grab_a_python_tutorial

September 26, 2018

1 Python, kind of a tutorial

This notebook will be available here :
https://wiki.student.info.ucl.ac.be/Documentation/Python
... and on Moodle

1.0.1 Note

In order to use this notebook as slides with the possiblity to execute Python code, we need

- iPython3 (+Jupyter if not bundled)
- Python3
- RISE : https://damianavila.github.io/RISE/

Of course, every slide MUST be labelled with the Slides, Subslides, and so on attributes.
To execute the notebook : `$ ipython notebook grab_a_python_tutorial.ipynb`

1.1 What is Python ?

- high-level, general purpose, object-oriented, interactive programming language
- batteries included : **huge** standard library
- **dynamicaly Interpreted**
  python code (.py) -> bytecode (.pyc) -> execution

1.2 Some useful links

- **Python language**
  - Python website: http://www.python.org/
  - Python documentation: https://docs.python.org/3/
  - Python tutorial: https://docs.python.org/3/tutorial/
- **Some tools**
  - Python interpreter (build in the Python distribution)
  - IDLE: https://docs.python.org/3/library/idle.html
  - iPython and iPython notebook: http://ipython.org/
  - Pycharm: http://www.jetbrains.com/pycharm/
  - Eclipse/Pydev: http://www.pydev.org/

1.3 Writing a Python program/script

- use the **.py** file extension

```
In [1]: #!/usr/bin/env python3
        # coding: utf-8

        print("Hello world!")
        print("Hello AI students!")
        print("test testing test")
```

```
Hello world!
Hello AI students!
test testing test
```

- allow execution:

```
chmod +x myfile.py
```

- execute:

```
./myfile.py

python3 myfile.py
```

1.4 Grab-01.py

The Grab project : build a **web indexer**

- capable of
  - analysing a web page to find and count words (indexer)
  - analysing a web page to find external links (crawler)
  - storing all the data into a adapted structure

```
In [ ]: #!/usr/bin/env python3
        # vim: set fileencoding=utf-8 :
        #
        # Grab.py - version 01 - check args

        # 1) args management
        URLPREF = "http://"

        import sys
        def usage():
            print("usage:", sys.argv[0], "<url to parse>")
            sys.exit(1)
```

```
if not (len(sys.argv) == 2 and sys.argv[1].startswith(URLPREF)) :
    usage()

url = sys.argv[1]
print("You ask to grab", url)
```

1.4.1 Variables, basic data types and strings

```
In [2]: # defining variables <- this is a comment
        a = 1
        b = 2
        c = 'string'

        print("a+b=", a + b, "\na-b=", a - b)
        print("a*b=", a * b, "\na/b=", a / b, "\nb^b=", b ** b)
```

```
a+b= 3
a-b= -1
a*b= 2
a/b= 0.5
b^b= 4
```

```
In [3]: d = 0.1
        e = 3.4 + 2j
        print("a is a", type(a), "\nb is a", type(b), "\nc is a", type(c))
        print("d is a", type(d), "\ne is a", type(e))
```

```
a is a <class 'int'>
b is a <class 'int'>
c is a <class 'str'>
d is a <class 'float'>
e is a <class 'complex'>
```

```
In [4]: # variables have no fixed type
        b = 1
        print(b)
        b = "another string"
        print(b)
```

```
1
another string
```

Playing with strings

```
In [5]: # string are iterable
        print(b)
```

```
print(b[0])
print(b[1])
print("Positive indexes", b[0], b[1], b[2])
print("Negative indexes", b[-1], b[-2], b[-3])
print(len(b))
```

```
another string
a
n
Positive indexes a n o
Negative indexes g n i
14
```

In [6]:
```
# take a slice of string
print(b)
print(b[0:2])        # first 2 chars
print(b[0:len(b)-1]) # all but the last
print(b[0:-1])       # all but the last (2nd version)
print(b[:-1])        # all but the last (3rd version)
print(b[1:])         # all but the first
print(b[:])          # a copy of b
```

```
another string
an
another strin
another strin
another strin
nother string
another string
```

In [7]:
```
# strings are *unmutable*
b[1] = "c"
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-7-a7f3a5381ba4> in <module>()
  1 # strings are *unmutable*
----> 2 b[1] = "c"


TypeError: 'str' object does not support item assignment
```

In [8]:
```
# playing with string : + and *
b = "Spam"
```

```
print("b+b=", b + b, "\n" + "b*3=", b*3)
print("Egg, " + (b + ", ") * 4 + b + " and Egg")
```

```
b+b= SpamSpam
b*3= SpamSpamSpam
Egg, Spam, Spam, Spam, Spam, Spam and Egg
```

Inspection and builtin help system

What are string methods ?

```
In [11]: print(dir(b))
         help(b.startswith)
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__forma
Help on built-in function startswith:

startswith(...) method of builtins.str instance
    S.startswith(prefix[, start[, end]]) -> bool

    Return True if S starts with the specified prefix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    prefix can also be a tuple of strings to try.
```

Some string methods

```
In [19]: # calling an instance method
         b = "Spam"
         print(b, b.startswith("Sp"))

         b = "Egg"
         print(b, b.startswith("Sp"))
```

```
Spam True
Egg False
```

```
In [20]: spacious = "   " + b + " \n\n   "
         spacious.strip()
```

```
Out[20]: 'Egg'
```

```
In [21]: b = "Spam"
         b.split('a')
```

```
Out[21]: ['Sp', 'm']

In [22]: b.upper()

Out[22]: 'SPAM'

In [23]: b.lower()

Out[23]: 'spam'
```

