Optimization

June 2, 2016

```
In [1]: # Just to know last time this was run:
        import time
        print time.ctime()
```

```
Thu Jun  2 09:37:14 2016
```

1 H Optimizing code writing

This is part of the Python lecture given by Christophe Morisset at IA-UNAM. More informations at: http://python-astro.blogspot.mx/

```
In [2]: import numpy as np
        from IPython.core.display import Image
```

1.0.1 Profiling the code: CPU usage

```
In [3]: %%writefile test_1_prof.py

        import numpy as np
        import os
        import urllib2
        from scipy.integrate import simps

        class Stel_Spectrum(object):
            """
            This object downloads a file from http://astro.uni-tuebingen.de/~rauch/
            and is able to make some plots.
            """

            spec_count = 0 # This attibute is at the level of the class, not of the
            def __init__(self, filename=None, T=None, logg=None, verbose=False):
                """
                Initialisation of the Stel_Spectrum object.
                Parameter:
                    - filename
                    - T: temperature in K, e.g. 150000
                    - logg: e.g. 7.5
                    - verbose: if True, some info are printed out
                The wl variable is an array of wavelengths in Angstrom.
                The fl variable is the flux in erg/s/cm2/A
                The variables T and logg are properties: changing them will reload
                """
                self.verbose = verbose
                if filename is None:
                    if T is not None and logg is not None:
                        self.__T = T # We need to initialize the hidden values, as
                        self.logg = logg
                        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'
                    else:
                        raise TypeError("T and logg must be given")
                else:
                    self.filename = filename
                    self.__T = float(filename.split('_')[0]) # We need to initializ
                    self.logg = float(filename.split('_')[1])
                Stel_Spectrum.spec_count += 1
                if self.verbose:
                    print('Instantiation done')

            def dlfile(self):
                """
                Downloading file if not already here. Put it in the current directo
                """
                if not os.path.exists(self.filename):
                    if self.verbose:
                        print('Downloading {}'.format(self.filename))
                    try:
                        stel_file = urllib2.urlopen('http://astro.uni-tuebingen.de/
                                                    self.filename)
                        output = open(self.filename,'wb')
                        output.write(stel_file.read())
                        output.close()
                        self.file_found=True
                    except:
                        if self.verbose:
                            print('file {} not found'.format(self.filename))
                        self.file_found=False
                else:
                    if self.verbose:
                        print('{} already on disk'.format(self.filename))
                    self.file_found=True

            def read_data(self):
                """
                read the data from the file
                """
                if self.file_found:
                    data = np.genfromtxt(self.filename, comments='*', names='wl, fl
                    self.fl = data['fl']
                    self.wl = data['wl'] # in A
                    self.fl /= 1e8 # F LAMBDA  GIVEN IN ERG/CM**2/SEC/CM -> erg/s/c
                    if self.verbose:
                        print('Read data from {}'.format(self.filename))
                else:
                    if self.verbose:
                        print('file not found {}'.format(self.filename))
                    self.wl = None
                    self.fl = None

            def plot_spr(self, ax=None, *args, **kwargs):
                """
                Plot the spectrum.
                Parameter:
                    - ax: an axis (optionnal). If Noe or absent, axis is created
                    - any extra parameter is passed to ax.plot
                """
                if self.wl is None:
                    print('No data to plot')
                    return
                if ax is None:
                    fig, ax = plt.subplots()
                ax.plot(self.wl, self.fl,
                        label='T3={0:.0f}, logg={1}'.format(self.T/1e3, self.logg),
                        *args, **kwargs) # Here are the transmissions of extra para
                ax.set_yscale('log')
                ax.set_ylim(1e6, 1e14)
                ax.set_xlabel('Wavelength (A)')

            def get_integ(self):
                """
                Return the integral of Flambda over lambda, in erg/s/cm2
                """
                if self.wl is None:
                    print('No data')
                    return None
                return simps(self.fl, self.wl) # perform the integral

            def __getT(self):
                return self.__T

            def __setT(self, value):
                if not isinstance(value, (int, long, float)): # check the type of t
                    raise TypeError('T must be an integer or a float')
                if float(value) not in np.linspace(40000, 190000, 16): # check the
                    raise ValueError('T value must be between 40000 and 190000K, by
                elif self.__T != value:
                    self.__T = value
                    self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.for
                    self.dlfile() # will download new data
                    self.read_data() # will update the data

            def __delT(self):
                print('T is needed')

            T = property(__getT, __setT, __delT, "Stellar effective temperature")

            def __getlogg(self):
                return self.__logg

            def __setlogg(self, value):
                try:
                    self.__logg
                except:
                    self.__logg = -1
                if not isinstance(value, (int, long, float)):
                    raise TypeError('logg must be an integer or a float')
                if float(value) not in (-1., 5., 6., 7. ,8., 9.):
                    raise ValueError('Error, logg must be 6, 7, 8, or 9')
                    self.__logg = None
                elif self.__logg != value:
                    self.__logg = value
                    self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.for
                    self.dlfile() # will download new data
