Optimization

November 25, 2015

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

```
Mon Oct 26 16:18:23 2015
```

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 [5]: %%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/TMAF/NLTE/He+C+N+O/
            and is able to make some plots.
            """

            spec_count = 0 # This attibute is at the level of the class, not of the object.
            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 the data
                """
                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 logg is still not de
                        self.logg = logg
                        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self
                    else:
                        raise TypeError("T and logg must be given")
                else:
                    self.filename = filename
                    self.__T = float(filename.split('_')[0]) # We need to initialize the hidden values,
                    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 directory
                """
                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/~rauch/TMAF/NLTE/He+
                                                    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/cm2/A
                    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 parameters to plot
    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 the input
        raise TypeError('T must be an integer or a float')
    if float(value) not in np.linspace(40000, 190000, 16): # check the value of the input
        raise ValueError('T value must be between 40000 and 190000K, by 10000K steps')
    elif self.__T != value:
        self.__T = value
        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self.logg
        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'.format(self.T, self.logg
                    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.filename, len(self.wl))

            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 temperature (check the
            spectra.append(Stel_Spectrum(T=T, logg=6, verbose=True)) # we fill the list with the object
        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 [8]: %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   :       0.71 s.
  System :       0.00 s.
Wall time:       0.71 s.
```

```
In [14]: %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 corresponding functi
```

```
In [12]: %%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/TMAF/NLTE/He+C+N+O/
             and is able to make some plots.
             """

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

             @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 the data
        """
        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 logg is still not d
                self.logg = logg
                self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, sel
            else:
                raise TypeError("T and logg must be given")
        else:
            self.filename = filename
            self.__T = float(filename.split('_')[0]) # We need to initialize the hidden values
            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 directory
        """
        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/~rauch/TMAF/NLTE/He
                                            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
        """
```

```python
        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/cm2/A
            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 parameters to plot
        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 the input
            raise TypeError('T must be an integer or a float')
        if float(value) not in np.linspace(40000, 190000, 16): # check the value of the input
            raise ValueError('T value must be between 40000 and 190000K, by 10000K steps')
        elif self.__T != value:
            self.__T = value
            self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self.log
            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'.format(self.T, self.lo
            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.filename, len(self.wl))

    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 temperature (check the
    spectra.append(Stel_Spectrum(T=T, logg=6, verbose=True)) # we fill the list with the objec
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 [13]: # 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: 4.08095 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=None, T=None, logg=None
    17                                                   """
    18                                                   Initialisation of the Stel_Spectrum object.
    19                                                   Parameter:
    20                                                       - filename
    21                                                       - T: temperature in K, e.g. 150000
    22                                                       - logg: e.g. 7.5
    23                                                       - verbose: if True, some info are printed
    24                                                   The wl variable is an array of wavelengths in
    25                                                   The fl variable is the flux in erg/s/cm2/A
    26                                                   The variables T and logg are properties: chang
    27                                                   """
    28         4            7      1.8      0.0          self.verbose = verbose
    29         4            5      1.2      0.0          if filename is None:
    30         4            5      1.2      0.0              if T is not None and logg is not None:
    31         4            5      1.2      0.0                  self.__T = T # We need to initialize th
    32         4      4080794 1020198.5    100.0                   self.logg = logg
    33         4           86     21.5      0.0                  self.filename = '0{0:06.0f}_{1:.2f}_33_
    34                                                       else:
    35                                                           raise TypeError("T and logg must be gi
    36                                                   else:
```

```
    37                                               self.filename = filename
    38                                               self.__T = float(filename.split('_')[0]) # 
    39                                               self.logg = float(filename.split('_')[1])
    40         4           18      4.5      0.0          Stel_Spectrum.spec_count += 1
    41         4            8      2.0      0.0          if self.verbose:
    42         4           18      4.5      0.0              print('Instantiation done')

Total time: 0.000161 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 already here. Put it i
    48                                               """
    49         4           97     24.2     60.2      if not os.path.exists(self.filename):
    50                                                   if self.verbose:
    51                                                       print('Downloading {}'.format(self.fil
    52                                                   try:
    53                                                       stel_file = urllib2.urlopen('http://as
    54                                                                                  self.filen
    55                                                       output = open(self.filename,'wb')
    56                                                       output.write(stel_file.read())
    57                                                       output.close()
    58                                                       self.file_found=True
    59                                                   except:
    60                                                       if self.verbose:
    61                                                           print('file {} not found'.format(s
    62                                                       self.file_found=False
    63                                               else:
    64         4            6      1.5      3.7          if self.verbose:
    65         4           40     10.0     24.8              print('{} already on disk'.format(self
    66         4           18      4.5     11.2          self.file_found=True

Total time: 4.08014 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 file
    72                                               """
    73         4            5      1.2      0.0      if self.file_found:
    74         4      4079430 1019857.5    100.0           data = np.genfromtxt(self.filename, comme
    75         4           67     16.8      0.0          self.fl = data['fl']
    76         4           11      2.8      0.0          self.wl = data['wl'] # in A
    77         4          532    133.0      0.0          self.fl /= 1e8 # F LAMBDA  GIVEN IN ERG/CM
    78         4            8      2.0      0.0          if self.verbose:
```

```
    79         4           90     22.5      0.0                 print('Read data from {}'.format(self.
    80                                                      else:
    81                                                          if self.verbose:
    82                                                              print('file not found {}'.format(self.
    83                                                          self.wl = None
    84                                                          self.fl = None

Total time: 4.08065 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            6      1.5      0.0      try:
   139         4           27      6.8      0.0          self.__logg
   140         4            5      1.2      0.0      except:
   141         4            7      1.8      0.0          self.__logg = -1
   142         4           14      3.5      0.0      if not isinstance(value, (int, long, float)):
   143                                                   raise TypeError('logg must be an integer o
   144         4           15      3.8      0.0      if float(value) not in (-1., 5., 6., 7. ,8., 9
   145                                                   raise ValueError('Error, logg must be 6, 7
   146                                                   self.__logg = None
   147         4           15      3.8      0.0      elif self.__logg != value:
   148         4            6      1.5      0.0          self.__logg = value
   149         4           50     12.5      0.0          self.filename = '0{0:06.0f}_{1:.2f}_33_50_02
   150         4          230     57.5      0.0          self.dlfile() # will download new data
   151         4      4080271 1020067.8    100.0           self.read_data() # will update the data
```

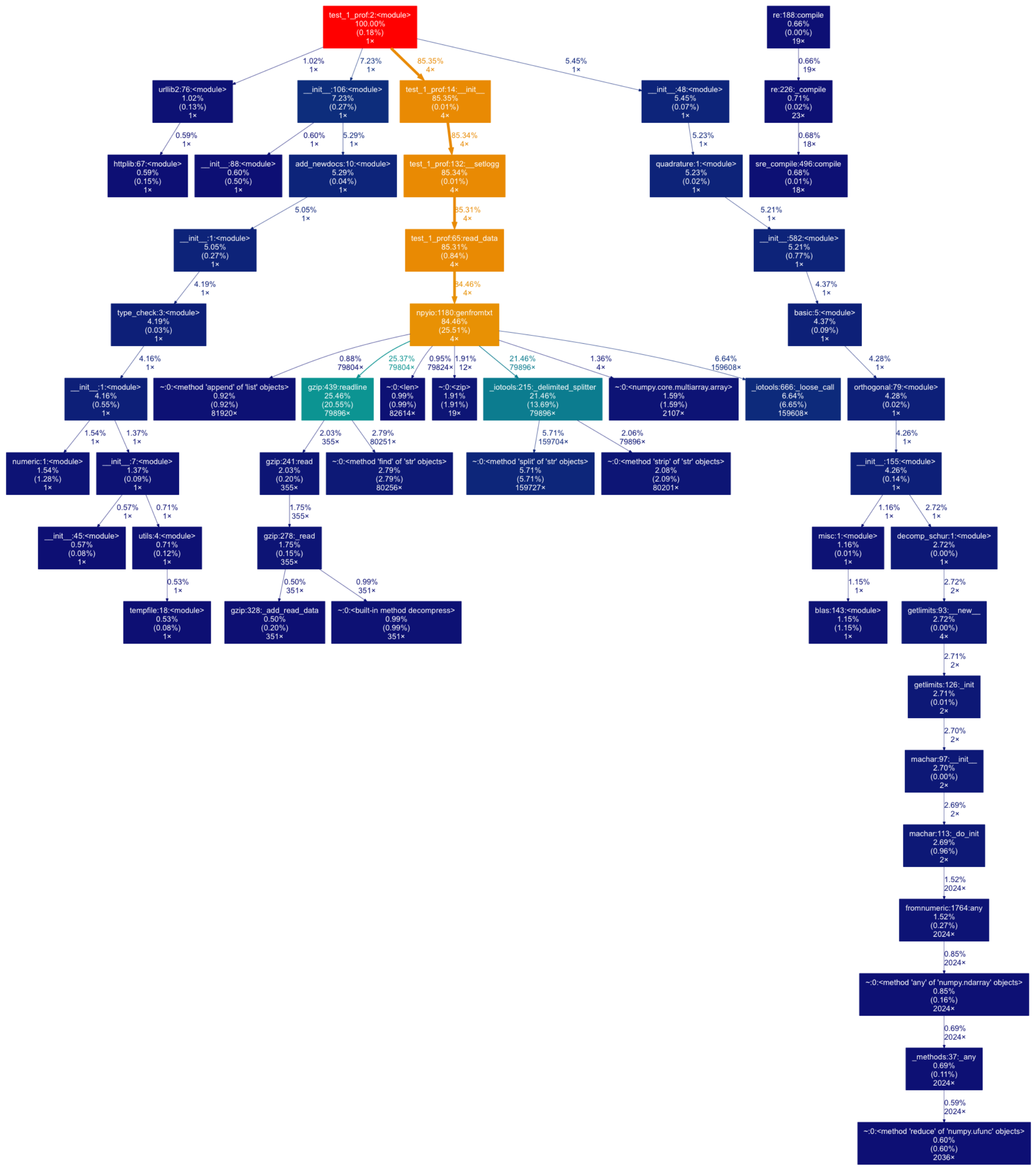
```
In [15]: # Use the test_1 because @profile is not compatible
         ! python -m cProfile -o test_1_prof.prof test_1_prof.py
```