1.4.2 Using modules

```
In [24]: # import modules elements into your program
         import sys
         help(sys)

Help on built-in module sys:

NAME
    sys

MODULE REFERENCE
    https://docs.python.org/3.5/library/sys.html

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

DESCRIPTION
    This module provides access to some objects used or maintained by the
    interpreter and to functions that interact strongly with the interpreter.

    Dynamic objects:

    argv -- command line arguments; argv[0] is the script pathname if known
    path -- module search path; path[0] is the script directory, else ''
    modules -- dictionary of loaded modules

    displayhook -- called to show results in an interactive session
    excepthook -- called to handle any uncaught exception other than SystemExit
      To customize printing in an interactive session or to install a custom
      top-level exception handler, assign other functions to replace these.

    stdin -- standard input file object; used by input()
    stdout -- standard output file object; used by print()
    stderr -- standard error object; used for error messages
      By assigning other file objects (or objects that behave like files)
```

```
  to these, it is possible to redirect all of the interpreter's I/O.

last_type -- type of last uncaught exception
last_value -- value of last uncaught exception
last_traceback -- traceback of last uncaught exception
  These three are only available in an interactive session after a
  traceback has been printed.

Static objects:

builtin_module_names -- tuple of module names built into this interpreter
copyright -- copyright notice pertaining to this interpreter
exec_prefix -- prefix used to find the machine-specific Python library
executable -- absolute path of the executable binary of the Python interpreter
float_info -- a struct sequence with information about the float implementation.
float_repr_style -- string indicating the style of repr() output for floats
hash_info -- a struct sequence with information about the hash algorithm.
hexversion -- version information encoded as a single integer
implementation -- Python implementation information.
int_info -- a struct sequence with information about the int implementation.
maxsize -- the largest supported length of containers.
maxunicode -- the value of the largest Unicode code point
platform -- platform identifier
prefix -- prefix used to find the Python library
thread_info -- a struct sequence with information about the thread implementation.
version -- the version of this interpreter as a string
version_info -- version information as a named tuple
__stdin__ -- the original stdin; don't touch!
__stdout__ -- the original stdout; don't touch!
__stderr__ -- the original stderr; don't touch!
__displayhook__ -- the original displayhook; don't touch!
__excepthook__ -- the original excepthook; don't touch!

Functions:

displayhook() -- print an object to the screen, and save it in builtins._
excepthook() -- print an exception and its traceback to sys.stderr
exc_info() -- return thread-safe information about the current exception
exit() -- exit the interpreter by raising SystemExit
getdlopenflags() -- returns flags to be used for dlopen() calls
getprofile() -- get the global profiling function
getrefcount() -- return the reference count for an object (plus one :-)
getrecursionlimit() -- return the max recursion depth for the interpreter
getsizeof() -- return the size of an object in bytes
gettrace() -- get the global debug tracing function
setcheckinterval() -- control how often the interpreter checks for events
setdlopenflags() -- set the flags to be used for dlopen() calls
setprofile() -- set the global profiling function
```

```
    setrecursionlimit() -- set the max recursion depth for the interpreter
    settrace() -- set the global debug tracing function

FUNCTIONS
    __displayhook__ = displayhook(...)
        displayhook(object) -> None

        Print an object to sys.stdout and also save it in builtins._

    __excepthook__ = excepthook(...)
        excepthook(exctype, value, traceback) -> None

        Handle an exception by displaying it with a traceback on sys.stderr.

    call_tracing(...)
        call_tracing(func, args) -> object

        Call func(*args), while tracing is enabled.  The tracing state is
        saved, and restored afterwards.  This is intended to be called from
        a debugger from a checkpoint, to recursively debug some other code.

    callstats(...)
        callstats() -> tuple of integers

        Return a tuple of function call statistics, if CALL_PROFILE was defined
        when Python was built.  Otherwise, return None.

        When enabled, this function returns detailed, implementation-specific
        details about the number of function calls executed. The return value is
        a 11-tuple where the entries in the tuple are counts of:
        0. all function calls
        1. calls to PyFunction_Type objects
        2. PyFunction calls that do not create an argument tuple
        3. PyFunction calls that do not create an argument tuple
           and bypass PyEval_EvalCodeEx()
        4. PyMethod calls
        5. PyMethod calls on bound methods
        6. PyType calls
        7. PyCFunction calls
        8. generator calls
        9. All other calls
        10. Number of stack pops performed by call_function()

    exc_info(...)
        exc_info() -> (type, value, traceback)

        Return information about the most recent exception caught by an except
        clause in the current stack frame or in an older stack frame.
```

```
exit(...)
    exit([status])

    Exit the interpreter by raising SystemExit(status).
    If the status is omitted or None, it defaults to zero (i.e., success).
    If the status is an integer, it will be used as the system exit status.
    If it is another kind of object, it will be printed and the system
    exit status will be one (i.e., failure).

get_coroutine_wrapper(...)
    get_coroutine_wrapper()

    Return the wrapper for coroutine objects set by sys.set_coroutine_wrapper.

getallocatedblocks(...)
    getallocatedblocks() -> integer

    Return the number of memory blocks currently allocated, regardless of their
    size.

getcheckinterval(...)
    getcheckinterval() -> current check interval; see setcheckinterval().

getdefaultencoding(...)
    getdefaultencoding() -> string

    Return the current default string encoding used by the Unicode
    implementation.

getdlopenflags(...)
    getdlopenflags() -> int

    Return the current value of the flags that are used for dlopen calls.
    The flag constants are defined in the os module.

getfilesystemencoding(...)
    getfilesystemencoding() -> string

    Return the encoding used to convert Unicode filenames in
    operating system filenames.

getprofile(...)
    getprofile()

    Return the profiling function set with sys.setprofile.
    See the profiler chapter in the library manual.
```

```
getrecursionlimit(...)
    getrecursionlimit()

    Return the current value of the recursion limit, the maximum depth
    of the Python interpreter stack.  This limit prevents infinite
    recursion from causing an overflow of the C stack and crashing Python.

getrefcount(...)
    getrefcount(object) -> integer

    Return the reference count of object.  The count returned is generally
    one higher than you might expect, because it includes the (temporary)
    reference as an argument to getrefcount().

getsizeof(...)
    getsizeof(object, default) -> int

    Return the size of object in bytes.

getswitchinterval(...)
    getswitchinterval() -> current thread switch interval; see setswitchinterval().

gettrace(...)
    gettrace()

    Return the global debug tracing function set with sys.settrace.
    See the debugger chapter in the library manual.

intern(...)
    intern(string) -> string

    ``Intern'' the given string.  This enters the string in the (global)
    table of interned strings whose purpose is to speed up dictionary lookups.
    Return the string itself or the previously interned string object with the
    same value.

is_finalizing(...)
    is_finalizing()
    Return True if Python is exiting.

set_coroutine_wrapper(...)
    set_coroutine_wrapper(wrapper)

    Set a wrapper for coroutine objects.

setcheckinterval(...)
    setcheckinterval(n)
```

```
        Tell the Python interpreter to check for asynchronous events every
        n instructions.  This also affects how often thread switches occur.

    setdlopenflags(...)
        setdlopenflags(n) -> None

        Set the flags used by the interpreter for dlopen calls, such as when the
        interpreter loads extension modules.  Among other things, this will enable
        a lazy resolving of symbols when importing a module, if called as
        sys.setdlopenflags(0).  To share symbols across extension modules, call as
        sys.setdlopenflags(os.RTLD_GLOBAL).  Symbolic names for the flag modules
        can be found in the os module (RTLD_xxx constants, e.g. os.RTLD_LAZY).

    setprofile(...)
        setprofile(function)

        Set the profiling function.  It will be called on each function call
        and return.  See the profiler chapter in the library manual.

    setrecursionlimit(...)
        setrecursionlimit(n)

        Set the maximum depth of the Python interpreter stack to n.  This
        limit prevents infinite recursion from causing an overflow of the C
        stack and crashing Python.  The highest possible limit is platform-
        dependent.

    setswitchinterval(...)
        setswitchinterval(n)

        Set the ideal thread switching delay inside the Python interpreter
        The actual frequency of switching threads can be lower if the
        interpreter executes long sequences of uninterruptible code
        (this is implementation-specific and workload-dependent).

        The parameter must represent the desired switching delay in seconds
        A typical value is 0.005 (5 milliseconds).

    settrace(...)
        settrace(function)

        Set the global debug tracing function.  It will be called on each
        function call.  See the debugger chapter in the library manual.

DATA
    __stderr__ = <_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF...
    __stdin__ = <_io.TextIOWrapper name='<stdin>' mode='r' encoding='UTF-8...
    __stdout__ = <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF...
```

```
    abiflags = 'm'
    api_version = 1013
    argv = ['/home/presenter/Desktop/tuto-26092018/venv/lib/python3.5/site...
    base_exec_prefix = '/usr'
    base_prefix = '/usr'
    builtin_module_names = ('_ast', '_bisect', '_codecs', '_collections', ...
    byteorder = 'little'
    copyright = 'Copyright (c) 2001-2016 Python Software Foundati...ematis...
    displayhook = <ipykernel.displayhook.ZMQShellDisplayHook object>
    dont_write_bytecode = False
    exec_prefix = '/home/presenter/Desktop/tuto-26092018/venv'
    executable = '/home/presenter/Desktop/tuto-26092018/venv/bin/python3.5...
    flags = sys.flags(debug=0, inspect=0, interactive=0, opt...ing=0, quie...
    float_info = sys.float_info(max=1.7976931348623157e+308, max_...epsilo...
    float_repr_style = 'short'
    hash_info = sys.hash_info(width=64, modulus=2305843009213693...iphash2...
    hexversion = 50660080
    implementation = namespace(_multiarch='x86_64-linux-gnu', cache_t...in...
    int_info = sys.int_info(bits_per_digit=30, sizeof_digit=4)
    last_value = TypeError("'str' object does not support item assignment"...
    maxsize = 9223372036854775807
    maxunicode = 1114111
    meta_path = [<class '_frozen_importlib.BuiltinImporter'>, <class '_fro...
    modules = {'IPython': <module 'IPython' from '/home/presenter/Desktop/...
    path = ['', '/usr/lib/python35.zip', '/usr/lib/python3.5', '/usr/lib/p...
    path_hooks = [<class 'zipimport.zipimporter'>, <function FileFinder.pa...
    path_importer_cache = {'/home/pre/.ipython': FileFinder('/home/pre/.ip...
    platform = 'linux'
    prefix = '/home/presenter/Desktop/tuto-26092018/venv'
    ps1 = 'In : '
    ps2 = '...: '
    ps3 = 'Out: '
    stderr = <ipykernel.iostream.OutStream object>
    stdin = <_io.TextIOWrapper name='<stdin>' mode='r' encoding='UTF-8'>
    stdout = <ipykernel.iostream.OutStream object>
    thread_info = sys.thread_info(name='pthread', lock='semaphore', versio...
    version = '3.5.2 (default, Nov 23 2017, 16:37:01) \n[GCC 5.4.0 2016060...
    version_info = sys.version_info(major=3, minor=5, micro=2, releaseleve...
    warnoptions = []

FILE
    (built-in)
```

```
In [25]: dir(sys)
```

```
Out[25]: ['__displayhook__',
```

```
'__doc__',
'__excepthook__',
'__interactivehook__',
'__loader__',
'__name__',
'__package__',
'__spec__',
'__stderr__',
'__stdin__',
'__stdout__',
'_clear_type_cache',
'_current_frames',
'_debugmallocstats',
'_getframe',
'_home',
'_mercurial',
'_xoptions',
'abiflags',
'api_version',
'argv',
'base_exec_prefix',
'base_prefix',
'builtin_module_names',
'byteorder',
'call_tracing',
'callstats',
'copyright',
'displayhook',
'dont_write_bytecode',
'exc_info',
'excepthook',
'exec_prefix',
'executable',
'exit',
'flags',
'float_info',
'float_repr_style',
'get_coroutine_wrapper',
'getallocatedblocks',
'getcheckinterval',
'getdefaultencoding',
'getdlopenflags',
'getfilesystemencoding',
'getprofile',
'getrecursionlimit',
'getrefcount',
'getsizeof',
'getswitchinterval',
```

```
 'gettrace',
 'hash_info',
 'hexversion',
 'implementation',
 'int_info',
 'intern',
 'is_finalizing',
 'last_traceback',
 'last_type',
 'last_value',
 'maxsize',
 'maxunicode',
 'meta_path',
 'modules',
 'path',
 'path_hooks',
 'path_importer_cache',
 'platform',
 'prefix',
 'ps1',
 'ps2',
 'ps3',
 'set_coroutine_wrapper',
 'setcheckinterval',
 'setdlopenflags',
 'setprofile',
 'setrecursionlimit',
 'setswitchinterval',
 'settrace',
 'stderr',
 'stdin',
 'stdout',
 'thread_info',
 'version',
 'version_info',
 'warnoptions']
```

```
In [26]: type(sys.argv)
```

```
Out[26]: list
```