                    self.read_data() # will update the data

            def __dellogg(self):
                print('logg is needed')

            logg = property(__getlogg, __setlogg, __dellogg, "Stellar logg")

            def print_info(self):
                """
                Print out the filename and the number of points
                """
                print self.__repr__()

            def __repr__(self):
                """
                This is what is used when calling "print <obj>" or <obj> ENTER
                """
                if self.wl is None:
                    return'Filename: {0}, No data'.format(self.filename)
                else:
                    return'Filename: {0}, number of points: {1}'.format(self.filena

            def __del__(self):
                Stel_Spectrum.spec_count -= 1

        spectra = [] # we create an empty list
        for T in np.linspace(40000, 190000, 4): # this is the list of available tem
            spectra.append(Stel_Spectrum(T=T, logg=6, verbose=True)) # we fill the
        T = np.array([sp.T for sp in spectra])
        F = np.array([sp.get_integ() for sp in spectra])
        for t, f in zip(T, F):
            print('Temperature = {0:.0f}K, Flux = {1:.2e} erg/s/cm2'.format(t, f))
```

```
Overwriting test_1_prof.py
```

```
In [4]: %run -t test_1_prof.py
```

```
0040000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0040000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0090000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0090000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0140000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0140000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0190000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0190000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
Temperature = 40000K, Flux = 4.00e+13 erg/s/cm2
Temperature = 90000K, Flux = 1.05e+15 erg/s/cm2
Temperature = 140000K, Flux = 6.93e+15 erg/s/cm2
Temperature = 190000K, Flux = 2.35e+16 erg/s/cm2

IPython CPU timings (estimated):
  User   :       0.42 s.
  System :       0.04 s.
Wall time:       0.53 s.
```

```
In [5]: %run -p test_1_prof.py
```

```
0040000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0040000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0090000_6.00_33_50_02_15.bin_0.1.gz already on disk
```

```
Read data from 0090000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0140000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0140000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0190000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0190000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
Temperature = 40000K, Flux = 4.00e+13 erg/s/cm2
Temperature = 90000K, Flux = 1.05e+15 erg/s/cm2
Temperature = 140000K, Flux = 6.93e+15 erg/s/cm2
Temperature = 190000K, Flux = 2.35e+16 erg/s/cm2
```

```
In [6]: # Inserting @profile before some functions leads to detailed report on the
```

```
In [7]: %%writefile test_2_prof.py

        import numpy as np
        import os
        import urllib2
        from scipy.integrate import simps

        class Stel_Spectrum(object):
            """
            This object downloads a file from http://astro.uni-tuebingen.de/~rauch/
            and is able to make some plots.
            """

            spec_count = 0 # This attibute is at the level of the class, not of the

            @profile
            def __init__(self, filename=None, T=None, logg=None, verbose=False):
                """
                Initialisation of the Stel_Spectrum object.
                Parameter:
                    - filename
                    - T: temperature in K, e.g. 150000
                    - logg: e.g. 7.5
                    - verbose: if True, some info are printed out
                The wl variable is an array of wavelengths in Angstrom.
                The fl variable is the flux in erg/s/cm2/A
                The variables T and logg are properties: changing them will reload
                """
                self.verbose = verbose
                if filename is None:
                    if T is not None and logg is not None:
                        self.__T = T # We need to initialize the hidden values, as
                        self.logg = logg
                        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'
                    else:
                        raise TypeError("T and logg must be given")
                else:
                    self.filename = filename
                    self.__T = float(filename.split('_')[0]) # We need to initializ
                    self.logg = float(filename.split('_')[1])
                Stel_Spectrum.spec_count += 1
                if self.verbose:
                    print('Instantiation done')

            @profile
            def dlfile(self):
                """
                Downloading file if not already here. Put it in the current directo
                """
                if not os.path.exists(self.filename):
                    if self.verbose:
                        print('Downloading {}'.format(self.filename))
                    try:
                        stel_file = urllib2.urlopen('http://astro.uni-tuebingen.de/
                                                    self.filename)
                        output = open(self.filename,'wb')
                        output.write(stel_file.read())
                        output.close()
                        self.file_found=True
                    except:
                        if self.verbose:
                            print('file {} not found'.format(self.filename))
                        self.file_found=False
                else:
                    if self.verbose:
                        print('{} already on disk'.format(self.filename))
                    self.file_found=True

            @profile
            def read_data(self):
                """
                read the data from the file
                """
                if self.file_found:
                    data = np.genfromtxt(self.filename, comments='*', names='wl, fl
                    self.fl = data['fl']
                    self.wl = data['wl'] # in A
                    self.fl /= 1e8 # F LAMBDA  GIVEN IN ERG/CM**2/SEC/CM -> erg/s/c
                    if self.verbose:
                        print('Read data from {}'.format(self.filename))
                else:
                    if self.verbose:
                        print('file not found {}'.format(self.filename))
                    self.wl = None
                    self.fl = None

            def plot_spr(self, ax=None, *args, **kwargs):
                """
                Plot the spectrum.
