

# ***Practical Python***

## ***Input and Output***



Dafydd Gibbon

First South African Workshop in Digital Humanities  
North-Western University, Potchefstroom, SA  
2015-04-04 to 2015-04-05

## *Summary of I/O types*

- Interactive Python environments for learning and testing
  - Python command line
  - Idle
  - IPython
  - Jupyter

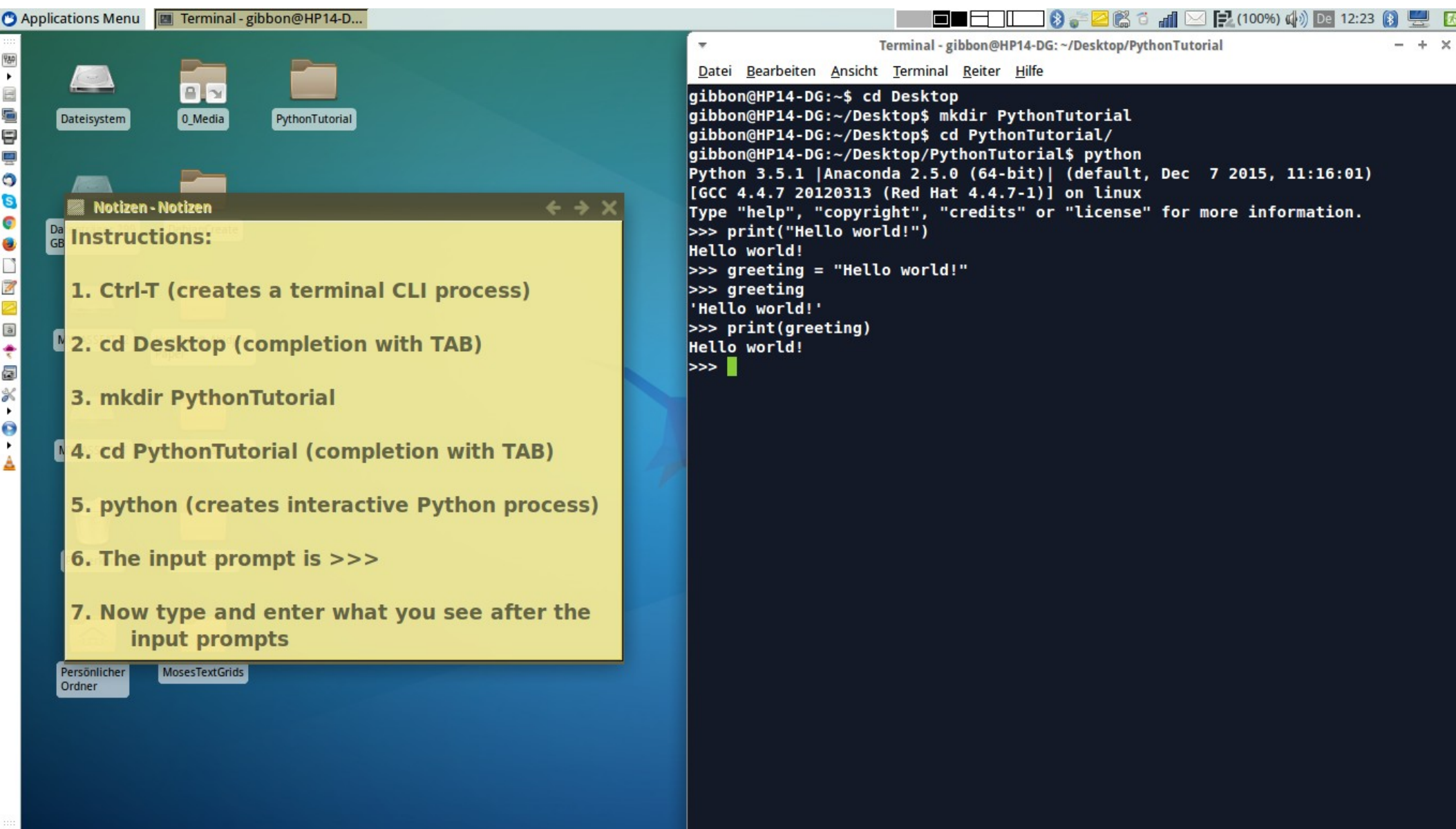
You can practise using these outside the tutorial.

