Lecture 5: jupyter revisited

Yaron Shaposhnik

CIS432 - Predictive analytics using Python

Simon Business School, Spring 2021

Agenda

- We are already working with jupyter
  - web-interface for creating formatted documents that run python code
  - our main working environment throughout the course
- Goal: Learn about advanced features of jupyter
  - OS Shell
  - Magic commands
  - Alternative ways of working with Python

The OS shell

- Earlier with saw this:

In [1]:
```
# show current working directory
!pwd
```

```
C:\Users\Yaron\Dropbox\Projects\Teaching\2021\Lectures\[Module2] Programming i
n Python
```

- This is not a Python command!
- What is it? what does it have to do with Python?

OS shell

- "An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs." Wikipedia
- OS Shell: interface to the OS
- Include a variety of useful programs
- Examples
  - Browsing: ls, pwd, cd
  - Files: cp, rm, mv, mkdir/md,
  - Text: cat, cut, head, sort, uniq, wc, grep,
  - But also: plotting, image conversion, encryption, compression, networking, etc.

- Can be called from jupyter using "!"
- Save output to python variables
- Use $ to pass Python variable as shell commands arguments

- Open terminal/command prompt (from jupyter and using the OS)

Example of OS shell commands

- Note: these tools need to be installed manually on Windows (http://gnuwin32.sourceforge.net/packages/coreutils.htm). This was part of the installation instructions at the beginning of the course.

In [2]:
```
# unix command for listing files in the current working directory
!ls
```

```
[present lecture 4].bat
[present lecture 5].bat
data
figures
folder1
Lecture04 (class ready).ipynb
Lecture04 (class ready).slides.html
Lecture04.ipynb
Lecture05 (class ready).ipynb
Lecture05 (class ready).slides.html
Lecture05.ipynb
output.csv
output.txt
```

In [3]:
```
# we can assign the output of the command to Python variables
folder_content = !ls
print(folder_content) # for all pratical purposes this is a list
```

```
['[present lecture 4].bat', '[present lecture 5].bat', 'data', 'figures', 'fol
der1', 'Lecture04 (class ready).ipynb', 'Lecture04 (class ready).slides.html',
'Lecture04.ipynb', 'Lecture05 (class ready).ipynb', 'Lecture05 (class ready).s
lides.html', 'Lecture05.ipynb', 'output.csv', 'output.txt']
```

In [4]:
```
folder_content[:3]
```

Out[4]:
```
['[present lecture 4].bat', '[present lecture 5].bat', 'data']
```

In [5]:
```
for f in folder_content:
    print(f)
```

```
[present lecture 4].bat
[present lecture 5].bat
data
figures
folder1
Lecture04 (class ready).ipynb
Lecture04 (class ready).slides.html
Lecture04.ipynb
Lecture05 (class ready).ipynb
Lecture05 (class ready).slides.html
Lecture05.ipynb
output.csv
output.txt
```

Display files

In [6]:
```
# show the files and folders inside the folder named "folder1"
!ls folder1
```

```
book1.txt
book1_copy.txt
book2.txt
book3.txt
create_list.py
create_list_x.py
create_long_list.py
forest.py
novel.txt
```

In [7]:
```
# show the content of the file create_list.py
!cat folder1/create_list.py
```

```
# generate a list
result = []
for i in range(10):
        print(i)
        result.append(i)
```

Display files

In [8]:
```
# print the first 4 lines in the file novel.txt
!head -4 folder1/novel.txt
```

```
ï»¿The Project Gutenberg EBook of A Christmas Carol, by Charles Dickens

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
```

In [9]:
```
# unix command for counting lines, words and charachters in files
!wc folder1/novel.txt
```

```
  3853  32326 186809 folder1/novel.txt
```