                Parameter:
                    - ax: an axis (optionnal). If Noe or absent, axis is created
                    - any extra parameter is passed to ax.plot
                """
                if self.wl is None:
                    print('No data to plot')
                    return
                if ax is None:
                    fig, ax = plt.subplots()
                ax.plot(self.wl, self.fl,
                        label='T3={0:.0f}, logg={1}'.format(self.T/1e3, self.logg),
                        *args, **kwargs) # Here are the transmissions of extra para
                ax.set_yscale('log')
                ax.set_ylim(1e6, 1e14)
                ax.set_xlabel('Wavelength (A)')

            def get_integ(self):
                """
                Return the integral of Flambda over lambda, in erg/s/cm2
                """
                if self.wl is None:
                    print('No data')
                    return None
                return simps(self.fl, self.wl) # perform the integral

            def __getT(self):
                return self.__T

            def __setT(self, value):
                if not isinstance(value, (int, long, float)): # check the type of t
                    raise TypeError('T must be an integer or a float')
                if float(value) not in np.linspace(40000, 190000, 16): # check the
                    raise ValueError('T value must be between 40000 and 190000K, by
                elif self.__T != value:
                    self.__T = value
                    self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.for
                    self.dlfile() # will download new data
                    self.read_data() # will update the data
            def __delT(self):
                print('T is needed')

            T = property(__getT, __setT, __delT, "Stellar effective temperature")

            def __getlogg(self):
                return self.__logg

            @profile
            def __setlogg(self, value):
                try:
                    self.__logg
                except:
                    self.__logg = -1
                if not isinstance(value, (int, long, float)):
                    raise TypeError('logg must be an integer or a float')
                if float(value) not in (-1., 5., 6., 7. ,8., 9.):
                    raise ValueError('Error, logg must be 6, 7, 8, or 9')
                    self.__logg = None
                elif self.__logg != value:
                    self.__logg = value
                    self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.for
                    self.dlfile() # will download new data
                    self.read_data() # will update the data

            def __dellogg(self):
                print('logg is needed')

            logg = property(__getlogg, __setlogg, __dellogg, "Stellar logg")

            def print_info(self):
                """
                Print out the filename and the number of points
                """
                print self.__repr__()

            def __repr__(self):
                """
                This is what is used when calling "print <obj>" or <obj> ENTER
                """
                if self.wl is None:
                    return'Filename: {0}, No data'.format(self.filename)
                else:
                    return'Filename: {0}, number of points: {1}'.format(self.filena

            def __del__(self):
                Stel_Spectrum.spec_count -= 1
        spectra = [] # we create an empty list
        for T in np.linspace(40000, 190000, 4): # this is the list of available tem
            spectra.append(Stel_Spectrum(T=T, logg=6, verbose=True)) # we fill the
        T = np.array([sp.T for sp in spectra])
        F = np.array([sp.get_integ() for sp in spectra])
        for t, f in zip(T, F):
            print('Temperature = {0:.0f}K, Flux = {1:.2e} erg/s/cm2'.format(t, f))
```

```
Overwriting test_2_prof.py
```

```
In [8]: # Need to pip install line-profiler
        ! kernprof -l -v test_2_prof.py
```

```
0040000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0040000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0090000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0090000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0140000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0140000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0190000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0190000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
Temperature = 40000K, Flux = 4.00e+13 erg/s/cm2
Temperature = 90000K, Flux = 1.05e+15 erg/s/cm2
Temperature = 140000K, Flux = 6.93e+15 erg/s/cm2
Temperature = 190000K, Flux = 2.35e+16 erg/s/cm2