- File editor, file saved as `filename.py`
  - In Python environment:  
`file import`
  - From command line:  
`python filename.py`
  - Linux: from command line, executable with shebang line:  
`filename.py`

- SDK: PyCharm

# Reminder: CLI with interactive Python environment (Linux)

## Interactive window



The screenshot shows a Linux desktop environment. On the left, there is a sidebar with icons for 'Dateisystem', '0\_Media', and 'PythonTutorial'. A window titled 'Notizen - Notizen' is open, displaying a list of instructions. On the right, a terminal window titled 'Terminal - gibbon@HP14-DG: ~/Desktop/PythonTutorial' is open, showing the execution of commands to create a directory, change to it, and run a Python script.

**Notizen - Notizen**

**Instructions:**

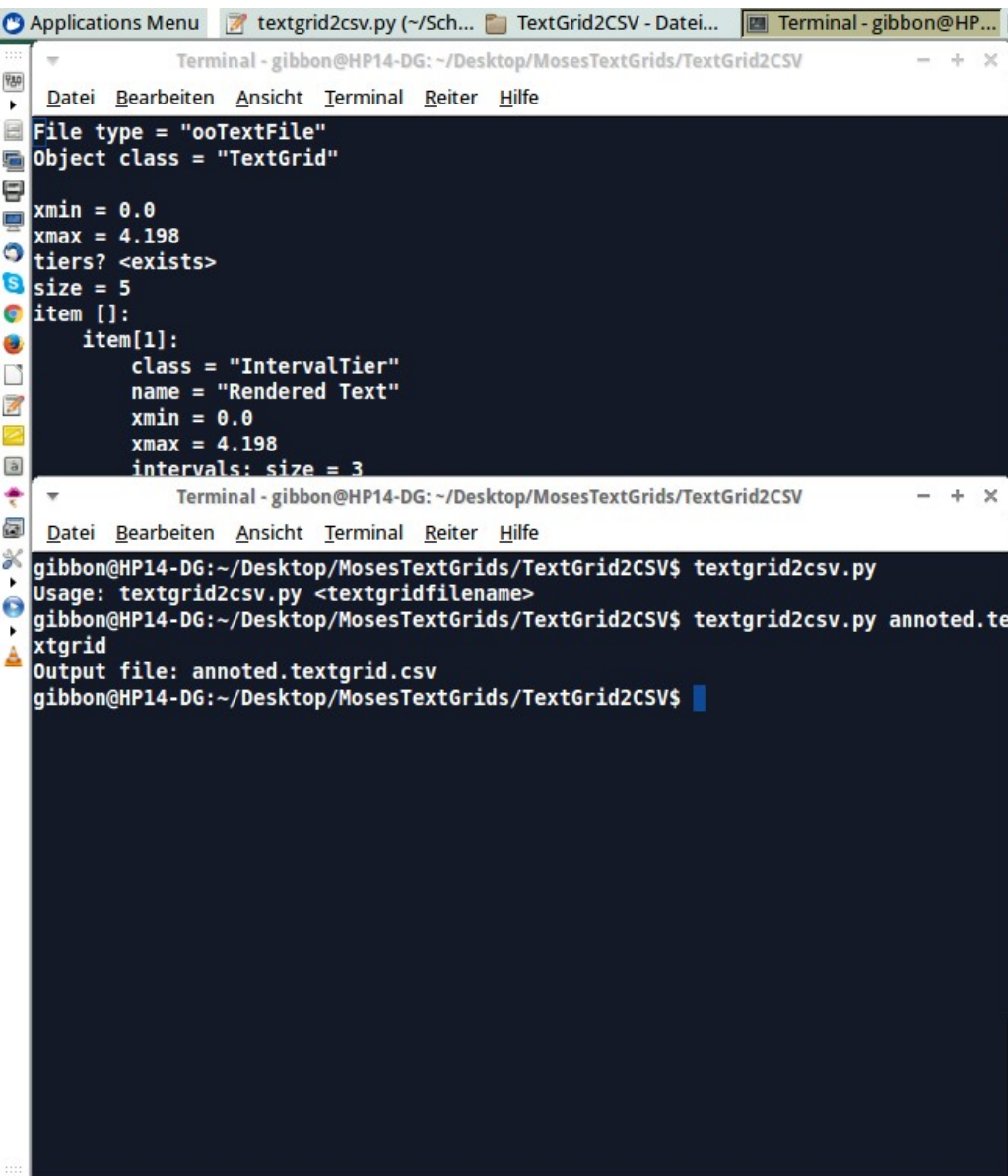
1. Ctrl-T (creates a terminal CLI process)
2. cd Desktop (completion with TAB)
3. mkdir PythonTutorial
4. cd PythonTutorial (completion with TAB)
5. python (creates interactive Python process)
6. The input prompt is >>>
7. Now type and enter what you see after the input prompts