Extract lines from a text file

In [10]:
```
# returns lines from text file that match a given pattern (e.g., contain the
!grep Christmas folder1/novel.txt
```

```
ï»¿The Project Gutenberg EBook of A Christmas Carol, by Charles Dickens
Title: A Christmas Carol
happy feeling toward Christmas, though the privations and hardships of
Dickens gave his first formal expression to his Christmas thoughts in
his series of small books, the first of which was the famous "Christmas
known to-day as the "Christmas Books." Of them all the "Carol" is the
stories. Whoever sees but a clever ghost story in the "Christmas Carol"
_"A Merry Christmas, Uncle! God save you!" cried a cheerful voice._     14
dog-days; and didn't thaw it one degree at Christmas.
Once upon a time--of all the good days in the year, on Christmas
"A merry Christmas, uncle! God save you!" cried a cheerful voice. It was
"Christmas a humbug, uncle!" said Scrooge's nephew. "You don't mean
"I do," said Scrooge. "Merry Christmas! What right have you to be merry?
[Illustration: _"A Merry Christmas, uncle! God save you!" cried a
of fools as this? Merry Christmas! Out upon merry Christmas! What's
Christmas-time to you but a time for paying bills without money; a time
Scrooge indignantly, "every idiot who goes about with 'Merry Christmas'
"Nephew!" returned the uncle sternly, "keep Christmas in your own way,
have not profited, I dare say," returned the nephew; "Christmas among
the rest. But I am sure I have always thought of Christmas-time, when it
your Christmas by losing your situation! You're quite a powerful
one thing in the world more ridiculous than a merry Christmas. "Good
in homage to Christmas, and I'll keep my Christmas humour to the last.
So A Merry Christmas, uncle!"
about a merry Christmas. I'll retire to Bedlam."
gentlemen, that is my answer. I don't make merry myself at Christmas,
fifty cooks and butlers to keep Christmas as a Lord Mayor's household
stooped down at Scrooge's keyhole to regale him with a Christmas carol;
boys, twenty times, in honour of its being Christmas-eve, and then ran
seven Christmas-eves ago. You have laboured on it since. It is a
"I am the Ghost of Christmas Past."
Christmas, as they parted at cross-roads and by-ways for their several
homes? What was merry Christmas to Scrooge? Out upon merry Christmas!
```

honest Ali Baba! Yes, yes, I know. One Christmas-time when yonder
"Nothing," said Scrooge. "Nothing. There was a boy singing a Christmas
"Let us see another Christmas!"
come back here; but first we're to be together all the Christmas long,
by the dressing of the shops, that here, too, it was Christmas-time
"Yo ho, my boys!" said Fezziwig. "No more work to-night. Christmas-eve,
Dick. Christmas, Ebenezer! Let's have the shutters up," cried old
or her a Merry Christmas. When everybody had retired but the two
shone out of the Ghost of Christmas Past.
Christmas toys and presents. Then the shouting and the struggling, and
"I am the Ghost of Christmas Present," said the Spirit. "Look upon me!"
The Ghost of Christmas Present rose.
hour of night, and they stood in the city streets on Christmas morning,
was good to eat and in its Christmas dress; but the customers were all
and for Christmas daws to peck at if they chose.
quarrel upon Christmas-day. And so it was! God love it, so it was!
Christmas Present blessed his four-roomed house!
your brother, Tiny Tim? And Martha warn't as late last Christmas-day by
rampant. "Not coming upon Christmas-day!"
remember upon Christmas-day who made lame beggars walk and blind men
ignited brandy, and bedight with Christmas holly stuck into the top.
"A merry Christmas to us all, my dears. God bless us!"
"My dear," said Bob, "the children! Christmas-day."
"It should be Christmas-day, I am sure," said she, "on which one drinks
"My dear!" was Bob's mild answer. "Christmas-day."
"not for his. Long life to him! A merry Christmas and a happy New Year!
he had any company but Christmas.
upon the barren waste, was singing them a Christmas song; it had been a
table at which they sat, they wished each other Merry Christmas in their
several stations; but every man among them hummed a Christmas tune, or
had a Christmas thought, or spoke below his breath to his companion of
some bygone Christmas-day, with homeward hopes belonging to it. And
"He said that Christmas was a humbug, as I live!" cried Scrooge's
I pity him. He may rail at Christmas till he dies, but he can't help
of Christmas Past. When this strain of music sounded, all the things
better than at Christmas, when its mighty Founder was a child himself.
Scrooge's nephew; and that the Ghost of Christmas Present knew it. The
"A merry Christmas and a happy New Year to the old man, whatever he is!"
of this, because the Christmas holidays appeared to be condensed into
"I am in the presence of the Ghost of Christmas Yet to Come?" said
"Seasonable for Christmas-time. You are not a skater, I suppose?"
lighted cheerfully, and hung with Christmas. There was a chair set close
The Ghost of Christmas Yet To Come conveyed him, as before--though at a
"I will honour Christmas in my heart, and try to keep it all the year. I
within me. Oh, Jacob Marley! Heaven and the Christmas Time be praised
a school-boy. I am as giddy as a drunken man. A merry Christmas to
Christmas Present sat! There's the window where I saw the wandering
"It's Christmas Day!" said Scrooge to himself. "I haven't missed it. The
Whoop! How are you? Merry Christmas!"
with the Ghost of Christmas Present; and, walking with his hands behind
fellows said, "Good morning, sir! A merry Christmas to you!" And Scrooge
yesterday. It was very kind of you. A merry Christmas to you, sir!"
"A merry Christmas, Bob!" said Scrooge with an earnestness that could
not be mistaken, as he clapped him on the back. "A merrier Christmas,
discuss your affairs this very afternoon, over a Christmas bowl of
him that he knew how to keep Christmas well, if any man alive possessed
End of the Project Gutenberg EBook of A Christmas Carol, by Charles Dickens