```
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 [16]: # 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 [13]: Image(filename='test_1-prof.png')
```

Out[13]:



In [17]:
```python
import pstats
p = pstats.Stats('test_1_prof.prof')
p.strip_dirs().sort_stats('time').print_stats(10);
```

```
Mon Oct 26 17:05:09 2015    test_1_prof.prof

         835816 function calls (835625 primitive calls) in 1.084 seconds

   Ordered by: internal time
```

```
List reduced from 802 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     4    0.275    0.069    0.917    0.229 npyio.py:1180(genfromtxt)
 79896    0.224    0.000    0.277    0.000 gzip.py:439(readline)
 79896    0.150    0.000    0.235    0.000 _iotools.py:215(_delimited_splitter)
159608    0.070    0.000    0.070    0.000 _iotools.py:666(_loose_call)
159727    0.063    0.000    0.063    0.000 {method 'split' of 'str' objects}
 80256    0.031    0.000    0.031    0.000 {method 'find' of 'str' objects}
 80201    0.023    0.000    0.023    0.000 {method 'strip' of 'str' objects}
    19    0.021    0.001    0.021    0.001 {zip}
  2107    0.018    0.000    0.018    0.000 {numpy.core.multiarray.array}
     1    0.014    0.014    0.017    0.017 numeric.py:1(<module>)
```

1.0.2 Profiling the code: RAM memory usage

```
In [15]: %%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/TMAF/NLTE/He+C+N+O/
             and is able to make some plots.
             """

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

             @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 the data
                 """
                 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 logg is still not d
                         self.logg = logg
                         self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, sel
                     else:
                         raise TypeError("T and logg must be given")
                 else:
```

```
            self.filename = filename
            self.__T = float(filename.split('_')[0]) # We need to initialize the hidden values
            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 directory
        """
        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/~rauch/TMAF/NLTE/He
                                            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/cm2/A
            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 parameters to plot
        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 the input
            raise TypeError('T must be an integer or a float')
        if float(value) not in np.linspace(40000, 190000, 16): # check the value of the input
            raise ValueError('T value must be between 40000 and 190000K, by 10000K steps')
        elif self.__T != value:
            self.__T = value
            self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self.lo
            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'.format(self.T, self.lo
                    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.filename, len(self.wl))

            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.get_integ()))
```

```
Overwriting test_3_prof.py
```

```
In [18]: # need to pip install -U memory_profiler
         # need to pip install -U psutil
         !python -m memory_profiler test_3_prof.py
```

```
Downloading 0100000_6.00_33_50_02_15.bin_0.1.gz
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     25.3 MiB      0.0 MiB       @profile
    17                                 def __init__(self, filename=None, T=None, logg=None, verbose=Fals
    18                                     """
    19                                     Initialisation of the Stel_Spectrum object.
    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 printed out
    25                                   The wl variable is an array of wavelengths in Angstrom.
    26                                   The fl variable is the flux in erg/s/cm2/A
    27                                   The variables T and logg are properties: changing them will 
    28                                   """
    29     25.3 MiB      0.0 MiB           self.verbose = verbose
    30     25.3 MiB      0.0 MiB           if filename is None:
    31     25.3 MiB      0.0 MiB               if T is not None and logg is not None:
    32     25.3 MiB      0.0 MiB                   self.__T = T # We need to initialize the hidden value
    33     32.2 MiB      6.9 MiB                   self.logg = logg
    34     32.2 MiB      0.0 MiB                   self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.
    35                                             else:
    36                                                 raise TypeError("T and logg must be given")
    37                                         else:
    38                                             self.filename = filename
    39                                             self.__T = float(filename.split('_')[0]) # We need to ini
    40                                             self.logg = float(filename.split('_')[1])
    41     32.2 MiB      0.0 MiB           Stel_Spectrum.spec_count += 1
    42     32.2 MiB      0.0 MiB           if self.verbose:
    43     32.2 MiB      0.0 MiB               print('Instantiation done')


Filename: test_3_prof.py

Line #    Mem usage    Increment   Line Contents
================================================
   104     32.2 MiB      0.0 MiB       @profile
   105                                 def get_integ(self):
   106                                     """
   107                                     Return the integral of Flambda over lambda, in erg/s/cm2
   108                                     """
   109     32.2 MiB      0.0 MiB           if self.wl is None:
   110                                         print('No data')
   111                                         return None
   112     32.8 MiB      0.6 MiB           return simps(self.fl, self.wl) # perform the integral


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

1.0.3 Debugger

From the terminal

```
In [17]: # ! ipython -m pdb test_1_prof.py # from a terminal
```

Breakpoint

```
In [18]: # import pdb # need to call the debugger at the breakpoint
         # Inserting a pdb.set_trace in the __init__ method to stop the program and inspect it
```

```
In [23]: %%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/TMAF/NLTE/He+C+N+O/
    and is able to make some plots.
    """

    spec_count = 0 # This attibute is at the level of the class, not of the object.
    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 the data
        """
        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, as logg is still not d
                self.logg = logg
                self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, sel
            else:
                raise TypeError("T and logg must be given")
        else:
            self.filename = filename
            self.__T = float(filename.split('_')[0]) # We need to initialize the hidden values
            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 directory
        """
        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/~rauch/TMAF/NLTE/He
                                            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/cm2/A
            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 parameters to plot
        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 the input
        raise TypeError('T must be an integer or a float')
    if float(value) not in np.linspace(40000, 190000, 16): # check the value of the input
        raise ValueError('T value must be between 40000 and 190000K, by 10000K steps')
    elif self.__T != value:
        self.__T = value
        self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self.lo
        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'.format(self.T, self.lo
        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.filename, len(self.wl))
```

```
            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 [24]: %run test_5_pdb.py
```

```
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(27)__init__()
-> self.verbose = verbose
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(28)__init__()
-> if filename is None:
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(29)__init__()
-> if T is not None and logg is not None:
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(30)__init__()
-> self.__T = T # We need to initialize the hidden values, as logg is still not defined
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(31)__init__()
-> self.logg = logg
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(32)__init__()
-> self.filename = '0{0:06.0f}_{1:.2f}_33_50_02_15.bin_0.1.gz'.format(self.T, self.logg)
(Pdb) n
> /Users/christophemorisset/Google Drive/Pro/Python-MySQL/Notebooks/test_5_pdb.py(39)__init__()
-> Stel_Spectrum.spec_count += 1
(Pdb) self.filename ='tralala'
(Pdb) c
ending
tralala
```

```
In []:
```