**Terminal - gibbon@HP14-DG: ~/Desktop/PythonTutorial**

```
gibbon@HP14-DG:~$ cd Desktop
gibbon@HP14-DG:~/Desktop$ mkdir PythonTutorial
gibbon@HP14-DG:~/Desktop$ cd PythonTutorial/
gibbon@HP14-DG:~/Desktop/PythonTutorial$ python
Python 3.5.1 |Anaconda 2.5.0 (64-bit)| (default, Dec 7 2015, 11:16:01)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world!")
Hello world!
>>> greeting = "Hello world!"
>>> greeting
'Hello world!'
>>> print(greeting)
Hello world!
>>>
```

# CLI workspace for slightly less retro work

## Data window

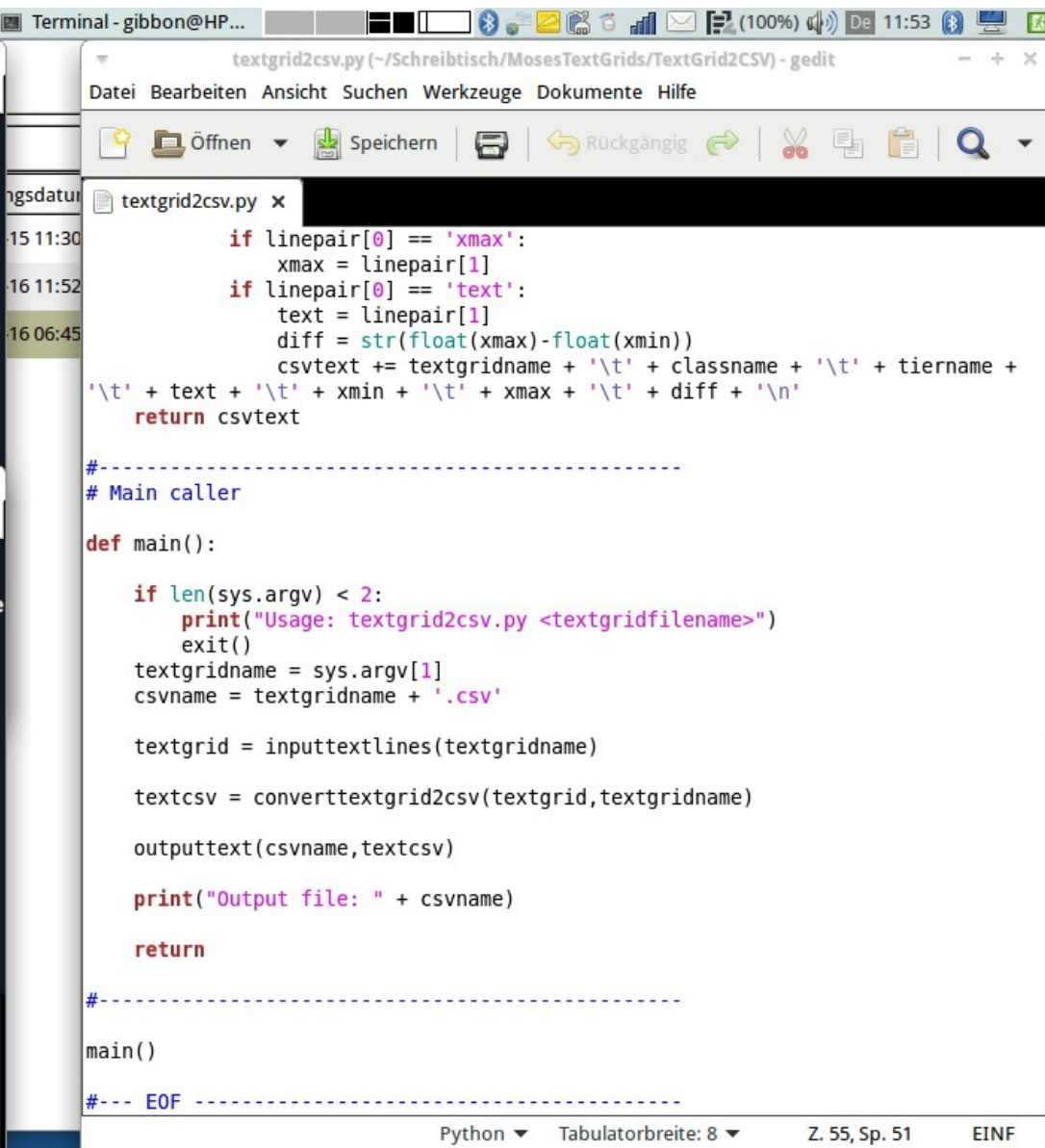


```
Terminal - gibbon@HP14-DG: ~/Desktop/MosesTextGrids/TextGrid2CSV
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0.0
xmax = 4.198
tiers? <exists>
size = 5
item []:
  item[1]:
    class = "IntervalTier"
    name = "Rendered Text"
    xmin = 0.0
    xmax = 4.198
    intervals: size = 3

Terminal - gibbon@HP14-DG: ~/Desktop/MosesTextGrids/TextGrid2CSV
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
gibbon@HP14-DG:~/Desktop/MosesTextGrids/TextGrid2CSV$ textgrid2csv.py
Usage: textgrid2csv.py <textgridfilename>
gibbon@HP14-DG:~/Desktop/MosesTextGrids/TextGrid2CSV$ textgrid2csv.py annotated.txtgrid
Output file: annotated.textgrid.csv
gibbon@HP14-DG:~/Desktop/MosesTextGrids/TextGrid2CSV$
```