Extract lines from a text file

In [11]:
```
# other options
!grep --help
```

```
Usage: grep [OPTION]... PATTERN [FILE]...
Search for PATTERN in each FILE or standard input.
PATTERN is, by default, a basic regular expression (BRE).
Example: grep -i 'hello world' menu.h main.c

Regexp selection and interpretation:
  -E, --extended-regexp     PATTERN is an extended regular expression (ERE)
  -F, --fixed-strings       PATTERN is a set of newline-separated fixed string
s
  -G, --basic-regexp        PATTERN is a basic regular expression (BRE)
  -P, --perl-regexp         PATTERN is a Perl regular expression
  -e, --regexp=PATTERN      use PATTERN for matching
  -f, --file=FILE           obtain PATTERN from FILE
  -i, --ignore-case         ignore case distinctions
  -w, --word-regexp         force PATTERN to match only whole words
  -x, --line-regexp         force PATTERN to match only whole lines
  -z, --null-data           a data line ends in 0 byte, not newline

Miscellaneous:
  -s, --no-messages         suppress error messages
  -v, --invert-match        select non-matching lines
  -V, --version             print version information and exit
      --help                display this help and exit
      --mmap                use memory-mapped input if possible

Output control:
  -m, --max-count=NUM       stop after NUM matches
  -b, --byte-offset         print the byte offset with output lines
  -n, --line-number         print line number with output lines
      --line-buffered       flush output on every line
  -H, --with-filename       print the filename for each match
  -h, --no-filename         suppress the prefixing filename on output
      --label=LABEL         print LABEL as filename for standard input
  -o, --only-matching       show only the part of a line matching PATTERN
  -q, --quiet, --silent     suppress all normal output
      --binary-files=TYPE   assume that binary files are TYPE;
                            TYPE is `binary', `text', or `without-match'
  -a, --text                equivalent to --binary-files=text
  -I                        equivalent to --binary-files=without-match
  -d, --directories=ACTION  how to handle directories;
                            ACTION is `read', `recurse', or `skip'
  -D, --devices=ACTION      how to handle devices, FIFOs and sockets;
                            ACTION is `read' or `skip'
  -R, -r, --recursive       equivalent to --directories=recurse
      --include=FILE_PATTERN  search only files that match FILE_PATTERN
      --exclude=FILE_PATTERN  skip files and directories matching FILE_PATTERN
      --exclude-from=FILE   skip files matching any file pattern from FILE
      --exclude-dir=PATTERN directories that match PATTERN will be skipped.
  -L, --files-without-match print only names of FILEs containing no match
  -l, --files-with-matches  print only names of FILEs containing matches
```

```
  -c, --count               print only a count of matching lines per FILE
  -T, --initial-tab         make tabs line up (if needed)
  -Z, --null                print 0 byte after FILE name

Context control:
  -B, --before-context=NUM  print NUM lines of leading context
  -A, --after-context=NUM   print NUM lines of trailing context
  -C, --context=NUM         print NUM lines of output context
  -NUM                      same as --context=NUM
      --color[=WHEN],
      --colour[=WHEN]       use markers to highlight the matching strings;
                            WHEN is `always', `never', or `auto'
  -U, --binary              do not strip CR characters at EOL (MSDOS)
  -u, --unix-byte-offsets   report offsets as if CRs were not there (MSDOS)

`egrep' means `grep -E'.  `fgrep' means `grep -F'.
Direct invocation as either `egrep' or `fgrep' is deprecated.
With no FILE, or when FILE is -, read standard input.  If less than two FILEs
are given, assume -h.  Exit status is 0 if any line was selected, 1 otherwise;
if any error occurs and -q was not given, the exit status is 2.

Report bugs to: bug-grep@gnu.org
GNU Grep home page: <http://www.gnu.org/software/grep/>
General help using GNU software: <http://www.gnu.org/gethelp/>
```