1.4.3 Lists are mutable and iterable

```
In [27]: l = [1, 2, "a", 3]
         ll = [4.5, l, 6.7]
         print(l)
         print(ll)
         print(l[0])
```

```
print(l[-1])
print(ll[1])
```

```
[1, 2, 'a', 3]
[4.5, [1, 2, 'a', 3], 6.7]
1
3
[1, 2, 'a', 3]
```

Take a slice of a list

```
In [28]: print("l \t= ", l)
         print("l[0:2] \t= ", l[0:2])  # first 2 chars
         print("l[:-1] \t= ", l[:-1])  # all but the last (2nd version)
         print("l[1:] \t= ", l[1:])    # all but the first
         print("l[:] \t= ", l[:])      # a copy of l
```

```
l          =  [1, 2, 'a', 3]
l[0:2]          =  [1, 2]
l[:-1]          =  [1, 2, 'a']
l[1:]          =  [2, 'a', 3]
l[:]          =  [1, 2, 'a', 3]
```

A list does support item assignment

```
In [29]: print("l = ", l)
         print("ll = ", ll)
         l[0] = 42
         l[0] += 1
         print(l)
```

```
l =  [1, 2, 'a', 3]
ll =  [4.5, [1, 2, 'a', 3], 6.7]
[43, 2, 'a', 3]
```

```
In [30]: print("l = ", l)
         print("ll = ", ll)
         ll[1][0] = 0.0 # double index
         print(ll)
         print(l)
```

```
l =  [43, 2, 'a', 3]
ll =  [4.5, [43, 2, 'a', 3], 6.7]
[4.5, [0.0, 2, 'a', 3], 6.7]
[0.0, 2, 'a', 3]
```

```
In [31]: print("l = ", l)
         # even slice assignment
         l[0:2] = ["A", "B", "C"]
         print(l)
```

```
l =  [0.0, 2, 'a', 3]
['A', 'B', 'C', 'a', 3]
```

Some list methods

```
In [32]: # append at the end
         l.append("new element")
         l
```

```
Out[32]: ['A', 'B', 'C', 'a', 3, 'new element']
```

```
In [33]: # pop at index (default -1)
         l.pop(1)
```

```
Out[33]: 'B'
```

```
In [34]: l
```

```
Out[34]: ['A', 'C', 'a', 3, 'new element']
```

```
In [35]: # insert at index
         l.insert(1, "another element")
         print(l)
         dir(l)
```

```
['A', 'another element', 'C', 'a', 3, 'new element']
```

```
Out[35]: ['__add__',
          '__class__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__gt__',
          '__hash__',
          '__iadd__',
```

```
 '__imul__',
 '__init__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

1.4.4 Control flow : if statement

/! Python blocs are delimited by indentation /!

```
In [36]: b = "Holy"
         if b == "Spam":  # line ending with : -> requires an indented block
             pass         # do nothing, in block statement
             print("OK")  # next in block statement
         elif b == "Egg":
             print("KO")
         elif b == "Holy":
             print("The", b, "Grail")
         else:
             print("wtf?")

         print("Out of the IF") # out of block statement
```

```
The Holy Grail
Out of the IF
```

```
In [37]: # boolean operators : and, or, not
         b = "Spam"
         if b.startswith("Sp") and b.endswith("am"):
             print("Spam" + b)
```

```
SpamSpam
```

1.4.5 Control flow : function definition (simplest form)

```
In [38]: def myfunction(arg1, arg2, arg3):  # line ending with : -> requires an indented block
             """ this is a docstring, meant
                 to give information on my function """

             print("This is myfunction")
             print("Args:", arg1, arg2, arg3)
```

```
In [39]: # call myfunction
         ret = myfunction(1, 2, 3)
```

```
This is myfunction
Args: 1 2 3
```

```
In [40]: print(ret)
```

```
None
```

```
In [41]: help(myfunction)
```

```
Help on function myfunction in module __main__:

myfunction(arg1, arg2, arg3)
    this is a docstring, meant
    to give information on my function
```

```
In [42]: def my2ndfunction(arg1, arg2, arg3):  # : => indented block
             """ this is a docstring """

             print("This is my2ndfunction" +
                      "test")
             return "Args: " + str(arg1) + ", " \
                             + str(arg2) + ", " + str(arg3)
```

```
In [43]: ret = my2ndfunction(1, 2, 3)
```

```
This is my2ndfunctiontest
```

```
In [44]: print(type(ret))
         print("ret = ", ret)
```

```
<class 'str'>
ret =  Args: 1, 2, 3
```

1.5 Grab-02.py

```
In [ ]: #!/usr/bin/env python3
        # vim: set fileencoding=utf-8 :
        #
        # Grab.py - version 02 - grab an URL

        # 1) args management
        import sys
        URLPREF = "http://"
        def usage():
            print("usage: ", sys.argv[0], "<url to parse>")
            sys.exit(1)

        if not (len(sys.argv) == 2 and sys.argv[1].startswith(URLPREF)) :
            usage()

        url = sys.argv[1]
        print("You ask to grab", url)

        # 2) retreive the URL
        import urllib.request

        data = urllib.request.urlopen(url)
        print("Here it is:")

        for line in data:
            print(line.decode("utf-8"), end="")
```

1.5.1 Control flow : for statement

```
In [45]: string = "abcdef"

         for letter in string:  # line ending with : -> indented block
             print(letter, end=" ")
```

```
a b c d e f
```

```
In [46]: alist = ["Lancelot", "Galaad", "Bedivere", "Arthur", "Robin"]

         for knight in alist:        # line ending with : -> indented block
             title = "Sir"
             if knight == "Arthur":  # line ending with : -> indented block
                 title = "King"
             print(title, knight)
```

```
Sir Lancelot
Sir Galaad
Sir Bedivere
King Arthur
Sir Robin
```

```
In [47]: list(enumerate(alist))
```

```
Out[47]: [(0, 'Lancelot'), (1, 'Galaad'), (2, 'Bedivere'), (3, 'Arthur'), (4, 'Robin')]
```

```
In [48]: for i, knight in enumerate(alist):
             if i == len(alist) - 1:
                 print("And the last: ", end="")

             print(i, knight)
```

```
0 Lancelot
1 Galaad
2 Bedivere
3 Arthur
And the last: 4 Robin
```

Using range

```
In [49]: list(range(5))
```

```
Out[49]: [0, 1, 2, 3, 4]
```

```
In [50]: for i in range(5):
             print(i)
```

```
0
1
2
3
4
```

```
In [51]: a = ['Mary', 'had', 'a', 'little', 'lamb']

         for i in range(len(a)):   # Don't do this, this is NOT pythonic! Use enumerate if you u
             print(i, a[i])
```

```
0 Mary
1 had
2 a
3 little
4 lamb
```

```
In [52]: range(5)   # range returns an *iterator*

Out[52]: range(0, 5)

In [53]: list(range(3, 10))

Out[53]: [3, 4, 5, 6, 7, 8, 9]

In [54]: list(range(2, 10, 2))

Out[54]: [2, 4, 6, 8]

In [55]: list(range(-10, -100, -30))

Out[55]: [-10, -40, -70]
```

1.5.2 File objects

```
In [56]: filename = "myfile.txt"
         f = open(filename, "w")    # mode = "r", "w", "r+"
                                    # and that can be combined
                                    # with "b" for binary files

In [57]: lines = ["First line", "Second line", "Last line"]

         for line in lines:
             f.write(line + "\n")

         f.close()

In [58]:       import os.path
         os.path.isfile(filename)

Out[58]: True

In [59]: ff = open(filename, "r")
         for line in ff:
             print(line, end='')

         ff.close()
```

```
First line
Second line
Last line
```

```
In [60]: ff = open(filename, "r")
         print(ff.read())
         ff.close()
```

```
First line
Second line
Last line
```

```
In [61]: ff = open(filename, "r")
         ff.readline()
```

```
Out[61]: 'First line\n'
```

```
In [62]: ff.readlines()
```

```
Out[62]: ['Second line\n', 'Last line\n']
```

```
In [63]: ff.close()
```

1.5.3 urllib module

```
In [65]: import urllib.request
         f = urllib.request.urlopen("http://localhost:8000")   # grab the content, return file o
```

```
In [66]: f.readline()
```

```
Out[66]: b'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
```

```
In [67]: f.readline()
```

```
Out[67]: b'\n'
```

```
In [68]: # this is a byte array
         f.readline().decode("utf-8")   # to produce usual string
```

```
Out[68]: '\n'
```

```
In [69]: for line in f:
             print(line.decode("utf-8"), end="")
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Python Documentation Index</title>
  <meta name="keywords" content="None" />
  <meta name="description" content="None" />
  <link rel="alternate" type="application/rss+xml" title="Community Events"
```

```
      href="http://www.python.org/channews.rdf" />
<link rel="alternate" type="application/rss+xml" title="Python Recipes"
      href="http://aspn.activestate.com/ASPN/Cookbook/Python/index_rss" />
<link rel="alternate" type="application/rss+xml" title="Usergroup News"
      href="http://python-groups.blogspot.com/feeds/posts/default" />
<link rel="alternate" type="application/rss+xml" title="Python Screencasts"
      href="http://www.showmedo.com/latestVideoFeed/rss2.0?tag=python" />
<link rel="alternate" type="application/rss+xml" title="Python Podcasts"
      href="http://www.awaretek.com/python/index.xml" />
<link rel="alternate" type="application/rss+xml" title="Foundation News"
      href="http://pyfound.blogspot.com/feeds/posts/default" />
<link rel="alternate" type="application/rss+xml" title="Python Enhancement Proposals"
      href="http://www.python.org/dev/peps/peps.rss" />
<link rel="stylesheet" type="text/css" media="screen" id="screen-switcher-stylesheet"
      href="/styles/screen-switcher-default.css" />
<link rel="stylesheet" type="text/css" media="sc&#82;een"
      href="/styles/netscape4.css" />
<link rel="stylesheet" type="text/css" media="print"
      href="/styles/print.css" />
<link rel="alternate stylesheet" type="text/css" media="screen"
      href="/styles/largestyles.css" title="large text" />
<link rel="alternate stylesheet" type="text/css" media="screen"
      href="/styles/defaultfonts.css" title="default fonts" />

<link rel="search" type="application/opensearchdescription+xml"
      title="Search under the www.python.org Domain"
      href="/search-pysite.xml">

<link rel="search" type="application/opensearchdescription+xml"
      title="Search within the Python Wiki"
      href="/search-pywiki.xml">

<link rel="search" type="application/opensearchdescription+xml"
      title="Search within Python Books at Google Book Search"
      href="/search-pybooks.xml">

<link rel="search" type="application/opensearchdescription+xml"
      title="Search within the Python Documentation"
      href="/search-pydocs.xml">

<link rel="search" type="application/opensearchdescription+xml"
      title="Search for a Module in the Standard Library"
      href="/search-pymodules.xml">

<link rel="search" type="application/opensearchdescription+xml"
      title="Search for Packages inside the Cheeseshop (PyPI)"
      href="/search-pycheese.xml">
```

```
  <link rel="search" type="application/opensearchdescription+xml"
        title="Search Archives of the Main Python Mailing List"
        href="/search-pythonlist.xml">

  <script type="text/javascript" src="/js/iotbs2-key-directors-load.js"></script>
  <script type="text/javascript" src="/js/iotbs2-directors.js"></script>
  <script type="text/javascript" src="/js/iotbs2-core.js"></script>

</head>


<body>
  <!-- Logo -->
  <h1 id="logoheader">
    <a href="/" id="logolink" accesskey="1"><img id="logo" src="/images/python-logo.gif" alt="ho
  </h1>
  <!-- Skip to Navigation -->
  <div class="skiptonav"><a href="#left-hand-navigation" accesskey="2"><img src="/images/trans.g
  <div class="skiptonav"><a href="#content-body" accesskey="3"><img src="/images/trans.gif" id="
  <!-- Utility Menu -->
  <div id="utility-menu">
    <!-- Search Box -->
    <div id="searchbox">
      <form method="get" action="http://google.com/search" id="searchform" name="searchform">
        <div id="search">
          <input type="hidden" id="domains" name="domains" value="www.python.org" />
          <input type="hidden" id="sitesearch" name="sitesearch" value="www.python.org" />
          <input type="hidden" id="sourceid" name="sourceid" value="google-search" />
          <input type="text" class="input-text" name="q" id="q" />
          <input type="submit" value="search" class="input-button" name="submit" id="submit" />
          <a href="/search" class="reference">Advanced Search</a>
        </div>
      </form>
    </div>
    <div id="screen-switcher"></div>
  </div>

  <div id="left-hand-navigation">
    <!-- Main Menu -->
    <div id="menu">
      <ul class="level-one">
            <li>
          <a href="/about/" title="About The Python Language">About</a>
        </li>
            <li>
          <a href="/news/" title="Major Happenings Within the Python Community">News</a>
        </li>
            <li class="selected">
```

```
  <a href="/doc/" title="Tutorials, Library Reference, C API" class="selected">Documenta
    <ul class="level-two">
        <li>
      <a href="http://docs.python.org">Current Docs</a>
      </li>
        <li>
      <a href="http://wiki.python.org/moin/BeginnersGuide">Beginner's Guide</a>
      </li>
        <li>
      <a href="/doc/faq/">FAQs</a>
      </li>
        <li>
      <a href="http://wiki.python.org/moin/">Wiki</a>
      </li>
        <li>
      <a href="/doc/other/">Other Doc Resources</a>
      </li>
        <li>
      <a href="/dev/peps">PEP Index</a>
      </li>
        <li>
      <a href="http://wiki.python.org/moin/PythonBooks">Book List</a>
      </li>
        <li>
      <a href="/doc/topics/">Topic Guides</a>
      </li>
        <li>
      <a href="/doc/newstyle/">New-Style Classes</a>
      </li>
        <li>
      <a href="http://docs.python.org/lib/module-re.html">Regular Expressions</a>
      </li>
        <li>
      <a href="/doc/av">Audio/Visual Talks</a>
      </li>
    </ul>
</li>
    <li>
  <a href="/download/" title="Start Running Python Under Windows, Mac, Linux and Others"
</li>
    <li>
  <a href="/community/" title="Mailing Lists, Jobs, Conferences, SIGs, Online Chats">Com
</li>
    <li>
  <a href="/psf/" title="Python Software Foundation">Foundation</a>
</li>
    <li>
  <a href="/dev/" title="Development of the Python language and website">Core Developmen
```

```
        </li>
            <li>
          <a href="/links/" title="Pointers to Useful Information">Links</a>
        </li>
      </ul>
    </div>

    <!-- Quick Links -->
  </div>

  <div id="content-body">
    <div id="body-main">
      <div id="content">

          <div id="breadcrumb">

            Documentation
          </div>



        <!--utf-8--><!--0.4.1--><h1 class="title">Python Documentation</h1>
<p>Python's standard documentation is substantial; download your own copy or browse it online!</
<ul>
<li><div class="first line-block">
<div class="line"><a class="reference" href="http://docs.python.org/download.html">Download Curr
<div class="line">(Many formats are available, including typeset versions for printing.)</div>
</div>
</li>
<li><p class="first">Search the docs with</p>
<ul class="simple">
<li><a class="reference" href="http://starship.python.net/crew/theller/pyhelp.cgi">pyhelp.cgi</a
</ul>
</li>
<li><p class="first"><a class="reference" href="http://docs.python.org/">Browse Current Document
<a class="reference" href="http://docs.python.org/modindex.html">(Module Index)</a></p>
<ul class="simple">
<li><a class="reference" href="http://docs.python.org/tut/tut.html">Tutorial</a></li>
<li><a class="reference" href="http://docs.python.org/lib/lib.html">Library Reference</a></li>
<li><a class="reference" href="http://docs.python.org/mac/mac.html">Macintosh Reference</a></li>
<li><a class="reference" href="http://docs.python.org/ref/ref.html">Language Reference</a></li>
<li><a class="reference" href="http://docs.python.org/ext/ext.html">Extending and Embedding</a><
<li><a class="reference" href="http://docs.python.org/api/api.html">Python/C API</a></li>
</ul>
</li>
<li><p class="first"><a class="reference" href="http://docs.python.org/dev">Browse the &quot;in-
</li>
<li><p class="first"><a class="reference" href="versions">Previous versions of these documents</
```

```
</li>
</ul>
<p>A variety of additional documentation is available as well:</p>
<ul class="simple">
<li><a class="reference" href="http://wiki.python.org/moin/BeginnersGuide">Beginner's Guide to P
<li><a class="reference" href="http://rgruet.free.fr/#QuickRef">Quick Reference Guide</a> (off-s
<li><a class="reference" href="newstyle">New-style classes (aka descrintro)</a></li>
<li><a class="reference" href="http://www.amk.ca/python/howto/">HOWTO documents</a> (off-site)</
<li><a class="reference" href="/doc/topics/">Topic guides</a></li>
<li><a class="reference" href="other">Other documentation collections</a></li>
<li><a class="reference" href="http://wiki.python.org/moin/PythonBooks">Python Books</a></li>
<li><a class="reference" href="http://www.awaretek.com/book.html">Python book reviews</a> (off-s
<li><a class="reference" href="http://wiki.python.org/moin/PythonPeriodicals">Python Periodicals
<li><a class="reference" href="essays/">Guido's essays</a> and <a class="reference" href="essays
<li><a class="reference" href="http://wiki.python.org/moin/Languages">Non-English documents</a><
<li><a class="reference" href="/dev/peps/">PEPs (Python Enhancement Proposals)</a></li>
</ul>
<p>as well as collections of audio/visual materials and reusable slideshows:</p>
<ul class="simple">
<li><a class="reference" href="/doc/av/">Audio/Visual talks</a></li>
<li><a class="reference" href="/doc/av/5minutes">5-minute Get-Acquainted screencasts</a></li>
<li><a class="reference" href="/doc/slideshows/">Slideshow collections</a></li>
</ul>


      </div>


      <div id="footer">
        <div id="credits">
           <a href="/about/website">Website maintained by the Python community</a><br/>
          <a href="http://www.xs4all.com/" title="Web and email hosting provided by xs4all, Neth
          <a href="http://www.pollenation.net/" title="Design and content management system by P
        </div>
        Copyright &copy; 1990-2007, <a href='/psf'>Python Software Foundation</a><br/>
        <a href="/about/legal">Legal Statements</a>
      </div>


    </div>
  </div>
</body>
</html>
```

2 Grab-03.py

In [ ]:
```
#!/usr/bin/env python3
# vim: set fileencoding=utf-8 :
#
# Grab.py - version 03 - grab an URL, isolates words

# 1) args management
import sys
URLPREF = "http://"
def usage():
    print("usage: ", sys.argv[0], "<url to parse>")
    sys.exit(1)

if not (len(sys.argv) == 2 and sys.argv[1].startswith(URLPREF)) :
    usage()

url = sys.argv[1]
print("You ask to grab", url)

# 2) retreive the URL
import urllib.request

raw_data = urllib.request.urlopen(url)

# 3) Parse it
import re
data = raw_data.read().decode("utf-8")
print(data)
print("Raw data\n" + "="*80)
input()


# 3.1) remove HTML header
data = re.sub(r"<head>.*(.*\n)*.*</head>", "", data,flags=re.MULTILINE)
print(data)
print("No HTML Header\n" + "="*80)
input()


# 3.2) remove HTML tags
data = re.sub(r"<[^<>]*>", "", data, flags=re.MULTILINE)
print(data)
print("No HTML Tag\n" + "="*80)
```

```
input()


# 3.3) remove HTML entities like  
data = re.sub(r"&[^ ;]*;", "", data)
print(data)
print("No HTML entity\n" + "="*80)
input()


# 3.4) remove unwanted chars
data = re.sub(r"[,.;:!?\"'()\[\]{}%$@#/+*_`-]", " ", data)
print(data)
print("No unwanted chars\n" + "="*80)
input()


# 3.5) isolate words
words = data.split()
print(words)
print("\nThe words\n" + "="*80)
input()


# 3.6) remove too short words and any word containing number
words = filter(lambda x: len(x) > 3, words)
words = list(filter(lambda x: not re.search("[0-9]",x), words))
print(words)
print("\nThe words with no digit and more than 3 chars\n" + "="*80)
input()


# 3.7) lower case
words = list(map(lambda x: x.lower(), words))
print(words)
print("\nAll in lowercase\n" + "="*80)
input()


# 3.8) sort it and remove duplicate instances
words = list(set(words))
words.sort() # sort *in place*, returns None !
print(words)
print("\nSorted words, with no duplicate\n" + "="*80)
```

2.0.1 Re module

```
In [70]: import re
         re.findall(r'\bf[a-z]*', 'which foot or hand fell fastest')
```

```
Out[70]: ['foot', 'fell', 'fastest']
```

```
In [71]: re.sub(r'(\b[a-z]+) \1', r'\1', 'cat in the the hat')
```

```
Out[71]: 'cat in the hat'
```

```
In [72]: re.split("(b|e)", "abcdef")
```

```
Out[72]: ['a', 'b', 'cd', 'e', 'f']
```

2.0.2 Match (at the beginning) and search (anywhere)

```
In [73]: re.match("c", "abcdef") # No match
```

```
In [74]: re.search("c", "abcdef")
```

```
Out[74]: <_sre.SRE_Match object; span=(2, 3), match='c'>
```

2.0.3 Lambda forms

```
In [75]: f = lambda x: x+42   # lambda forms are unamed functions
         f
```

```
Out[75]: <function __main__.<lambda>>
```

```
In [76]: f(5)
```

```
Out[76]: 47
```

```
In [77]: def make_incrementor(inc):   # a function that returns functions!
             return lambda x: x + inc

         f = make_incrementor(42)
         f
```

```
Out[77]: <function __main__.make_incrementor.<locals>.<lambda>>
```

```
In [78]: f(5)
```

```
Out[78]: 47
```

```
In [79]: ff = make_incrementor(5)
         ff(5)
```

```
Out[79]: 10
```

```
In [80]: ff = f   # ff is now an alias of f
         ff(5)
```

```
Out[80]: 47
```

2.0.4 Functional tool: filter

filter(function, sequence) apply a filter to a list i.e. returns the list of elements of **sequence** for wich the function **filter** return **True**.

```
In [81]: def is_even(x):
             return x % 2 == 0

         print(list(range(10)))
         print(list(filter(is_even, range(10))))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

```
In [82]: list(filter(lambda x: x%2 != 0, range(10)))

Out[82]: [1, 3, 5, 7, 9]
```

2.0.5 Functionnal tool: map

map(function, sequence) applies **function(item)** for each **item** in the **sequence** and returns a list of the return values.

```
In [83]: def cube(x): return x*x*x

         list(map(cube, range(1, 11)))

Out[83]: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

In [84]: list(map(lambda x: x**3, range(1, 11)))

Out[84]: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

In [85]: seq = range(8)

         def add(x, y): return x+y

         list(map(add, seq, seq))

Out[85]: [0, 2, 4, 6, 8, 10, 12, 14]

In [86]: list(map(lambda x, y: x + y, seq, seq))

Out[86]: [0, 2, 4, 6, 8, 10, 12, 14]
```

2.0.6 Sets

```
In [87]: basket = ["apple", "orange", "apple", "pear", "orange", "banana"]

In [88]: fruit = set(basket)   # create a set without duplicates

         fruit

Out[88]: {'apple', 'banana', 'orange', 'pear'}

In [89]: "orange" in fruit

Out[89]: True

In [90]: "salad" in fruit

Out[90]: False

In [91]: a = set("abracadabra")
         b = set("alacazam")
         print("a \tunique letters in abracadabra: \t",    a)
         print("a - b \tletters in a but not in b: \t",    a - b)
         print("a | b \tletters in either a or b: \t",     a | b)
         print("a & b \tletters in a and b: \t\t",         a & b)
         print("a ^ b \tletters in a or b but not both: ", a ^ b)
```

```
a         unique letters in abracadabra:          {'c', 'd', 'a', 'b', 'r'}
a - b         letters in a but not in b:          {'d', 'b', 'r'}
a | b         letters in either a or b:          {'m', 'a', 'c', 'd', 'b', 'r', 'z', 'l'}
a & b         letters in a and b:                  {'c', 'a'}
a ^ b         letters in a or b but not both:  {'m', 'd', 'b', 'r', 'z', 'l'}
```

3 Grab-04.py

```
In [ ]: #!/usr/bin/env python3
        # vim: set fileencoding=utf-8 :

        # Grab.py - version 04
        #   - grab an URL
        #   - isolate words, count occurencies
        #   - store everything into a dict

        import sys
        import urllib.request
        import re

        # data
        URLPREF = "http://"
```

```python
MIN_WORD_LENGHT = 3

def getWords(data):
    """return the list of tuples with
        - words contained within a HTML string
        - number of occurencies of this word within the string"""

    # remove HTML header
    data = re.sub(r"<head>.*(.*\n)*.*</head>", "", data, flags=re.MULTILINE)
    # remove HTML tags
    data = re.sub(r"<[^<>]*>", "", data, flags=re.MULTILINE)
    # remove HTML entities
    data = re.sub(r"&[^ ;]*;", "", data)
    # remove special chars
    data = re.sub(r"[,.;:!?\"'()\[\]{}%$@#/+*_`-]", " ", data)
    # isolate words
    words = data.split()
    # remove too short words and any word containing number
    words = filter(lambda x: len(x) > 3, words)
    words = list(filter(lambda x: not re.search("[0-9]",x), words))
    # lower case
    words = list(map(lambda x: x.lower(), words))
    # sort it and count occurencies
    words.sort() # sort *in place*, returns None !

    # count occurencies
    # result = [(word1, n1), (word2, n2), ..., (wordN, nN)]
    result = []  # note. [] is also False
    for word in words:
        if result and word == result[-1][0]: # first elt. of the last tuple of result
            result[-1] = (word, result[-1][1] + 1)
            continue
        result.append((word, 1))

    return result

# 1) args management
URLPREF = "http://"
def usage():
    print("usage: ", sys.argv[0], "<url to parse>")
    sys.exit(1)

if not (len(sys.argv) == 2 and sys.argv[1].startswith(URLPREF)) :
    usage()

url = sys.argv[1]
print("You ask to grab", url)
```

```
# 2) retreive the URL
raw_data = urllib.request.urlopen(url)
data = raw_data.read().decode("utf-8")

# 3) isolate words and store them
#    The idea is to produce a structure that can contain the result of
#    many URL analysis: use a dictionary

#   Dict structure:
#       key    ->   value
#       ---         -----
#
#       words  ->   [(nmbr1, url1), (nmbr2, url2), ...]
#
storage = {}
for word, n in getWords(data):
    #### Most obvious version ###

    if not word in storage:
        storage[word] = [(n, url)]
    else:
        storage[word].append((n, url))

    #### it is easier to ask for forgiveness than for permission ###
    #
    # try:
    #     storage[word].append((n, url))
    # except KeyError:
    #     storage[word] = [(n, url)]
    #

print(storage)
```

3.0.1 A word about the truth

```
In [92]: if [] or "" or {} or () or 0 or 0.0 or 0.0 + 0j or None:
             print("Some are True")
         else:
             print("All are False")
```

```
All are False
```

```
In [93]: if 1:
             print("1 is True")
```

```
1 is True
```

3.0.2 Tuples: unmutable list

```
In [94]: t = (12345, 54321, 'hello!')
         t[0]

Out[94]: 12345

In [95]: t

Out[95]: (12345, 54321, 'hello!')

In [96]: # Tuples may be nested:
         u = (t, (1, 2, 3, 4, 5))

         u

Out[96]: ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))

In [97]: u[0] = 4


        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-97-3803e895a7da> in <module>()
    ----> 1 u[0] = 4


        TypeError: 'tuple' object does not support item assignment


In [98]: # Beware of mutable inside unmutable!
         v = (1, [2, 3], 4)

         v

Out[98]: (1, [2, 3], 4)

In [99]: v[1] = range(5)


        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-99-ecf75485165d> in <module>()
    ----> 1 v[1] = range(5)


        TypeError: 'tuple' object does not support item assignment
```

```
In [100]: v[1][0] = "mutables can be modified!"

          v

Out[100]: (1, ['mutables can be modified!', 3], 4)

In [101]: len(v)

Out[101]: 3

In [102]: for elt in v:
              print(elt)

1
['mutables can be modified!', 3]
4


In [103]: empty = ()
          singleton = "hello",    # <-- note the trailing comma

In [104]: len(empty)

Out[104]: 0

In [105]: len(singleton)

Out[105]: 1

In [106]: singleton

Out[106]: ('hello',)

In [107]: ('hello') # <-- without trailing comma

Out[107]: 'hello'
```

3.0.3 Dictionaries: associative tables

```
In [108]: tel = {'jack': 4098, 'sape': 4139}

          tel['guido'] = 4127

          tel

Out[108]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [109]: tel['jack']

Out[109]: 4098
```

```
In [110]: del tel['sape']

          tel['irv'] = 4127

          tel

Out[110]: {'guido': 4127, 'irv': 4127, 'jack': 4098}

In [111]: list(tel.keys())

Out[111]: ['irv', 'jack', 'guido']

In [112]: 'guido' in tel

Out[112]: True
```

Building a **dict** from a **list**

```
In [113]: dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])

Out[113]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [114]: help(__builtin__.dict)

Help on class dict in module builtins:

class dict(object)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a mapping object's
 |      (key, value) pairs
 |  dict(iterable) -> new dictionary initialized as if via:
 |      d = {}
 |      for k, v in iterable:
 |          d[k] = v
 |  dict(**kwargs) -> new dictionary initialized with the name=value pairs
 |      in the keyword argument list.  For example:  dict(one=1, two=2)
 |
 |  Methods defined here:
 |
 |  __contains__(self, key, /)
 |      True if D has a key k, else False.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
```

```
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(...)
 |      D.__sizeof__() -> size of D in memory, in bytes
 |
 |  clear(...)
 |      D.clear() -> None.  Remove all items from D.
 |
 |  copy(...)
 |      D.copy() -> a shallow copy of D
 |
 |  fromkeys(iterable, value=None, /) from builtins.type
```

```
 |      Returns a new dict with keys from iterable and values equal to value.
 |
 |  get(...)
 |      D.get(k[,d]) -> D[k] if k in D, else d.  d defaults to None.
 |
 |  items(...)
 |      D.items() -> a set-like object providing a view on D's items
 |
 |  keys(...)
 |      D.keys() -> a set-like object providing a view on D's keys
 |
 |  pop(...)
 |      D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
 |      If key is not found, d is returned if given, otherwise KeyError is raised
 |
 |  popitem(...)
 |      D.popitem() -> (k, v), remove and return some (key, value) pair as a
 |      2-tuple; but raise KeyError if D is empty.
 |
 |  setdefault(...)
 |      D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D
 |
 |  update(...)
 |      D.update([E, ]**F) -> None.  Update D from dict/iterable E and F.
 |      If E is present and has a .keys() method, then does:  for k in E: D[k] = E[k]
 |      If E is present and lacks a .keys() method, then does:  for k, v in E: D[k] = v
 |      In either case, this is followed by: for k in F:  D[k] = F[k]
 |
 |  values(...)
 |      D.values() -> an object providing a view on D's values
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

```
In [115]: knights = {'gallahad': 'the pure', 'robin': 'the brave'}

          for k, v in knights.items():    # order NOT fixed! BEWARE !!!
              print("Sir", k[0].upper() + k[1:], v)
```

```
Sir Gallahad the pure
Sir Robin the brave
```

4 Grab-05.py

In [ ]:
```
#!/usr/bin/env python3
# vim: set fileencoding=utf-8 :

# Grab.py - version 05
#   - grab an URL
#   - isolate words, count occurencies
#   - store everything into a dict
#   - crawl from the URL and repeat the treatment on linked pages

import sys
import urllib.request
import re

# data
URLPREF = "http://"
MIN_WORD_LENGHT = 3

storage = {}

# tools
def getWords(data):
    """return the list of tuples with
        - words contained within a HTML string
        - number of occurencies of this word within the string"""
    data = re.sub(r"<head>.*(.*\n)*.*</head>", "", data, flags=re.MULTILINE)
    data = re.sub(r"<[^<>]*>", "", data, flags=re.MULTILINE)
    data = re.sub(r"&[^ ;]*;", "", data)
    data = re.sub(r"[,.;:!?\"'()\[\]{}%$@#/+*_`-]", " ", data)
    words = data.split()

    ### List comprehension to do 3 steps in a single one ###
    words = [w.lower() for w in words if len(w) > 3 and not re.search("[0-9]",w)]
    #
    # This has the same effect as
    ## words = filter(lambda x: len(x) > 3, words)
    ## words = list(filter(lambda x: not re.search("[0-9]",x), words))
    ## words = list(map(lambda x: x.lower(), words))

    words.sort() # sort *in place*, returns None !
    result = []
    for word in words:
        if result and word == result[-1][0]: # first elt. of the last tuple of result
            result[-1] = (word, result[-1][1] + 1)
            continue
        result.append((word, 1))
```

```
    return result

def storeWords(words, url):
    """
        Store words into our storage dict, but using dict instead of list

        {word : {URL1: #1, URL2: #2, ...}
    """

    for word, n in words:
        # the most efficient way to initialize/modify a dict entry
        storage.setdefault(word, {})[url] = n
      # |                           |
      # +---------------------------+
      #               |
      #               v
      #      return storage[word]
      #      if it does not exist, add it
      #      with {} as value before returning

# ============ Do something usefull from here ============
# 1) args management
URLPREF = "http://"
def usage():
    print("usage: ", sys.argv[0], "<url to parse>")
    sys.exit(1)

if not (len(sys.argv) == 2 and sys.argv[1].startswith(URLPREF)) :
    usage()

url = sys.argv[1]
print("You ask to grab", url)

# 2) retreive the URL
raw_data = urllib.request.urlopen(url)
data = raw_data.read().decode("utf-8")

# 3) isolate words and store them
storeWords(getWords(data), url)

# 4) isolate external links (very simple implementation)
links = re.findall(r"(?i)<a[^>]*href=\"([^\"#]*)\"", data)
print(links)
input()

# 5) repeat the treatment on these URLs
for link in links:
    if not link.startswith(URLPREF):    # relative links
```

```
            link = re.sub("/[^/]*$", "/", url) + link
        print(" -> grabbing", link)
        try:
            raw_data = urllib.request.urlopen(link).read().decode("utf-8")
            storeWords(getWords(raw_data), link)
        except:
            pass # ignore errors : don't do that ;-)

    words = list(storage.keys())
    words.sort()
    for word in words:
        # get a list of urls mentionning word, sorted by descending number of occurences
        urls = sorted(storage[word].items(), key=lambda x: x[1], reverse=True) # cfr. itemge
        print(word, urls)
```

4.0.1 List comprehensions

```
In [116]: freshfruit = [' banana', ' loganberry \n', 'passion fruit    ']


          [weapon.strip() for weapon in freshfruit]

Out[116]: ['banana', 'loganberry', 'passion fruit']
```

Syntax:
[value with a variable for variable in an iterable **]**

```
In [117]: vec = [2, 4, 6]

          [3*x for x in vec]

Out[117]: [6, 12, 18]

In [118]: [3*x for x in vec if x > 3]  # add conditions

Out[118]: [12, 18]

In [119]: [3*x for x in vec if x < 2]

Out[119]: []

In [120]:
          [[x,x**2] for x in vec] # build (potentialy) complex objects

Out[120]: [[2, 4], [4, 16], [6, 36]]
```

With more than one variable too!

```
In [121]: vec1 = [2, 4, 6]
          vec2 = [4, 3, -9]

          [x*y for x in vec1 for y in vec2] # combine all possible pairs
```

```
Out[121]: [8, 6, -18, 16, 12, -36, 24, 18, -54]
In [122]: [vec1[i]*vec2[i] for i in range(len(vec1))] # explicit association by index
Out[122]: [8, 12, -54]
In [123]: [v1*v2 for v1, v2 in zip(vec1, vec2)] # zip does that
Out[123]: [8, 12, -54]
```

4.0.2 Sorting

```
In [124]: sorted([5, 2, 3, 1, 4]) # sorted returns the sorted object
Out[124]: [1, 2, 3, 4, 5]
In [125]: print( [5, 2, 3, 1, 4].sort() ) # sort method sort in place, returns nothing
None

In [126]: l = [5, 2, 3, 1, 4]
          l.sort()
          l
Out[126]: [1, 2, 3, 4, 5]
In [127]: sorted("copyright")     # works with ANY iterable
Out[127]: ['c', 'g', 'h', 'i', 'o', 'p', 'r', 't', 'y']
```

The **key** parameter

```
In [128]: sorted("This is a test string from Andrew".split())
Out[128]: ['Andrew', 'This', 'a', 'from', 'is', 'string', 'test']
In [129]: sorted("This is a test string from Andrew".split(), key=str.lower)
Out[129]: ['a', 'Andrew', 'from', 'is', 'string', 'test', 'This']
In [130]: student_tuples = [  ('john', 'A', 15),  # (name, grade, age)
                              ('jane', 'B', 12),
                              ('dave', 'B', 10) ]

          # sort by age
          sorted(student_tuples, key=lambda student: student[2])
Out[130]: [('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
```

The **reverse** parameter

```
In [131]: sorted(student_tuples, key=lambda student: student[2], reverse=True)
Out[131]: [('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]
```

5 That's all folks !

With all this in mind, you can understand the Grab example.