## Editor window



```
textgrid2csv.py (~/.Schreibtisch/MosesTextGrids/TextGrid2CSV) - gedit
Datei Bearbeiten Ansicht Suchen Werkzeuge Dokumente Hilfe
Öffnen Speichern Rückgängig

textgrid2csv.py x
    if linepair[0] == 'xmax':
        xmax = linepair[1]
    if linepair[0] == 'text':
        text = linepair[1]
        diff = str(float(xmax)-float(xmin))
        csvtext += textgridname + '\t' + classname + '\t' + tiername +
'\t' + text + '\t' + xmin + '\t' + xmax + '\t' + diff + '\n'
    return csvtext

#-----
# Main caller

def main():
    if len(sys.argv) < 2:
        print("Usage: textgrid2csv.py <textgridfilename>")
        exit()
    textgridname = sys.argv[1]
    csvname = textgridname + '.csv'

    textgrid = inputtextlines(textgridname)

    textcsv = converttextgrid2csv(textgrid, textgridname)

    outputtext(csvname, textcsv)

    print("Output file: " + csvname)

    return

#-----

main()

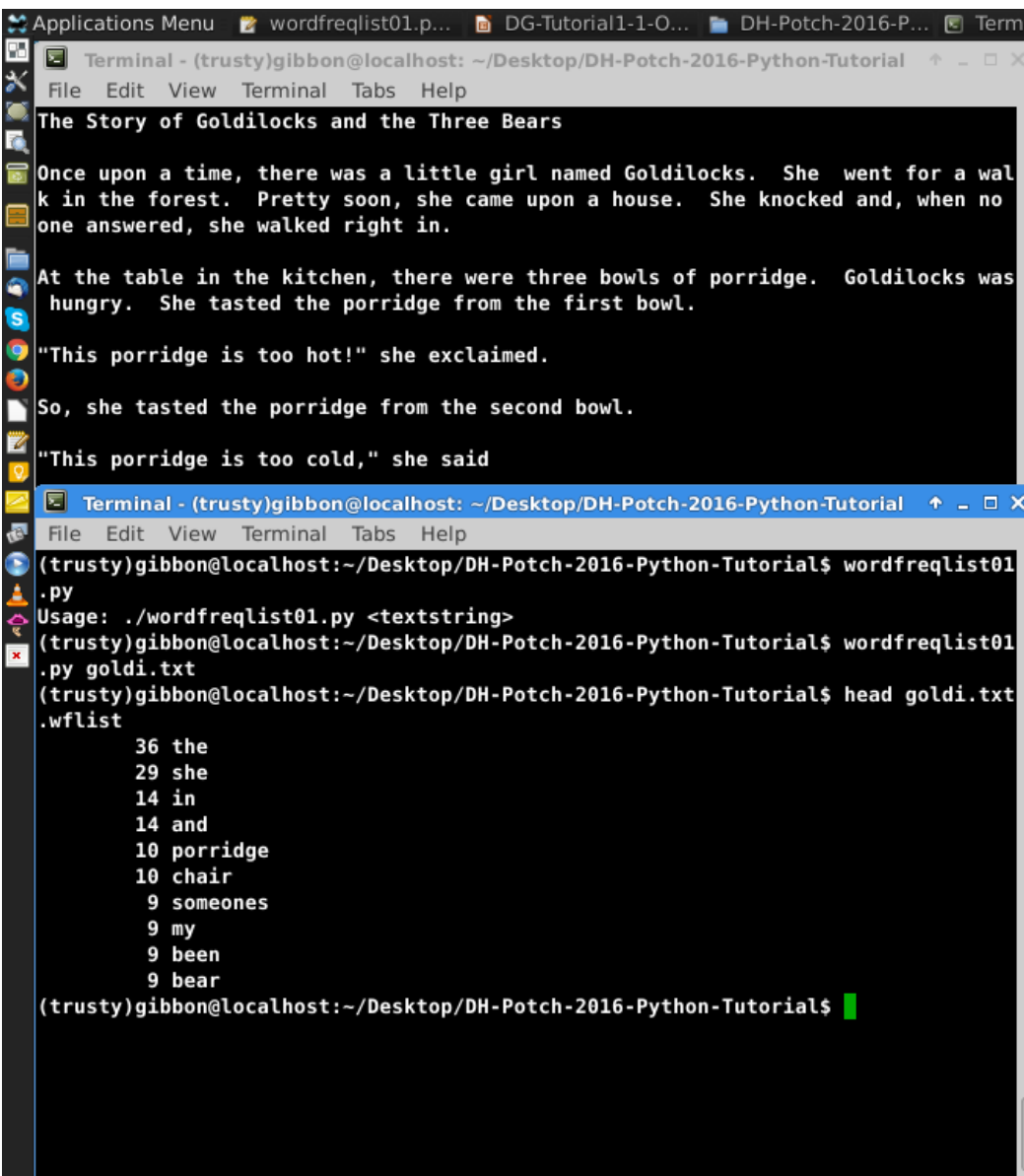
#--- EOF ---

Python Tabulatorbreite: 8 Z. 55, Sp. 51 EINF
```

## Runtime window

# CLI workspace for slightly less retro work

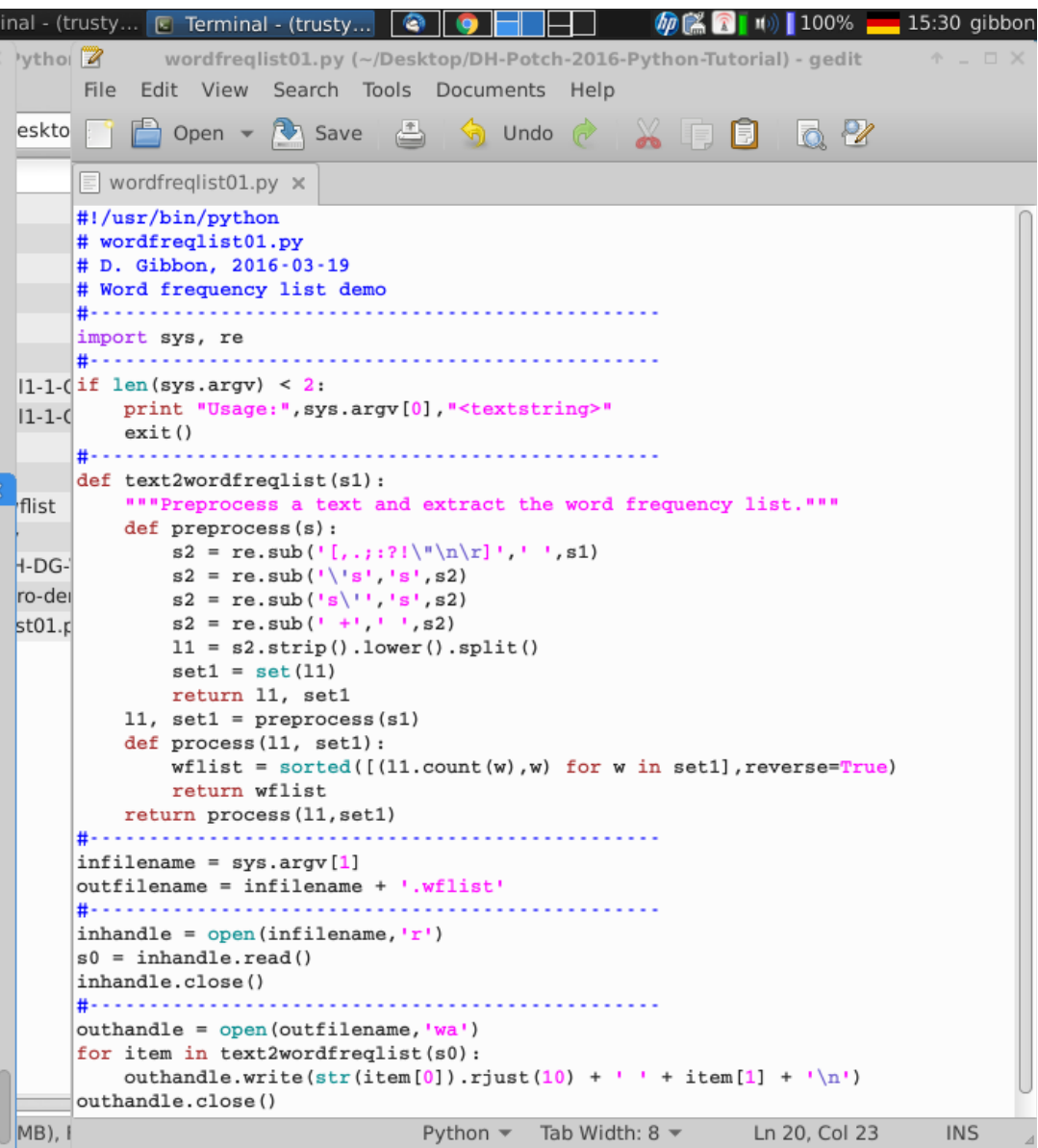
## Data window



The terminal window shows the execution of the `wordfreqlist01.py` script. The script takes the text of "The Story of Goldilocks and the Three Bears" as input and outputs a word frequency list.

```
(trusty)gibbon@localhost: ~/Desktop/DH-Potch-2016-Python-Tutorial$ wordfreqlist01.py
Usage: ./wordfreqlist01.py <textstring>
(trusty)gibbon@localhost:~/Desktop/DH-Potch-2016-Python-Tutorial$ wordfreqlist01.py goldi.txt
(trusty)gibbon@localhost:~/Desktop/DH-Potch-2016-Python-Tutorial$ head goldi.txt.wflist
36 the
29 she
14 in
14 and
10 porridge
10 chair
9 someones
9 my
9 been
9 bear
```

## Editor window



The code editor window displays the source code of the `wordfreqlist01.py` script. The script is a Python program that processes a text file and outputs a word frequency list.

```
#!/usr/bin/python
# wordfreqlist01.py
# D. Gibbon, 2016-03-19
# Word frequency list demo
#-----
import sys, re
#-----
if len(sys.argv) < 2:
    print "Usage:", sys.argv[0], "<textstring>"
    exit()
#-----
def text2wordfreqlist(s1):
    """Preprocess a text and extract the word frequency list."""
    def preprocess(s):
        s2 = re.sub('[,.;?!\\n\\r]', ' ', s1)
        s2 = re.sub('\\s', ' ', s2)
        s2 = re.sub('\\s+', ' ', s2)
        s2 = re.sub(' +', ' ', s2)
        l1 = s2.strip().lower().split()
        set1 = set(l1)
        return l1, set1
    l1, set1 = preprocess(s1)
    def process(l1, set1):
        wflist = sorted([(l1.count(w), w) for w in set1], reverse=True)