Copy files

In [12]:
```
# list files
!ls folder1
```

```
book1.txt
book1_copy.txt
book2.txt
book3.txt
create_list.py
create_list_x.py
create_long_list.py
forest.py
novel.txt
```

In [13]:
```
# create a copy of the file book1.txt
!cp folder1/book1.txt folder1/book1_copy.txt
!ls folder1
```

```
book1.txt
book1_copy.txt
book2.txt
book3.txt
create_list.py
create_list_x.py
create_long_list.py
forest.py
novel.txt
```

Delete files

In [14]:
```
# delete the file book1_copy.txt
!rm folder1/book1_copy.txt
!ls folder1
```

```
book1.txt
book2.txt
book3.txt
create_list.py
create_list_x.py
create_long_list.py
forest.py
novel.txt
```

In [15]:
```
# copy file (using the $ sign before python variables)
x='folder1/book1.txt'
y='folder1/book1_copy.txt'
!cp $x $y
!ls folder1
```

```
book1.txt
book1_copy.txt
book2.txt
book3.txt
create_list.py
create_list_x.py
create_long_list.py
forest.py
novel.txt
```

In [16]:
```
!rm book1_copy.txt
```

```
rm: cannot remove `book1_copy.txt': No such file or directory
```

Exercise 1

Print the first row and the total number of lines and words of each text file in the code directory.

In [17]:
```
# write solution here
```

'Magic' commands

- Facilitate common programming tasks
- Unlike OS shell command, magic commands are unique to jupyter
- Examples
  - Start with ''%'' (for lines) and "%%" (for cells)
  - %run - executes python files
  - %time - time code execution of a single command
  - %who - print interactive variables
  - %debug - enters debug mode just before exception raised (exit debug mode by typing "exit")

Example of magic commands - run

In [19]:
```
!cat folder1/create_list.py
```

```
# generate a list
result = []
for i in range(10):
        print(i)
        result.append(i)
```

In [20]:
```
# run the file "create_list.py" on an empty Python environment, print the out
%run folder1/create_list.py
```

```
0
1
2
3
4
5
6
7
8
9
```

In [21]:
```
print(result) # notice that results was defined in the create_list.py but not
```

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

Example of magic commands - time

- Example: comparing the running time of two functions for matrix multiplication (they take two matrixes and return another matrix, the details are not important)

In [22]:
```
# Here is an implementation of matrix multiplication
import numpy as np
def multiply_manually(A,B):
    if ((len(A.shape)!=2) or (len(B.shape)!=2) or (A.shape[1]!=B.shape[0])):
        raise Exception('Matrix multiplication is not defined for the given m
    result_py = []
    for i in range(A.shape[0]):
        result_py.append([])
        for j in range(B.shape[1]):
            temp = 0
            for t in range(A.shape[1]):
                temp+= A[i,t]*B[t,j]
            result_py[-1].append(temp)

    return(np.array(result_py))
```

Example of magic commands - time (cont.)