Wrote profile results to test_2_prof.py.lprof
Timer unit: 1e-06 s

Total time: 1.7807 s
File: test_2_prof.py
Function: __init__ at line 15

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    15                                           @profile
    16                                           def __init__(self, filename=No
    17                                               """
    18                                               Initialisation of the Stel
    19                                               Parameter:
    20                                                   - filename
    21                                                   - T: temperature in K,
    22                                                   - logg: e.g. 7.5
    23                                                   - verbose: if True, so
    24                                                   The wl variable is an arra
    25                                                   The fl variable is the flu
    26                                                   The variables T and logg a
    27                                                   """
    28         4            9      2.2      0.0          self.verbose = verbose
    29         4            4      1.0      0.0          if filename is None:
    30         4            3      0.8      0.0              if T is not None and l
    31         4            1      0.2      0.0                  self.__T = T # We
    32         4      1780598 445149.5    100.0                  self.logg = logg
    33         4           56     14.0      0.0                  self.filename = '(
    34                                                       else:
    35                                                           raise TypeError("T
    36                                                   else:
    37                                                       self.filename = filena
    38                                                       self.__T = float(filen
    39                                                       self.logg = float(file
    40         4           13      3.2      0.0          Stel_Spectrum.spec_count +
    41         4            4      1.0      0.0          if self.verbose:
    42         4           14      3.5      0.0              print('Instantiation o

Total time: 0.000127 s
File: test_2_prof.py
Function: dlfile at line 44

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    44                                               @profile
    45                                               def dlfile(self):
    46                                                   """
    47                                                   Downloading file if not al
    48                                                   """
    49         4           81     20.2     63.8          if not os.path.exists(self
    50                                                       if self.verbose:
    51                                                           print('Downloading
    52                                                       try:
    53                                                           stel_file = urllib
    54
    55                                                           output = open(self
    56                                                           output.write(stel_
    57                                                           output.close()
    58                                                           self.file_found=Tr
    59                                                       except:
    60                                                           if self.verbose:
    61                                                               print('file {}
    62                                                           self.file_found=Fa
    63                                                   else:
    64         4            3      0.8      2.4              if self.verbose:
    65         4           39      9.8     30.7                  print('{} already
    66         4            4      1.0      3.1              self.file_found=True

Total time: 1.78017 s
File: test_2_prof.py
Function: read_data at line 68

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    68                                           @profile
    69                                           def read_data(self):
    70                                               """
    71                                               read the data from the fil
    72                                               """
    73         4            3      0.8      0.0      if self.file_found:
    74         4      1779602 444900.5    100.0          data = np.genfromtxt(s
    75         4           23      5.8      0.0          self.fl = data['fl']
    76         4            5      1.2      0.0          self.wl = data['wl'] #
    77         4          452    113.0      0.0          self.fl /= 1e8 # F LAM
    78         4            3      0.8      0.0          if self.verbose:
    79         4           82     20.5      0.0              print('Read data f
    80                                               else:
    81                                                   if self.verbose:
    82                                                       print('file not fo
    83                                                   self.wl = None
    84                                                   self.fl = None

Total time: 1.7805 s
File: test_2_prof.py
Function: __setlogg at line 136

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   136                                           @profile
   137                                           def __setlogg(self, value):
   138         4            4      1.0      0.0      try:
   139         4           18      4.5      0.0          self.__logg
   140         4            4      1.0      0.0      except:
   141         4            3      0.8      0.0          self.__logg = -1
   142         4            9      2.2      0.0      if not isinstance(value, 
   143                                                   raise TypeError('logg 
   144         4            9      2.2      0.0      if float(value) not in (-1
   145                                                   raise ValueError('Erro
   146                                                   self.__logg = None
   147         4            4      1.0      0.0      elif self.__logg != value:
   148         4            4      1.0      0.0          self.__logg = value
   149         4           39      9.8      0.0          self.filename = '0{0:0
   150         4          164     41.0      0.0          self.dlfile() # will d
   151         4      1780244 445061.0    100.0          self.read_data() # wil
```

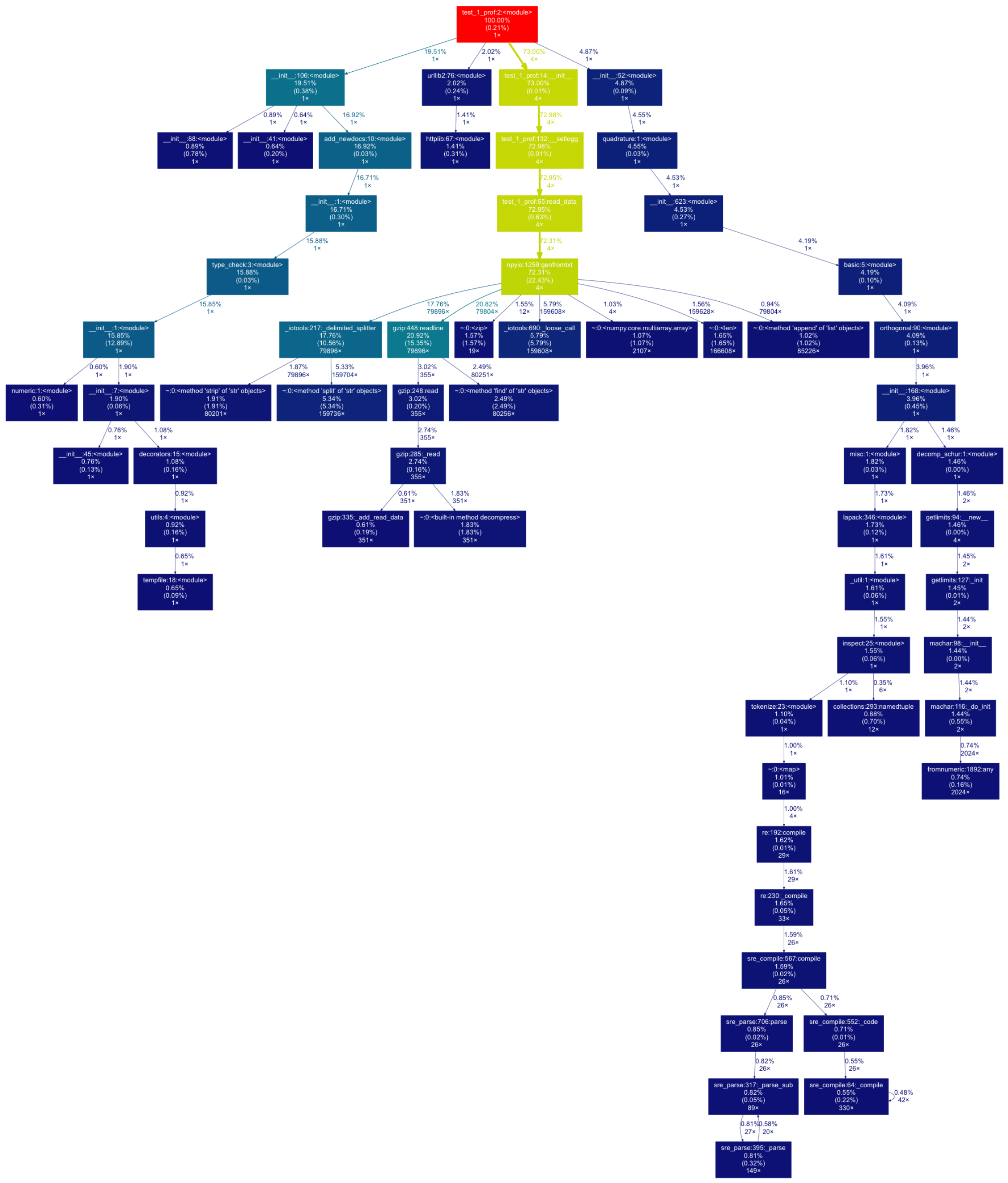
```
In [9]: # Use the test_1 because @profile is not compatible
        ! python -m cProfile -o test_1_prof.prof test_1_prof.py

0040000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0040000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0090000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0090000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0140000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0140000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
0190000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0190000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
Temperature = 40000K, Flux = 4.00e+13 erg/s/cm2
Temperature = 90000K, Flux = 1.05e+15 erg/s/cm2
Temperature = 140000K, Flux = 6.93e+15 erg/s/cm2
Temperature = 190000K, Flux = 2.35e+16 erg/s/cm2
```

```
In [10]: # need to pip install gprof2dot
         # dot is installed by yum install graphviz
         ! gprof2dot -f pstats test_1_prof.prof | dot -Tpng -o test_1-prof.png
```

```
In [11]: Image(filename='test_1-prof.png')

Out[11]:
```



```
In [12]: import pstats
         p = pstats.Stats('test_1_prof.prof')
         p.strip_dirs().sort_stats('time').print_stats(10);

Thu Jun  2 09:37:33 2016    test_1_prof.prof

         934669 function calls (933646 primitive calls) in 0.749 seconds
```

```
   Ordered by: internal time
   List reduced from 858 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        4    0.168    0.042    0.542    0.136 npyio.py:1259(genfromtxt)
    79896    0.115    0.000    0.157    0.000 gzip.py:448(readline)
        3    0.099    0.033    0.245    0.082 __init__.py:1(<module>)
    79896    0.079    0.000    0.133    0.000 _iotools.py:217(_delimited_splitter)
   159608    0.043    0.000    0.043    0.000 _iotools.py:690(_loose_call)
   159736    0.040    0.000    0.040    0.000 {method 'split' of 'str' objects}
    80256    0.019    0.000    0.019    0.000 {method 'find' of 'str' objects}
    80201    0.014    0.000    0.014    0.000 {method 'strip' of 'str' objects}
      351    0.014    0.000    0.014    0.000 {built-in method decompress}
166608/166423    0.012    0.000    0.012    0.000 {len}
```

1.0.2 Profiling the code: RAM memory usage

```
In [13]: %%writefile test_3_prof.py

         import numpy as np
         import os
         import urllib2
         from scipy.integrate import simps
         from memory_profiler import profile

         class Stel_Spectrum(object):
             """
             This object downloads a file from http://astro.uni-tuebingen.de/~rauch
             and is able to make some plots.