        return wflist
    return process(l1, set1)
#-----
infilename = sys.argv[1]
outfilename = infilename + '.wflist'
#-----
inhandle = open(infilename, 'r')
s0 = inhandle.read()
inhandle.close()
#-----
outhandle = open(outfilename, 'w')
for item in text2wordfreqlist(s0):
    outhandle.write(str(item[0]).rjust(10) + ' ' + item[1] + '\n')
outhandle.close()
```

## Runtime window

## *The import statements*

- There are various import statements

```
import matplotlib
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import pyplot
from matplotlib import *
import numpy as np
```

- Note:

- Some libraries have standard abbreviations such as those given above

```
import numpy as np
```

- With a very large standard library such as matplotlib it is usually advisable not to import the whole module.

- With your own module files, e.g. **filename.py**, the filename without the **.py** extension is the module name:

```
import filename
from filename import *          .... etc.
```

# *Input and output of data*

Official Python Tutorial:

<https://docs.python.org/2/tutorial/>

- Interactive input:

- raw input: input an arbitrary string of characters:

```
mystring = raw_input('Input anything: ')\nmystring.upper()\nmystring.lower()
```

- structured Python data input:

```
mynumber = input('Input Python number: ')\nmylist = input('Input Python list:')
```

- Output:

- printing with and without newline:

```
print pythondata, pythondata\nprint pythondata, pythondata,
```



## *Input and output of data*

Official Python Tutorial:

<https://docs.python.org/2/tutorial/>

- Input from text file as text string:

```
filehandle = open(filename.txt, 'r')  
mytext = filehandle.read()
```

```
mytext = open(filename.txt, 'r').read()
```

- Input from text file as list of strings

```
filehandle = open(filename, 'r')  
mystringlist = filehandle.readlines()
```

```
mystringlist = open(filename, 'r').readlines()
```



# *Input and output of data*

Official Python Tutorial:

<https://docs.python.org/2/tutorial/>

- Input from text file as text string:  
`filehandle = open(filename.txt, 'r')`  
`mytext = filehandle.read()`
  - or directly:  
`mytext = open(filename.txt, 'r').read()`
- Creating a list of strings from the input string:  
`textlist = mytext.split('\n')`  
`splitlist = [ x.split() for x in textlist ]`
- Exercise: simple word list  
`swl = ???`

# *Input and output of data*

Official Python Tutorial:

<https://docs.python.org/2/tutorial/>

- Input from text file as text string:  

```
filehandle = open(filename.txt, 'r')  
mytext = filehandle.read()
```

  - or directly:  

```
mytext = open(filename.txt, 'r').read()
```
- Next step: list of strings from the input string:  

```
textlist = mytext.split('\n')  
splitlist = [ x.split() for x in  
mytextlist ]
```
- Exercise: simple word list  

```
swl = sorted(set( [ x.lower() for y in  
splitlist for x in y ] ))
```

# *Input and output of data*

Official Python Tutorial:  
<https://docs.python.org/2/tutorial/>

- Input from file as list of strings (saving)

```
filehandle = open(filename, 'r')  
textlist = filehandle.readlines()
```

```
textlist = open(filename, 'r').readlines()
```

- Remove trailing whitespace (e.g. newlines):

```
textlist = [x.rstrip() for x in textlist]
```

- If the textlist also contains leading whitespace, you can combine methods to remove leading whitespace, too:

```
textlist = [x.rstrip().lstrip() for x in  
textlist]
```

# *Input and output of data*

Official Python Tutorial:

<https://docs.python.org/2/tutorial/>

- Output string to file:
  - first construct your string according to your needs

```
textstring = 'Once upon a time\nThere were  
three bears.'
```

- then

```
filehandle = open(filename, 'w')  
filehandle.write(textstring)
```

- or:

```
open(filename, 'w').write(textstring)
```

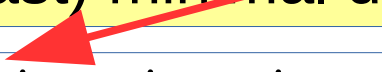
# *Input and output exercises*

Official Python Tutorial:  
<https://docs.python.org/2/tutorial/>

- Write a Python script which will do the following:
  - input a text (which you will write first)
  - make a word list (including punctuation symbols)
  - save the word list in a file, using the conventions:  
    textfilename.txt  
    wordlistfilename.wl.txt
- Use the CLI technique rather than the interactive Python environment technique (see next slide)
- Do the same but instead of a wordlist make a concordance and save the concordance.

## ***Making the Python file into a standalone script***

Add the 'shebang line' (a Unix / Linux feature) as first line, and (at least) minimal documentation (sometimes works with Windows):



```
#!/usr/bin/python
# banana.py
# D. Gibbon 2016.03.19
# Word frequency list demo

a = 'a red one a blue one a yellow one and a banana'

b = a.strip().split()

c = sorted([(b.count(p),p) for p in set(b)],reverse=True)

print c
```

Make the file executable, run the code directly, no python call:

```
chmod 755 banana.py
banana.py
[(4, 'a'), (3, 'one'), (1, 'yellow'), (1, 'red'), (1,
'blue'), (1, 'banana'), (1, 'and')]
```

***Advanced I/O: Interactive web input/output:  
a very basic server-side Python app***



## *First get access to a web server which supports CGI*

- This assumes that you can (learn to) install and configure your own web server:
  - A widely used web server is Apache2
    - To install this on Windows, just download the install file and run the file.
  - Using Ubuntu Linux: you can install Apache2 from the CLI:  
`sudo apt-get install apache2`
  - Actually I use Lighty (lighttpd):  
`sudo apt-get install lighttpd`
- Second, test the webserver with a browser:  
<http://localhost>
- You will need to inspect and maybe edit the configuration file to ensure that `.py` files will be recognised as scripts and not simply as text.
  - Can be tricky, but we can discuss this...

## ***Next step: design an ultra-simply dialogue***

Computer: 'Please input your name:'

User: 'Joe Bloggs'

Computer: 'Hi there, Joe Bloggs!'

