)

Create the partition with list $[4,3,1,1]$ and assign it to the Python name `p` :

```
sage: p = Partition([4, 3, 1, 1])
```

(*hint*: to see documentation and examples for the Permutation command, type `Partition?`)

Find the conjugate of `p` (and name it `q`):

```
sage: q = p.conjugate(); q
[4, 2, 2, 1]
```

Which of `p` and `q` dominates the other ?

```
sage: print p.dominates(q)
....: print q.dominates(p)
True
False
```

(*hint*: to see the methods available to the partition named `p`, you can type `p.` and hit `<TAB>`)

How did Sage decide whether `p` dominates `q` ?

```
sage: p.dominates??
```

(*hint*: to access the source code, use `p??`)

Exercise 1.2

Find out what is the name of the function to construct a matrix

```
sage: matrix([[1,2],[3,4]])
[1 2]
[3 4]
```

Compute the rank, the determinant and the Hermite normal form of the following 5x5 integer matrix

$$\begin{pmatrix} 8 & 0 & 7 & 5 & 6 \\ 9 & 0 & 6 & 9 & 4 \\ 4 & 9 & 5 & 3 & 0 \\ 6 & 9 & 4 & 9 & 3 \\ 0 & 8 & 4 & 5 & 6 \end{pmatrix}$$

```
sage: M = matrix(5, 5, [8,9,4,6,0,0,0,9,9,8,7,6,5,4,4,5,9,3,9,5,6,4,0,3,6]); M
....: print M.rank()
....: print M.determinant()
....: print M.hermite_form()
5
-7117
```

```
[   1    0    0    0 2308]
[   0    1    0    0 6657]
[   0    0    1    0 2420]
[   0    0    0    1  744]
[   0    0    0    0 7117]
```

Obtain a latex code for this matrix using the function `latex`

```
sage: latex(M)
\left(\begin{array}{rrrrr}
8 & 9 & 4 & 6 & 0 \\
0 & 0 & 9 & 9 & 8 \\
7 & 6 & 5 & 4 & 4 \\
5 & 9 & 3 & 9 & 5 \\
6 & 4 & 0 & 3 & 6
\end{array}\right)
```

Playing with integers

To create an integer, simply write it in base 10 as:

```
sage: 12395851
12395851
```

To assign the integer 14585 to a variable named `n` use:

```
sage: n = 14585
```

Once the variable is created, you can access all its methods using tab-completion. Write `n.` in the following cell and press the tab key

```
sage: n.
```

Exercise 1.3

Find out the name of the method on integers that give the list of prime factors of an integer `n`

```
sage: n.prime_factors()
[5, 2917]
```

(*note*: to do that, you first need to assign an integer to a variable and use tab-completion on this variable)

Solve Euler problem 3: what is the largest prime factor of the number 600851475143

```
sage: 600851475143.prime_factors()[-1]
6857
```

Exercise 1.4

Find the name of the function that given an integer `n` returns the `n`-th prime number

```
sage: primes_first_n(5)[-1]
11
```

Solve Euler problem 7: what is the 100001-th prime number

```
sage: primes_first_n(100001)[-1]
1299721
```

Exercise 1.5

Solve Euler problem 10: what is sum of prime numbers below two milions?

```
sage: sum(prime_range(2000000))
142913828922
```

(*hint*: You can use the function `sum` to make the sum of elements in a list.)

Exercise 1.6

What is the name of the method on integer that provides the list of digits?

```
sage: n = 131
....: n.digits()
[1, 3, 1]
```

Check that the sum of digits of $2^15 = 32768$ is $3+2+7+6+8 = 26$

```
sage: n = 32768
....: sum(n.digits()) == 26
True
```

Solve Euler problem 16 and Euler problem 20

```
sage: print sum((2^1000).digits())
....: print sum((factorial(100)).digits())
1366
648
```

Note that the comparison operators are == (for equality), != (for difference). You already used the sign = but for another purpose: assignment of value to a variable.