In [23]:
```
d = 100 # matrix dimension

# generate two random matrixes A and B
A = np.random.randint(1,10,size=(d,d))
B = np.random.randint(1,10,size=(d,d))

# Multiply matrices
m1 = np.dot(A,B)            # numpy implementation
m2 = multiply_manually(A,B) # our own implementation
print(np.all(m1==m2))       # check if the result is the same
```

```
True
```

In [24]:
```
# measure the running time of two implementations of the same function.
# The first line uses the numpy package, which is implemented in c and runs s
%time m1 = np.dot(A,B)
%time m2 = multiply_manually(A,B)
```

```
Wall time: 996 µs
Wall time: 696 ms
```

Resolving bugs in the code

In [25]:
```
# here is some code that generates errors
for i in range(-4,4,1):
    print("i=",i, "1/i=",1/i) # expect an error
```

```
i= -4 1/i= -0.25
i= -3 1/i= -0.3333333333333333
i= -2 1/i= -0.5
i= -1 1/i= -1.0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-25-b029b0b5818c> in <module>
      1 # here is some code that generates errors
      2 for i in range(-4,4,1):
----> 3     print("i=",i, "1/i=",1/i) # expect an error

ZeroDivisionError: division by zero
```

Resolving bugs in the code

- The debugger is a tool for inspecting the code and finding errors
- It allows us to enter the exact moment of time just before the program crashed
- We can then inspect the values of variables, execute commands, and even go up the traceback (the functions that called the current function that resulted in an error)

- Another example of erroneous code
  - A simple synthetic example of a function calling another function whose code results in a crash

In [26]:
```
def f():
    x = 1
    for i in range(-4,4,1):
        print("i=",i, "1/i=",1/i) # expect an error
    return(0)

def g():
    x = 2
    return(f())
```

In [27]:
```
g() # call the code
```

```
i= -4 1/i= -0.25
i= -3 1/i= -0.3333333333333333
i= -2 1/i= -0.5
i= -1 1/i= -1.0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-27-c859c9894fc5> in <module>
----> 1 g() # call the code

<ipython-input-26-642e8700513f> in g()
      7 def g():
      8     x = 2
----> 9     return(f())

<ipython-input-26-642e8700513f> in f()
      2     x = 1
      3     for i in range(-4,4,1):
----> 4         print("i=",i, "1/i=",1/i) # expect an error
      5     return(0)
      6

ZeroDivisionError: division by zero
```

Example of magic command - %debug

- We type **%debug** to enter debug mode, and it is important to type **exit** to leave it
- Typing u will take us up the traceback

In [28]:
```
%debug
```

```
> <ipython-input-26-642e8700513f>(4)f()
      2     x = 1
      3     for i in range(-4,4,1):
----> 4         print("i=",i, "1/i=",1/i) # expect an error
      5     return(0)
      6

ipdb> exit
```

Python vs. IPython

- If you open the terminal/command prompt and type **python** and **ipython**, you will respectively see these two screens
- This command line interface (CLI) is another way of working with Python
- In jupyter, we use IPython, which is basically Python + magic commands + OS shell commands
- These added functionalities are very convenient for data analysis

jupyter

Key points

- IPython is a generalization of Python
- jupyter notebooks and python shells are instances of IPython
  - each notebook has a Python shell running in the background
  - when we run a notebook, we effectively copy-paste commands (to the shell) and output (to the notebook)
  - when we simultaneously open multiple notebooks there are separate shells
    - e.g., a variable defined in one notebook (x=3) will not be recognized in other notebooks
- QT Console

QTConsole

- An **ipython** shell
- Images can be embedded in the shell
- Not commonly used in practice

Spyder

- An integrated development environment
- The Python equivalent of R Studio

When to use each environment?

- For data analysis, jupyter notebooks are typically a good choice as they are convenient for
  - exploring data
  - generating and showing plots
  - sharing results
- For heavier code, use spyder (or alternative IDEs such as pydev)
- For quick calculations or when graphical interface is unavailable (e.g., when connecting remotely), use IPython shell

Further Reading

- Shell commands: grep, sort, awk, uniq, cut, ... (tutorials: 1 2 3 IPython documentation Jupyter documentation)
- Debugging and Profiling with IPython: Chapter 3 in Python for Data Analysis and the IPython tutorial
- Parallel computing, Remote Server
- IPython IDEs: Spyder, PyDev, Enthought, Python(x,y), WinPython, Pyzo
- Version control and sharing notebooks
  - GitHub gist
  - GitHub and Jupyter nbviewer
  - IPython notebooks Gallery with many tutorials
- https://colab.research.google.com

Summary

- Discussed advanced features of jupyter
  - OS Shell
  - Magic commands
  - Alternative ways of working with Python

In [ ]: