Case Study for HitFox

Author: Ihar Marfin

Contact: <marfin@mail.desy.de>

Status: work in progress

Organization: DESY

Version: v0.1

Date: 12/31/13

Dedication

For HitFox Company

Abstract

This document is a description of the tasks and steps I carried out to fulfil them. The introduction into designed classes for C++/python/Java is given. The three different language specifications are used to solve the given problems.

Table of Contents

Note

The solution, presented here, are not optimal! Additional efforts can be requested to find better ways of solving the problems.

1 Task A

1.1 Introduction to Taks A

- Given is an unordered list of numbers (which can appear repeatedly), and we would like to generate a data structure which has the following structure:

```
DivisorsHash { key: 'number' => value: 'divisors of the number in the list' }
```

- Moreover, there some extra requirements.
  - Keys must be stored ordered in the data structure, in a way that when iterated over its keys they will be returned in an ordered manner.
  - List of values for each key must be ordered.

```
Example Input:
 [7,4,2,10,3,6,4,5]

Example Output:
 {
    2 => [1],
    3 => [1],
    4 => [1,2],
    5 => [1],
    6 => [1,2,3],
    7 => [1],
    10 => [1,2,5],
 }
```

- Code the defined data structure and write an algorithm which, given a list of numbers, returns a filled instance of it. Tasks:
- Write class `DivisorsHash`
- Write `DivisorsHash generate(...)` method on class `ProblemA`

1.2 Realization of Taks A

First, we consider the C++ implementation. There are two classes:

- `class DivisorsHash` --> to store divisors of one key
- `class ProblemA` --> to generate a list of divisors for a list of keys

The method `std::vector<int> DivisorsHash::findDivisors ()` performs the finding for all divisors of any non-zero and positivie integer [divisors_in_c].

1.2.1 *Task A in C++*

`cat include/DivisorsHash.h` [code].

```
#include <string>
#include <iostream>
#include <vector>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <math.h>
#include <algorithm>

#ifndef DivisorsHash_H
#define DivisorsHash_H


 bool wayToSortInt(int i, int j) { return i < j; }


 class DivisorsHash {
 public:

/// ctor by default is empty
     DivisorsHash (): key(0){}


     ~DivisorsHash() {}


     void setKey (int i) { key=i; return; }
     int  getKey () const { return key;}
```

```
        std::vector<int> findDivisors () {
             if (key<0) key*=-1;

                unsigned int  keysqrt = (unsigned int) sqrt (key);
                for (unsigned int i=1; i<=keysqrt;i++ )
                    if (key%i==0) { divisors.push_back(i);
                    if (i>1 && i< key/i)
                     divisors.push_back(key/i); }

                std::sort(divisors.begin(),divisors.end(),wayToSortInt);
                return divisors;
     }


         std::vector<int> getDivisors() const { return divisors;}

/// to print out the DivisorsHash
friend std::ostream & operator <<(std::ostream & out,
             const DivisorsHash & right){

             std::stringstream ss;
             std::vector<int> Divisors = right.getDivisors();
             int Key = right.getKey();
             char str[100];
             sprintf(str, "%d => [", Key);
             std::string tmp = str;
             ss<<tmp;
             for (int i=0;i<Divisors.size()-1;i++)
                     {
                             sprintf(str,"%d,",Divisors[i]);
                             tmp=str;
                             ss<<tmp;
                     }
             sprintf(str,"%d]",Divisors[Divisors.size()-1]);
             tmp=str;
             ss<<tmp;
             out<<ss.str();
     }

private:

     int key;
        std::vector<int> divisors;

};


#endif
```

cat include/ProblemA.h

```
#include <string>
#include <iostream>
#include <vector>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <algorithm>
#include "DivisorsHash.h"

#ifndef ProblemA_H
#define ProblemA_H

bool wayToSort(DivisorsHash i, DivisorsHash j) { return i.getKey() < j.getKey(); }

class ProblemA {
public:

/// ctor by default is empty
     ProblemA () {}
     ~ProblemA() {}


     void setKeys (std::vector<int>  _keys) { keys=_keys; return;}

     std::vector<DivisorsHash> generate ( ) const {
             std::vector<DivisorsHash> divisors;
             for (int i=0; i<keys.size();i++) {
              divisors.push_back(DivisorsHash());
              divisors.back().setKey(keys[i]);
              divisors.back().findDivisors();
             }

     return divisors;
     }

/// to print out the ProblemA
friend std::ostream & operator <<(std::ostream & out,
                     const ProblemA & right){

             std::vector<DivisorsHash> Divisors = right.generate();
             std::sort(Divisors.begin(),Divisors.end(),wayToSort);
             out<<"{\n";
             for (int i=0; i< Divisors.size(); i++)
              {out<<"\t"; out<< Divisors[i]; out<<",\n";}
             out<<"}\n";
     }


private:

     std::vector<int> keys;
        std::vector<DivisorsHash> divisors;

};

#endif
```