Graphics and Symbolic functions

It is often helpful to make pictures in mathematics. Sage comes with a lot of graphical capabilities. The main commands for 2d plotting are

- `plot` : plot an object
- `point2d` : plot a list of points
- `line2d` : draw a broken line between points
- `polygon2d` : draw a polygon

For example:

```
sage: plot(sin(x), (x, 0, 2*pi))
```

```
sage: point2d([(0,0), (1,1), (2,0), (3,1)], color='red')
```

Graphics can be assigned to variables and it is possible to superimpose them using addition:

```
sage: P1 = plot(sin(x), (x, 0, 2*pi), color='blue')
....: P2 = plot(cos(x), (x, 0, 2*pi), color='red')
....: P = P1 + P2
....: P.show()
```

Note also that many objects possess a `plot` method that allows to produce graphics:

```
sage: m = AlternatingSignMatrices(20).random_element()
....: fpl = m.to_fully_packed_loop()
....: fpl.plot(link_color_map='rainbow')
```

To learn more about graphics, have a look at http://doc.sagemath.org/html/en/reference/plotting/sage/plot/plot.html and https://doc.sagemath.org/html/en/tutorial/tour_plotting.html

Exercise 1.7

Draw a graphics of the function $f(x) = \sin(x) + \cos(x) - x\exp(x)$ for x between 0 and 1

```
sage: f(x) = sin(x) + cos(x) - x * exp(x)
....: plot(f, x, 0, 1)
```

Using the function `find_root`, obtain an approximation of the unique solution r of $f(r) = 0$ with $r \in (0,1)$.

```
sage: r = find_root(f, 0, 1); r
0.6998282883190581
```

Draw a graphics of f together with a red vertical line between $(r,-1)$ and $(r,1)$

```
sage: P1 = plot(f, 0, 1)
....: P2 = plot(line(((r, -1), (r, 1))))
....: P1 + P2
```

Exercise 1.8

What are the first 20 terms of the Taylor expansion at $x = 0$ of the function $g(x) = 1/\sqrt{1-4*x}$

```
sage: g(x) = 1 / sqrt(1 - 4 * x)
....: g.taylor(x, 0, 19)
x |--> 35345263800*x^19 + 9075135300*x^18 + 2333606220*x^17 + 601080390*x^16 + 1551
```

Check on the taylor expansion that

$$g(x) = \sum_{n\geq 0} \binom{2n}{n} x^n$$

```
sage: for n in range(20):
....:     print n, binomial(2*n, n)
0 1
1 2
2 6
3 20
4 70
5 252
6 924
7 3432
8 12870
9 48620
10 184756
11 705432
12 2704156
13 10400600
14 40116600
15 155117520
16 601080390
17 2333606220
18 9075135300
```

```
19 35345263800
```

Combinatorics

In Sage, it is possible to build set of objects such that the alternating sign matrices of size 100×100 :

```
sage: A = AlternatingSignMatrices(100)
....: A
Alternating sign matrices of size 100
```

Exercise 1.9

How many alternating sign matrices of size 100×100 are there?

```
sage: A.cardinality()
676465987581353649297661052020640615488806351651710084314673306630740208242743501456
```

How does it compare with the number of atoms in the universe?

Exercise 1.10

Euler problem 24: what is the millionth lexicographic permutation of $\{0,1,\ldots,9\}$?

```
sage: p = Permutations(10)
....: p[999999]
[3, 8, 9, 4, 10, 2, 6, 5, 7, 1]
```

Exercise 1.11

Solve Euler problem 5: what is the smallest positive number that is divisible by all of the numbers from 1 to 20?

```
sage: # NAIVE algorithm
....: # def Euler_5():
....: #     n = 1
....: #     found = false
....: #     while found == false:
....: #         found = true
....: #         for d in range(1,21):
....: #             if n % d != 0:
....: #                 found = false
....: #                 break
....: #         if found == true :
....: #                 return n
....: #         n += 1
....:
....: # optimized
....: def Euler_5(m):
....:     factors = []
....:     for n in range(1, m + 1):
....:         if is_prime(n):
....:             i = 1
....:             while n^i < m:
....:                 i += 1
```

```
....:             factors.append(n^(i-1))
....:     return prod(factors)
....:
....: Euler_5(20)
232792560
```

More resources

On the Euler project website you can find mathematical challenges to be solved with a computer. You can challenge yourself with the following ones that do not require any programming but only to find the right Sage function

Some other tutorial are available on internet, in particular

- Official Python tutorial