             """

             spec_count = 0 # This attibute is at the level of the class, not of th

             @profile
             def __init__(self, filename=None, T=None, logg=None, verbose=False):
                 """
                 Initialisation of the Stel_Spectrum object.
                 Parameter:
                     - filename
                     - T: temperature in K, e.g. 150000
                     - logg: e.g. 7.5
                     - verbose: if True, some info are printed out
                 The wl variable is an array of wavelengths in Angstrom.
                 The fl variable is the flux in erg/s/cm2/A
                 The variables T and logg are properties: changing them will reload
```

```
        """
        self.verbose = verbose
        if filename is None:
            if T is not None and logg is not None:
                self.__T = T # We need to initialize the hidden values, as
                self.logg = logg
                self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz
            else:
                raise TypeError("T and logg must be given")
        else:
            self.filename = filename
            self.__T = float(filename.split('_')[0]) # We need to initiali
            self.logg = float(filename.split('_')[1])
        Stel_Spectrum.spec_count += 1
        if self.verbose:
            print('Instantiation done')

    def dlfile(self):
        """
        Downloading file if not already here. Put it in the current direct
        """
        if not os.path.exists(self.filename):
            if self.verbose:
                print('Downloading {}'.format(self.filename))
            try:
                stel_file = urllib2.urlopen('http://astro.uni-tuebingen.de
                                            self.filename)
                output = open(self.filename,'wb')
                output.write(stel_file.read())
                output.close()
                self.file_found=True
            except:
                if self.verbose:
                    print('file {} not found'.format(self.filename))
                self.file_found=False
        else:
            if self.verbose:
                print('{} already on disk'.format(self.filename))
            self.file_found=True

    def read_data(self):
        """
        read the data from the file
        """
        if self.file_found:
            data = np.genfromtxt(self.filename, comments='*', names='wl, f
            self.fl = data['fl']
            self.wl = data['wl'] # in A
            self.fl /= 1e8 # F LAMBDA  GIVEN IN ERG/CM**2/SEC/CM -> erg/s/
            if self.verbose:
                print('Read data from {}'.format(self.filename))
        else:
            if self.verbose:
                print('file not found {}'.format(self.filename))
            self.wl = None
            self.fl = None

    def plot_spr(self, ax=None, *args, **kwargs):
        """
        Plot the spectrum.
        Parameter:
            - ax: an axis (optionnal). If Noe or absent, axis is created
            - any extra parameter is passed to ax.plot
        """
        if self.wl is None:
            print('No data to plot')
            return
        if ax is None:
            fig, ax = plt.subplots()
        ax.plot(self.wl, self.fl,
                label='T3={0:.0f}, logg={1}'.format(self.T/1e3, self.logg)
                *args, **kwargs) # Here are the transmissions of extra par
        ax.set_yscale('log')
        ax.set_ylim(1e6, 1e14)
        ax.set_xlabel('Wavelength (A)')

    @profile
    def get_integ(self):
        """
        Return the integral of Flambda over lambda, in erg/s/cm2
        """
        if self.wl is None:
            print('No data')
            return None
        return simps(self.fl, self.wl) # perform the integral

    def __getT(self):
        return self.__T

    def __setT(self, value):
        if not isinstance(value, (int, long, float)): # check the type of
            raise TypeError('T must be an integer or a float')
        if float(value) not in np.linspace(40000, 190000, 16): # check the
            raise ValueError('T value must be between 40000 and 190000K, b
        elif self.__T != value:
            self.__T = value
            self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.fo
            self.dlfile() # will download new data
            self.read_data() # will update the data

    def __delT(self):
        print('T is needed')

    T = property(__getT, __setT, __delT, "Stellar effective temperature")

    def __getlogg(self):
        return self.__logg

    def __setlogg(self, value):
        try:
            self.__logg
        except:
            self.__logg = -1
        if not isinstance(value, (int, long, float)):
            raise TypeError('logg must be an integer or a float')
        if float(value) not in (-1., 5., 6., 7. ,8., 9.):
            raise ValueError('Error, logg must be 6, 7, 8, or 9')
            self.__logg = None
        elif self.__logg != value:
            self.__logg = value
            self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.fo
            self.dlfile() # will download new data
            self.read_data() # will update the data

    def __dellogg(self):
        print('logg is needed')

    logg = property(__getlogg, __setlogg, __dellogg, "Stellar logg")

    def print_info(self):
        """
        Print out the filename and the number of points
        """
        print self.__repr__()

    def __repr__(self):
        """
        This is what is used when calling "print <obj>" or <obj> ENTER
        """
        if self.wl is None:
            return'Filename: {0}, No data'.format(self.filename)
        else:
            return'Filename: {0}, number of points: {1}'.format(self.filen
    def __del__(self):
        Stel_Spectrum.spec_count -= 1


sp = Stel_Spectrum(T=100000, logg=6, verbose=True)
print('Temperature = {0:.0f}K, Flux = {1:.2e} erg/s/cm2'.format(sp.T, sp.g
```

```
Overwriting test_3_prof.py
```

```
In [14]: # need to pip install -U memory_profiler
         # need to pip install -U psutil
         !python -m memory_profiler test_3_prof.py
```

```
0100000_6.00_33_50_02_15.bin_0.1.gz already on disk
Read data from 0100000_6.00_33_50_02_15.bin_0.1.gz
Instantiation done
Filename: test_3_prof.py

Line #    Mem usage    Increment   Line Contents
================================================
    16     40.4 MiB      0.0 MiB       @profile
    17                                 def __init__(self, filename=None, T=None, lo
    18                                     """
    19                                     Initialisation of the Stel_Spectrum obje
    20                                     Parameter:
    21                                         - filename
    22                                         - T: temperature in K, e.g. 150000
    23                                         - logg: e.g. 7.5
    24                                         - verbose: if True, some info are pr
    25                                     The wl variable is an array of wavelengt
    26                                     The fl variable is the flux in erg/s/cm2
    27                                     The variables T and logg are properties:
    28                                     """
    29     40.4 MiB      0.0 MiB           self.verbose = verbose
    30     40.4 MiB      0.0 MiB           if filename is None:
    31     40.4 MiB      0.0 MiB               if T is not None and logg is not Non
    32     40.4 MiB      0.0 MiB                   self.__T = T # We need to initia
    33     45.5 MiB      5.1 MiB                   self.logg = logg
    34     45.5 MiB      0.0 MiB                   self.filename = '0{0:06.0f}_{1:.
    35                                         else:
    36                                             raise TypeError("T and logg must
    37                                     else:
    38                                         self.filename = filename
    39                                         self.__T = float(filename.split('_')
    40                                         self.logg = float(filename.split('_'
    41     45.5 MiB      0.0 MiB           Stel_Spectrum.spec_count += 1
    42     45.5 MiB      0.0 MiB           if self.verbose:
```

```
    43     45.5 MiB      0.0 MiB               print('Instantiation done')


Filename: test_3_prof.py

Line #    Mem usage    Increment   Line Contents
================================================
   104     45.5 MiB      0.0 MiB       @profile
   105                                 def get_integ(self):
   106                                     """
   107                                     Return the integral of Flambda over lamb
   108                                     """
   109     45.5 MiB      0.0 MiB           if self.wl is None:
   110                                         print('No data')
   111                                         return None
   112     45.9 MiB      0.5 MiB           return simps(self.fl, self.wl) # perforn


Temperature = 100000K, Flux = 1.79e+15 erg/s/cm2
```

1.0.3 Debugger

From the terminal

```
In [15]: # ! ipython -m pdb test_1_prof.py # from a terminal
```

Breakpoint

```
In [16]: # import pdb # need to call the debugger at the breakpoint
         # Inserting a pdb.set_trace in the __init__ method to stop the program anc

In [17]: %%writefile test_5_pdb.py
         import pdb # This is needed to use the debugger
         import numpy as np
         import os
         import urllib2
         from scipy.integrate import simps

         class Stel_Spectrum(object):
             """
             This object downloads a file from http://astro.uni-tuebingen.de/~rauch
             and is able to make some plots.
             """

             spec_count = 0 # This attibute is at the level of the class, not of th
             def __init__(self, filename=None, T=None, logg=None, verbose=False):
                 """
                 Initialisation of the Stel_Spectrum object.
```

```
        Parameter:
            - filename
            - T: temperature in K, e.g. 150000
            - logg: e.g. 7.5
            - verbose: if True, some info are printed out
        The wl variable is an array of wavelengths in Angstrom.
        The fl variable is the flux in erg/s/cm2/A
        The variables T and logg are properties: changing them will reloa
        """
        pdb.set_trace() # THIS IS A BREAKPOINT
        self.verbose = verbose
        if filename is None:
            if T is not None and logg is not None:
                self.__T = T # We need to initialize the hidden values, a
                self.logg = logg
                self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.g
            else:
                raise TypeError("T and logg must be given")
        else:
            self.filename = filename
            self.__T = float(filename.split('_')[0]) # We need to initiali
            self.logg = float(filename.split('_')[1])
        Stel_Spectrum.spec_count += 1
        if self.verbose:
            print('Instantiation done')

    def dlfile(self):
        """
        Downloading file if not already here. Put it in the current direc
        """
        if not os.path.exists(self.filename):
            if self.verbose:
                print('Downloading {}'.format(self.filename))
            try:
                stel_file = urllib2.urlopen('http://astro.uni-tuebingen.d
                                            self.filename)
                output = open(self.filename,'wb')
                output.write(stel_file.read())
                output.close()
                self.file_found=True
            except:
                if self.verbose:
                    print('file {} not found'.format(self.filename))
                self.file_found=False
        else:
            if self.verbose:
                print('{} already on disk'.format(self.filename))
            self.file_found=True
    def read_data(self):
        """
        read the data from the file
        """
        if self.file_found:
            data = np.genfromtxt(self.filename, comments='*', names='wl, f
            self.fl = data['fl']
            self.wl = data['wl'] # in A
            self.fl /= 1e8 # F LAMBDA  GIVEN IN ERG/CM**2/SEC/CM -> erg/s/
            if self.verbose:
                print('Read data from {}'.format(self.filename))
        else:
            if self.verbose:
                print('file not found {}'.format(self.filename))
            self.wl = None
            self.fl = None

    def plot_spr(self, ax=None, *args, **kwargs):
        """
        Plot the spectrum.
        Parameter:
            - ax: an axis (optionnal). If Noe or absent, axis is created
            - any extra parameter is passed to ax.plot
        """
        if self.wl is None:
            print('No data to plot')
            return
        if ax is None:
            fig, ax = plt.subplots()
        ax.plot(self.wl, self.fl,
                label='T3={0:.0f}, logg={1}'.format(self.T/1e3, self.logg)
                *args, **kwargs) # Here are the transmissions of extra par
        ax.set_yscale('log')
        ax.set_ylim(1e6, 1e14)
        ax.set_xlabel('Wavelength (A)')

    def get_integ(self):
        """
        Return the integral of Flambda over lambda, in erg/s/cm2
        """
        if self.wl is None:
            print('No data')
            return None
        return simps(self.fl, self.wl) # perform the integral

    def __getT(self):
        return self.__T
```

```
def __setT(self, value):
    if not isinstance(value, (int, long, float)): # check the type of
        raise TypeError('T must be an integer or a float')
    if float(value) not in np.linspace(40000, 190000, 16): # check the
        raise ValueError('T value must be between 40000 and 190000K, b
    elif self.__T != value:
        self.__T = value
        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.fo
        self.dlfile() # will download new data
        self.read_data() # will update the data

def __delT(self):
    print('T is needed')

T = property(__getT, __setT, __delT, "Stellar effective temperature")

def __getlogg(self):
    return self.__logg

def __setlogg(self, value):
    try:
        self.__logg
    except:
        self.__logg = -1
    if not isinstance(value, (int, long, float)):
        raise TypeError('logg must be an integer or a float')
    if float(value) not in (-1., 5., 6., 7. ,8., 9.):
        raise ValueError('Error, logg must be 6, 7, 8, or 9')
        self.__logg = None
    elif self.__logg != value:
        self.__logg = value
        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.fo
        self.dlfile() # will download new data
        self.read_data() # will update the data

def __dellogg(self):
    print('logg is needed')

logg = property(__getlogg, __setlogg, __dellogg, "Stellar logg")

def print_info(self):
    """
    Print out the filename and the number of points
    """
    print self.__repr__()

def __repr__(self):
```

```
                """
                This is what is used when calling "print <obj>" or <obj> ENTER
                """
                if self.wl is None:
                    return'Filename: {0}, No data'.format(self.filename)
                else:
                    return'Filename: {0}, number of points: {1}'.format(self.filen

            def __del__(self):
                Stel_Spectrum.spec_count -= 1

        sp = Stel_Spectrum(T=100000, logg=6)
        print 'ending'
        print sp.filename
```

```
Overwriting test_5_pdb.py
```

The commands that can be used once inside the pdb debugger session are: * l(list) Lists the code at the current position * u(p) Walk up the call stack * d(own) Walk down the call stack * n(ext) Execute the next line (does not go down in new functions) * s(tep) Execute the next statement (goes down in new functions) * bt Print the call stack * a Print the local variables * !command Execute the given Python command (by opposition to pdb commands * break N Set a breakpoint at line number N. If no N, list all the breakpoints * disable N Remove the breakpoin number N * c(ontinue) Run until the next breakpoint or the end of the program * return Continues executing until the function is about to execute a return statement, and then it pauses. This gives you time to look at the return value before the function returns.

```
In [18]: %run test_5_pdb.py
```

```
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/Notebooks/test_
-> self.verbose = verbose
(Pdb) cont
ending
0100000_6.00_33_50_02_15.bin_0.1.gz
```