The steering program `main()` uses the `class DivisorsHash` and `class ProblemA` are in the following manner

```
#include "ProblemA.h"
#include <fstream>
#include <iostream>
#include <vector>


using namespace std;
int main ()
{

     int indexs [] = {7,4,2,10,3,6,18,5};
     size_t size = sizeof(indexs)/sizeof(int);
     std::vector<int> vec_indx (size);
     vec_indx.assign(indexs,indexs+size);


     ProblemA prblmA;

     prblmA.setKeys(vec_indx);

     cout<<prblmA;

return 0;
}
```

It produces the output:

```
{
     2 => [1],
     3 => [1],
     4 => [1,2],
     5 => [1],
     6 => [1,2,3],
     7 => [1],
     10 => [1,2,5],
     18 => [1,2,3,6,9],
}
```

```
def test():
""" test """
 print "How are you"?

 return
```

1.2.2 Task A in Python

The same classes were written using the sysntax of `Python`. Two modules

- `DivisorsHash.py` and
- `ProblemA.py`

were developed to code the task.

`cat scripts/ProblemA/DivisorsHash.py`

```
#! /usr/bin/env python

"""
This is the DivisorsHash  class to  generate   and store divisors of any numeric key

  "DivisorsHash { key: 'number' => value: 'divisors of the number in the list' }"

To test the class :

       ./%prog --test  --debug
```

```
"""

__author__ =  'Igor Marfin'
__copyright__ = "Copyright 2013, DESY HiggsGroup"
__credits__ = ["Igor Marfin", "DESY HiggsGroup"]
__license__ = "GPL"
__version__ = "0.0.1"
__maintainer__ = "Igor Marfin"
__email__ = "marfin@mail.desy.de"
__status__ = "Test"


# import all modules which might be useful


import time
import os
import sys
import commands
import re
from optparse import OptionParser

import types
import math
import unittest
import logging
import inspect

parser2=OptionParser(usage=__doc__)


parser2.add_option("--test",dest="test",
help="to perform test of helper classes",
default=False,action="store_true")
parser2.add_option("--debug",dest="debug",
help="to print debug info",default=False,action="store_true")
parser2.add_option("--tkinter",dest="tkinter",
help="to print debug info",default=False,action="store_true")



if ("pydoc" in str(sys.argv)):
 parser2.add_option("-w",dest="none",default=False,action="store_true")

(options2,args2)=parser2.parse_args()

##################
# Logging Service
##################
#
# logging level
if options2.debug:
 LOG_LEVEL=logging.DEBUG
 logging.basicConfig(stream=sys.stderr, level=LOG_LEVEL)
 logger = logging.getLogger(inspect.stack()[-1][1])
else:
 LOG_LEVEL=logging.WARNING
 logging.basicConfig(stream=sys.stderr, level=LOG_LEVEL)
 logger = logging.getLogger(inspect.stack()[-1][1])


#
#my autolog
#
def autolog(message,mylogger=None):
 """ to print debug messages """

# Get the previous frame in the stack, otherwise it would
# be this function!!!
 func = inspect.currentframe().f_back.f_code
# Dump the message + the name of this function to the log.
 if (mylogger==None):
  logger.debug("%s in %i ==> %s  " % (
  func.co_name,
  func.co_firstlineno,
  message
  ))
 else:
  mylogger.debug("%s in %s:%i ==> %s  " % (
  func.co_name,
  func.co_filename,
  func.co_firstlineno,
  message
  ))
  return
##################

"""
helper classes   and tests

"""

class DivisorsHash(object):
 """ class to store and generate divsors """


 def __init__(self, key=None):
  """ constructor """
  self.logger = logging.getLogger(self.__class__.__name__)
  self.__key=key
  self.__divisors=[]
  pass


 def setKey(self,key=None):
  """ to set the key """
  if key == None:
   self.logger.warning("provide me a numeric key")
   raise ValueError("key is empty ")
  if not isinstance(key, types.IntType):  raise TypeError("key is non-integer")
  self.__key=key
  pass



 def getKey(self):
  """ to get the key """
  return self.__key


 def findDivisors(self):
  """ find and returns all divisors (as an ordered list) """

  if (self.__key == None):
   self.logger.warning("provide me a numeric key")
   raise ValueError("key is empty ")

  if not isinstance(self.__key, types.IntType):  raise TypeError("key is non-integer")

  if self.__key < 0 : self.__key*=-1
  keysqrt = math.sqrt(self.__key)
  for i in range(1,int(keysqrt)+1):
   if self.__key%i==0:
    self.__divisors.append(i)
    if i>1 and i<self.__key/i: self.__divisors.append(self.__key/i)
  self.__divisors.sort()
  return self.__divisors


 def getDivisors(self):
  """ return the list of divisors """
  return self.__divisors


# uncomment the line if you need a public "key" attribute
# key = property(getKey, setKey)




 def __repr__ (self):
  """ return the representation "{ key: 'number' => value: 'divisors of the number in the list' }" """
  return "  key: %d => %s " %(self.getKey(),repr(self.getDivisors()))


 def __str__ (self):
  """ return the representation "{ key: 'number' => value: 'divisors of the number in the list' }" """
  return "  key: %d => %s " %(self.getKey(),repr(self.getDivisors()))
```

```
class MyTests(unittest.TestCase):
 """ to test features """

 def __ini__(self):
  pass

 def test1(self):
  """ to test setKey()/getKey() """
  autolog("test of setKey() ")

  dh = DivisorsHash()
  try:
   dh.setKey(None)
  except Exception as e:
   autolog("Some problems are detected: %s"%e)


  try:
   dh.setKey("test")
  except Exception as e:
   autolog("Some problems are detected: %s"%e)

  try:
   dh.setKey(10)
   autolog("the key is %d"%dh.getKey())
  except Exception as e:
   autolog("Some problems are detected: %s"%e)

  self.failUnless(True)


 def test2(self):
  """ to test findDivisors() """

  autolog("test of findDivisors() ")

  dh = DivisorsHash()

  try:
   dh.setKey(10)
   autolog("the key is %d"%dh.getKey())
   autolog("the list of divisors is %s"%repr(dh.findDivisors()))
  except Exception as e:
   autolog("Some problems are detected: %s"%e)

  self.failUnless(True)



 def test3(self):
  """ to test representation of DivisorsHash """

  autolog("test of __str__() ")

  dh = DivisorsHash()

  try:
   dh.setKey(10)
   dh.findDivisors()
   autolog("%s"%dh)
  except Exception as e:
   autolog("Some problems are detected: %s"%e)
  self.failUnless(True)



"""
main subroutine

"""

if  __name__ == '__main__':
 """ main subroutine """

### read options and prepare settings

 if options2.test and str(__status__).lower()=="test":
  autolog("Test of the helper classes:\n\n\n")
  sys.argv=[sys.argv[0]]
  unittest.main()
```

The following features are exploited in the code:

- `logging` service to dump the debug information;
- rising `Exception` to indicate the wrong input given by user;
- `__str__()` and `__repr__()` methods to properly print out the `DivisorsHash`;
- Implementation of `class unittest.TestCase` to make basic tests of the functionality.

The `class problemA` has `logging` service, `__str__()` and `__repr()__` supports. But `unittest.TestCase` is not introduced. The module `ProbleA` has two `entry points` used in the `setup.py` tool. This will be discussed later.

`cat scripts/ProblemA/ProblemA.py`

```
#! /usr/bin/env python



"""
This is the ProblemA  class to  generate     divisors of  numeric keys in a list:

Example Input:
[7,4,2,10,3,6,4,5]

Example Output:
{
2 => [1],
3 => [1],
4 => [1,2],
5 => [1],
6 => [1,2,3],
7 => [1],
10 => [1,2,5],
}
```

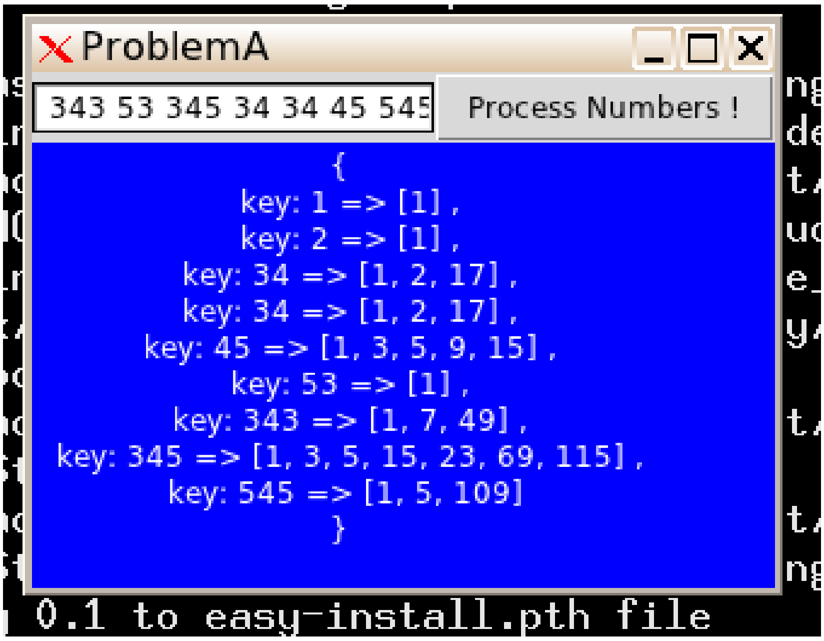
```
To test the class :

       ./%prog   [--debug] [--tkinter] <args>
       where <args> is the space-separated list of numbers:   7 4 2 10 3 6 4 5


"""

__author__ =  'Igor Marfin'
__copyright__ = "Copyright 2013, DESY HiggsGroup"
__credits__ = ["Igor Marfin", "DESY HiggsGroup"]
__license__ = "GPL"
__version__ = "0.0.1"
__maintainer__ = "Igor Marfin"
__email__ = "marfin@mail.desy.de"
__status__ = "Test"


# import all modules which might be useful


import time
import os
import sys
import commands
import re
from optparse import OptionParser

import types
import math
import unittest
import logging
import inspect


parser=OptionParser(usage=__doc__)


parser.add_option("--debug",dest="debug",
help="to print debug info",default=False,action="store_true")
parser.add_option("--tkinter",dest="tkinter",
help="to start gui",default=False,action="store_true")



if ("pydoc" in str(sys.argv)):
 parser.add_option("-w",dest="none",default=False,action="store_true")

(options,args)=parser.parse_args()



##################
```

```
# Logging Service
##################
```

```python
#
# logging level
if options.debug:
 LOG_LEVEL=logging.DEBUG
 logging.basicConfig(stream=sys.stderr, level=LOG_LEVEL)
 logger = logging.getLogger(inspect.stack()[-1][1])
else:
 LOG_LEVEL=logging.WARNING
 logging.basicConfig(stream=sys.stderr, level=LOG_LEVEL)
 logger = logging.getLogger(inspect.stack()[-1][1])


#
#my autolog
#
def autolog(message,mylogger=None):
 """ to print debug messages """

# Get the previous frame in the stack, otherwise it would
# be this function!!!
 func = inspect.currentframe().f_back.f_code
# Dump the message + the name of this function to the log.
 if (mylogger==None):
  logger.debug("%s in %i ==> %s  " % (
  func.co_name,
  func.co_firstlineno,
  message
  ))
 else:
  mylogger.debug("%s in %s:%i ==> %s  " % (
  func.co_name,
  func.co_filename,
  func.co_firstlineno,
  message
  ))
  return
##################




"""
helper classes   and tests

"""

from DivisorsHash import DivisorsHash



class ProblemA(object):
 """ class of ProblemA """
```

```
def __init__(self, keys=None):
 """ constructor """
 self.logger = logging.getLogger(self.__class__.__name__)
 self.__keys=keys
 self.__divisors=[]
 if ( isinstance(keys,types.ListType)): self.__keys.sort()
 pass


def setKeys(self,keys=None):
 """ to set the keys """
 if keys == None:
  self.logger.warning("provide me a numeric key")
  raise ValueError("keys are empty ")
 if ( not isinstance(keys,types.ListType)):
  raise TypeError("It's not  the list of keys")
 if not all(isinstance(key, types.IntType) for key in keys):
  raise TypeError("keys are not integers all")
 self.__keys=keys
 self.__keys.sort()
 pass


def generate(self):
 """ to generate divisors for all input number """
 if self.__keys == None:
  self.logger.warning("provide me a numeric key")
  raise ValueError("keys are empty ")
 if ( not isinstance(self.__keys,types.ListType)):
  raise TypeError("It's not  the list of keys")
 if not all(isinstance(key, types.IntType) for key in self.__keys):
  raise TypeError("keys are not integers all")
 for key in self.__keys:
  dh=DivisorsHash(key)
  dh.findDivisors()
  self.__divisors.append(dh)
 return self.__divisors



def __repr__ (self):
 """ return the list of the representation "{ key:
 'number' => value: 'divisors of the number in the list' }" """
 str1="{\n"
 for i in range(len(self.__keys)):
  if (i<len(self.__keys)-1): str1+=repr(self.__divisors[i])+",\n"
  else: str1+=repr(self.__divisors[i])+"\n } \n"

 return str1



def __str__ (self):
  """ return the list of the representation
  "{ key: 'number' => value: 'divisors of the number in the list' }" """
  str1="{\n"
  for i in range(len(self.__keys)):
   if (i<len(self.__keys)-1): str1+=str(self.__divisors[i])+",\n"
   else: str1+=str(self.__divisors[i])+"\n } \n"

  return str1







"""
main subroutine

"""

def main():
 """ main subroutine """

### read options and prepare settings

 if (len(args)<1):
  print "provide the input list"
  print __doc__
 else:
  keys=[int(x) for x in args]
  pA = ProblemA()
  pA.setKeys(keys)
  pA.generate()
  print pA
 return

def mainTkinter():
 """ main subroutine with GUI """

 import Tkinter

 class simpleapp_tk(Tkinter.Tk):
    def __init__(self,parent):
        Tkinter.Tk.__init__(self,parent)
        self.parent = parent
        self.initialize()
        pass

    def initialize(self):
        self.grid()
        self.title("ProblemA")

        self.entryVariable = Tkinter.StringVar()
        self.labelVariable = Tkinter.StringVar()
        self.entry = Tkinter.Entry(self,textvariable=self.entryVariable)
        self.entry.bind("<Return>", self.OnPressEnter)
        self.entry.grid(column=0,row=0,sticky='EW')

        button = Tkinter.Button(self,text=u"Process Numbers !",
        command=self.OnButtonClick)
        button.grid(column=1,row=0)


        label = Tkinter.Label(self,textvariable=self.labelVariable,
                              anchor="w",fg="white",bg="blue")
        label.grid(column=0,row=1,columnspan=2,sticky='EW')

        self.grid_columnconfigure(0,weight=1)
        self.resizable(True,False)

        pass


    def OnButtonClick(self):
        self.OnPressEnter(self)

    def OnPressEnter(self,event):
        args1 =  self.entryVariable.get().split()
        keys=[int(x) for x in args1]
        pA = ProblemA()
        pA.setKeys(keys)
        pA.generate()
        self.labelVariable.set("%s"%pA)



 app = simpleapp_tk(None)
 app.mainloop()
 return


if  __name__ == '__main__':
 """ main subroutine """

 if options.tkinter: mainTkinter()
 else: main()
```

The option `--tkinter` and the entry point `mainTkinter()` are intended to implement the GUI using Tkinter windows manager [tkineter]. Running the `scripts/ProblemA/ProblemA.py --tkinter` will give the window like one shown in the Figure 1.



```
343 53 345 34 34 45 545
```

```
{
key: 1 => [1] ,
key: 2 => [1] ,
key: 34 => [1, 2, 17] ,
key: 34 => [1, 2, 17] ,
key: 45 => [1, 3, 5, 9, 15] ,
key: 53 => [1] ,
key: 343 => [1, 7, 49] ,
key: 345 => [1, 3, 5, 15, 23, 69, 115] ,
key: 545 => [1, 5, 109]
}
```

Figure 1. Tkinter window realized in `ProblemA`

1.2.3 Task A installation in Python

To install properly `ProblemA.py` and `DivisorsHash.py` modules, we organized them in the package `ProblemA` with `__init__.py` support [python_package]. This is a tree of the package [tree]

```
scripts/ProblemA/
  |---- DivisorsHash.py
  |----DivisorsHash.pyc
  |---- __init__.py
  |---- __init__.pyc
  |---- ProblemA.py
  |---- ProblemA.pyc

0 directories, 6 files
```

2 Section Second

2.1 Bullet Lists

- A bullet list
    - Nested bullet list.
    - Nested item 2.

- Item 2.

  Paragraph 2 of item 2.

  - Nested bullet list.
  - Nested item 2.
    - Third level.

      Item 2.
  - Nested item 3.

2.2 Enumerated Lists

1. Arabic numerals.

   a. lower alpha)

      i. (lower roman)

         A. upper alpha.

            I. upper roman)

2. Lists that don't start at 1:

   3. Three

   4. Four

   C. C

   D. D

   iii. iii

   iv. iv

3. List items may also be auto-enumerated.

2.3 Definition Lists

Term
: Definition

Term : *classifier*
: Definition paragraph 1.

  Definition paragraph 2.

Term
: Definition

3 Section Third

3.1 Literal Blocks

Literal blocks are indicated with a double-colon ("::") at the end of the preceding paragraph (over there `-->`). They can be indented:

```
if literal_block:
    text = 'is left as-is'
    spaces_and_linebreaks = 'are preserved'
    markup_processing = None
```

Or they can be quoted without indentation:

```
>> Great idea!
>
> Why didn't I think of that?
```

3.2 Block Quotes

Block quotes consist of indented body elements:

> My theory by A. Elk. Brackets Miss, brackets. This theory goes as follows and begins now. All brontosauruses are thin at one end, much much thicker in the middle and then thin again at the far end. That is my theory, it is mine, and belongs to me and I own it, and what it is too.
>
> Anne Elk (Miss)

3.3 Tables

Here's a grid table followed by a simple table:

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	Cells may span columns.		
body row 3	Cells may span rows.	• Table cells • contain • body elements.	
body row 4			
body row 5	Cells may also be empty: `-->`		

Inputs		Output
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

4 Section Fourth

4.1 Citations

Here's a reference to the above, [CIT2002] citation.

5 Section Fifth

5.1 Targets

This paragraph is pointed to by the explicit "example" target. A reference can be found under Section Second, above.

Section headers are implicit targets, referred to by name. See Targets.

Explicit external targets are interpolated into references such as "Python" or such as [2].

Targets may be indirect and anonymous. Thus this phrase may also refer to the Targets section.

6 Section Sixth

6.1 More details

you can find more details here [3] and here [4]

7 HOW-TO obtain different format from RST source

- the HTML format:

```
rst2html.py HOW-TO.rst --syntax-highlight="long" > HOW-TO.html
```

- the PDF format:

```
rst2pdf   HOW-TO.rst
```

- the TeX format:

```
rst2latex   HOW-TO.rst  > HOW-TO.tex
```

- the twiki (for CERN twiki) format:

```
cat   HOW-TO.rst | rst2twiki.py  > HOW-TO.twiki
```

- the wiki format:

```
cat   HOW-TO.rst | rst3wiki.py  > HOW-TO.wiki
```

Good Luck!

8 References

CIT2002	Citations are text-labeled footnotes. They may be rendered separately and differently from footnotes.
2	http://www.python.org/
3	http://docutils.sourceforge.net/docs/user/rst/demo.txt http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.txt http://docutils.sourceforge.net/rst.txt
4	http://docutils.sourceforge.net/docs/user/rst/demo.html#inline-markup
code	The code insertion are obtained using the following shell command `cat src/ProblemA.cc | awk '{printf "   %s\n",$0}' > code.rst`
divisors_in_c	The fastest method to find all divisors http://www.cplusplus.com/forum/beginner/11510/
tkineter	Building a basic GUI application step-by-step in Python with Tkinter and wxPython http://sebsauvage.net/python/gui/#import
python_package	How do I write good/correct __init__.py files http://stackoverflow.com/questions/1944569/how-do-i-write-good-correct-init-py-files
tree	The package tree was built by means of `tree scripts/ProblemA`
1	A footnote contains body elements, consistently indented by at least 3 spaces. This is the footnote's second paragraph.
5	Footnotes may be numbered, either manually (as in [1]) or automatically using a "#"-prefixed label. This footnote has a label so it can be referred to from multiple places, both as a footnote reference ([5]) and as a hyperlink reference (label).
6	This footnote is numbered automatically and anonymously using a label of "#" only.
*	Footnotes may also use symbols, specified with a "*" label. Here's a reference to the next footnote: [*].
†	This footnote shows the next symbol in the sequence.