***Next step: create an HTML form with filename dialogue.html***

```
<html>
  <head><title>Dialogue</title></head>
  <body>
    <h1>Dialogue test application</h1>
    <form method="post" action="/cgi-bin/dialogue.py">
      Please enter your name:
      <input name="name" size="20" maxlength="20"
value="" type="text">
      <input value="Reset" type="reset">
      <input value="Submit" type="submit">
    </form>
  </body>
</html>
```

***Next step: create a Python script with filename dialogue.py***

```
#!/usr/bin/python
# D. Gibbon, 2016-04-03
# dialogue.py CGI script
# Description: Test dialogue demo.

import cgi, cgitb
cgitb.enable()

fs = cgi.FieldStorage()
username = fs["name"].value

print "Content-type: text/html\n"
print "<html> <head> <title>Dialogue</title> </head>
<body>"
print "Hi there, " + username + "!"
print "</body></html>"
```

## *Install the HTML file and the Python script*

- Put the HTML file and the Python script into the directories required by your web server. These are the directories used by the `lighttpd` server under Linux:
  - for HTML: `/var/www/`
  - for Python: `/usr/lib/cgi-bin/`
- Make the Python script executable; under Linux:  
`chmod 755 dialogue.py`
- Navigate your browser to  
<http://localhost/dialogue.html>
- Enter a name and submit the form. If the server is correctly configured, you should see the dialogue. If not, google for the correct configuration for your web server...

# *A text analysis application* – textanalysis.html

```
<html>
  <head><title>Text analysis</title></head>
  <body>

    <h1>Text analysis application</h1>

    <form method="post" action="/cgi-bin/dialogue.py">

      Copy and paste a text here:
      <textarea name="text" cols="60" rows="20">

        <input value="Reset" type="reset">
        <input value="Submit" type="submit">

    </form>
  </body>
</html>
```

# *A text analysis application* – textanalysis.html

```
<html>
  <head><title>Text analysis</title></head><body>

    <h1>Text analysis application</h1>

    <form method="post" action="/cgi-bin/textanalysis.py">

      <b>Copy and paste a text here:</b>
      <br>
      <textarea name="text" cols="60" rows="20">
      </textarea>
      <br>
      <input value="Reset" type="reset">
      <input value="Submit" type="submit">

    </form>

  </body>
</html>
```



## *A text analysis application – p.1 of textanalysis.html*

- Shebang line and minimal metadata documentation:

```
#!/usr/bin/python
# textanalysis.py
# D. Gibbon, 2016-04-11
# Demo CGI script for text analysis
```

- Import Common Gateway Interface modules:

```
import cgi, cgitb
cgitb.enable()
```

- Decode form input:

```
fs = cgi.FieldStorage()
text = fs["text"].value
```

## *A text analysis application – p.2 of `textanalysis.html`*

- Basic text normalisation to lower case:

```
text = text.lower()
```

- Removal of line breaks

```
text = text.replace('\n', ' ')
```

- Normalisation of punctuation:

```
punct = '.,:;!?"\'=-_()[ ]{$%&/\\+ #<>|@^~'
clist = [' '+c+' ' if c in punct else c for c in text]
text = ''.join(clist)
```

## *A text analysis application – p.3 of `textanalysis.html`*

- Tokenisation and vocabulary extraction:

```
tokenlist = text.split()
typeset = sorted(set(tokenlist))
```

- Check for existence of tokens before counting:

```
if len(tokenlist) == 0:
    tokens = types = ttr = 0
else:
    tokens = len(tokenlist)
    types = len(typeset)
    ttr = 1.0 * types/tokens
```

## *A text analysis application – p.2 of textanalysis.html*

- Creation of output table:

```
table = [ ["Types", types]]
table += [ ["Tokens", tokens]]
table += [ ["TTR", ttr]]
table += [ ["Vocabulary:", ' '.join(typeset)]]
table += [ ["Normalised text:", ' '.join(tokenlist)]]
```

- Transformation and printing of output table as HTML:

```
def printtable(rows):
    print "<table>"
    for row in rows:
        print "<tr>"
        for cell in row:
            print "<td valign=top>", cell, "</td>"
        print "</tr>"
    print "</table>"
    return True
```

## *A text analysis application – p.2 of textanalysis.html*

- HTML output:

```
print "Content-type: text/html\n"
```

```
print "<html>"
```

```
print "<head>"
```

```
print "<title>Text Analysis</title>"
```

```
print "</head>"
```

```
print "<body>"
```

```
printtable(table)
```

```
print "</body>"
```

```
print "</html>"
```

***End of Unit 5***