Welcome to our first 2019 coding skills course!!
This etherpad is for you to take notes. The notes will be stored to the event page after/during the workshop!
Use the chat window for chatting offline.

All links and notes will be shared through the event page:
    https://indico.mpi-cbg.de/event/135/timetable/#20190114

Please download and expand this zipfile:
    http://swcarpentry.github.io/shell-novice/data/data-shell.zip
Current course material: https://swcarpentry.github.io/shell-novice/

```
    /Users/wiegand
    /Users/love
    /home/danils
    cd
    /c/Users/blee
/Users/poser
/Useqq
rs/hpetzoldls -F

/Users/victoriayan
/Users/fberndt
/Users/dsaha/Desktop/data-shell
/Users/Johannes/Desktop/data-shell
/home/steinbac
/c/Users/vinograd/Documents/data-shell/data-shell
/c/Users/kellerp
/Users/dsaha/Desktop/data-shell
/home/rhaase/
/mnt/c/User/Cedric/Documents/GitHub/MPI-CPG/data_analysis/data-shell/data
/Users/guhr
/Users/janosch
```

```
*      __...--~~~~~~-._   _.-~~~~~~--...__
*    //              `V'              \\
*   //                |                \\
*  //__...--~~~~~~~-._ | _.-~~~~~~~--...__\\
* //__.....----~~~~._\ | /_.~~~~----.....__\\
*===================\\|//====================
*                dwb `---`
```

lscd - change directory

Unix/Linux commands
http://cheatsheetworld.com/programming/unix-linux-cheat-sheet/

clear
CTRL-L

TAB - tab completion,
arrow up/down = recall last commands

mkdir - make directory

https://wiki.mpi-cbg.de/compdoc/Filenames

nana
yes = 0 bytes
Stephan Janosch: yes
blank text file, 0 bytes,
empty file, 0 bytes
yes 0 bytes

Use touch to check if you are allowed to write to a folder

rm - remove

sudo rm -rf / <<-- DON't do this. It will delete your whole file system. Also stuff of your colleagues if the filerserver is mounted. ?? Actually, who wrote this here?

Day1, morning feedback:
(green)
- I liked the clear explanations of the commands to use in terminal. Very useful!
- Alles is gute!
- interactive learning,+2
- very detailed explanations
- everything
- using terminal and command lines
- certain shortcuts
- never used touch or nano
- already knew except for some flags like 'ls -a'
- really good step-by-step intro
- cp & mv can commands can change filenames
- I learned some new bash commands
- moving through directories
- like the setup
(red)
- need more time
- moving/copying files, +1
- maybe a bit faster, +1
- mv/cp a bit too fast
- risk of overwriting files

*Day1 after lunch

ls th?sis*/*txt

*Pipes and Filters

wc -l [] count lines
sort [] sort file
head [] show the first lines of a file
tail [] show the last lines of a file; example: "tail -n 1 sorted-lengths.txt" gives you the last line

cat text1.txt > overwrittenFile.txt
cat text1.txt > overwrittenFile.txt

cat text1.txt >> appendingFile.txt
cat text1.txt >> appendingFile.txt

CTRL a - go to beginning of line
CTRL e - go to end of line

- ls NENE*[AB].txt

```
*       .
*      ,i \
*    ,' 8b \
*  ,;o  `8b \
* ;  Y8. d8  \
*-+._ 8: d8. i:
*    `:8 `8i `8
*      `._Y8  8:  ___
*         `'---Yjdp  "8m._
*              ,"' _,o9   `m._
*              | o8P"   _.8d8P`-._
*              :8'   _oodP"   ,dP'`-._
*               `: dd8P'   ,odP'  do8'`.
*                 `-'   ,o8P'  ,o8P' ,8P`.
*                   `._dP'   ddP'  ,8P' ,..
*                      "`._ PP'  ,8P' _d8'L..___
*                          `"-._88'  .PP,'7 ,8.`-.._
*                               ``"'--"'  | d8' :8i `i.
*                                         l d8  d8  dP/
*                                          \`' J8' `P'
*                                           \ ,8F  87
*                                            `.88  ,'
*                                             `.,-' mh
```

OPEN MIULTIPLE FILES WITH NANO
open nano with: nano -F <filename>
CTRL+R opens a second, third, .. file
ALT+, and ALT+. allows you to switch between files

if you want to remember which cool things you did recently with the grep command run history | grep grep

Day1, afternoon feedback:
(green)
- covered a good spread of functions and gave each the appropriate amount of attention
- good overview, nice sessions today
- how to organize files with wildcards, pipes and loops
- peter's way of teaching
- questions with 4 answers
- so many tools, I learned today. Many of my crapy scripts are oboslete now because I can do everything simply from the command line now
- I liked the examples, +1
- I liked it all, +1
- got a good idea what can be done with tools, pipes, filters and loops
- very in depth

(red)
- a bit fast, but I will review your detailed notes
- too fast for me, need to spend more time on it individually
- the afternoon is quite packed
- not sure what is the most elegant way with real filesets
- grep/find could have been introduced earlier
- too slow to cover all material

*Day 2

*String manipulation with Bash

https://www.learnshell.org/de/Basic_String_Operations
https://gist.github.com/magnetikonline/90d6fe30fc247ef110a1

looking at the PATH and location of excecutable programs
```
echo $PATH
```
to get a clean list of PATH use
```
echo $PATH | tr ":" "\n"
```
one possibility of adding scripts to your path to make scripts excecutable from any directory (Use an absolute path, not the relative)
```
PATH=/PATH/TO/SCRIPT/script.sh:$PATH
```

Book recommendation
The Pragmatic Programmer https://en.wikipedia.org/wiki/The_Pragmatic_Programmer
https://www.nceclusters.no/globalassets/filer/nce/diverse/the-pragmatic-programmer.pdf

please download this zip file

- https://swcarpentry.github.io/python-novice-inflammation/data/python-novice-inflammation-data.zip

for obtaining the training data

for a later part in the lesson, please also download the following zip file with code:
https://swcarpentry.github.io/python-novice-inflammation/code/python-novice-inflammation-code.zip

numercial python (numpy)
import in python
via
import numpy

print(data.shape)
gives you rows and numbers of a .csv object inside python
print(data[0:4, 0:10]) prints row 0-3 excluding the right hand number and row 0-9

https://software-carpentry.org/lessons/index.html
http://swcarpentry.github.io/python-novice-inflammation/

day2 feedback morning session
- green

- - recap from yesterday +1
- - python itself nicely explained +3
- - sample dataset nice
- - good expla. of functions
- - clear python3 explanation incl structure and packages
- - i realy like rows and cols expl.
- - good speed
- - looking forward of more command usage
- - very nice, getting more exited
- - got ideas about slicing data

- red

- - need more time for recapping things
- - inflammation data set is not self explaination
- - hard to know what to expect
- - maybe a bit faster
- - Peter should get a new laptop
- - looking for more advanced data analysis in afternoon
- - which we got started with a good python editor and workspaces/consoles: spyder or pycharm
- - where to find more packages (anaconda [navigator] or google)
- - other ways for printing arrays (format)
- - emphasise where to find course material for personal recap
- - row/col confusion
- - mention potential pitfalls / common issues
-
-

*import numpy
*import matplotlib.pyplot
*
*data = numpy.loadtxt(fname="inflammation-01.csv",delimiter=",")
*
*fig = matplotlib.pyplot.figure(figsize=(10.0,3.0))
*
*axes1= fig.add_subplot(1,3,1)

```
*axes2= fig.add_subplot(1,3,2)
*axes3= fig.add_subplot(1,3,3)
*
*axes1.set_ylabel("average")
*axes1.plot(numpy.mean(data,axis=0))
*
*axes2.set_ylabel("max")
*axes2.plot(numpy.max(data,axis=0))
*
*axes3.set_ylabel("min")
*axes3.plot(numpy.min(data,axis=0))
*
*fig.tight_layout()
*
*matplotlib.pyplot.show()
```

```
result = 1
for i in range(0, 3):
    result = result * 5
print(result)
```

```
x = 1
y = 5
for i in range(0,3):
    x = y*x
print(x)
```

```
num_sqr = 5
power = 3
sqr_accum = num_sqr
for i in range(power-1):
    sqr_accum = sqr_accum*num_sqr
print sqr_accum
```

```
text = "Newton"
result = "";
for i in range(0, len(text)):
    result = result + text[len(text) - i - 1]
print(result)
```

```
word = "Newton"
newword = ""
for i in word:
    newword = i + newword
print(newword)
```

```
word = "Newton"
new = ""
for i in range(len(word)):
    new = new + word[-(i+1)]
print(new)


import glob
import numpy
import matplotlib.pyplot

filenames = glob.glob("inflammation-*.csv")
filenames = filenames[:3]

for f in filenames:
    data = numpy.loadtxt(fname=f,delimiter=",")

    fig = matplotlib.pyplot.figure(figsize=(10.0,3.0))

    axes1= fig.add_subplot(1,3,1)
    axes2= fig.add_subplot(1,3,2)
    axes3= fig.add_subplot(1,3,3)

    axes1.set_ylabel("average")
    axes1.plot(numpy.mean(data,axis=0))

    axes2.set_ylabel("max")
    axes2.plot(numpy.max(data,axis=0))

    axes3.set_ylabel("min")
    axes3.plot(numpy.min(data,axis=0))

    fig.tight_layout()

    matplotlib.pyplot.show()

data = numpy.loadtxt(fname="inflammation-01.csv", delimiter=",")

max_inflammation_0 = numpy.max(data,axis=0)[0]
max_inflammation_20 = numpy.max(data,axis=0)[20]

if max_inflammation_0 == 0 and max_inflammation_20 == 20:
    print("Suspicious looking data!")
elif numpy.sum(numpy.min(data,axis=0)) == 0:
    print("minima add up to 0")
else:
    print("Seems Ok!")
```

*fully self contained functions example

```
import glob
import matplotlib.pyplot
import numpy

def analyze(filename):
    """
    function to open <filename> (as .csv file) and plot the mean/max/min across axis 0
    """
    data = numpy.loadtxt(fname=filename,delimiter=",")

    fig = matplotlib.pyplot.figure(figsize=(10.0,3.0))

    axes1= fig.add_subplot(1,3,1)
    axes2= fig.add_subplot(1,3,2)
    axes3= fig.add_subplot(1,3,3)

    axes1.set_ylabel("average")
    axes1.plot(numpy.mean(data,axis=0))

    axes2.set_ylabel("max")
    axes2.plot(numpy.max(data,axis=0))

    axes3.set_ylabel("min")
    axes3.plot(numpy.min(data,axis=0))

    fig.tight_layout()

    matplotlib.pyplot.show()

def detect_problems(filename):
    """
    function to open <filename> (as .csv file) and check if the maxima do NOT follow a linear function between 0 and 20.
    also to check whether the minima across axis 0 add up to 0, otherwise declare dataset as OK
    """
    data = numpy.loadtxt(fname=filename, delimiter=",")

    max_inflammation_0 = numpy.max(data,axis=0)[0]
    max_inflammation_20 = numpy.max(data,axis=0)[20]

    if max_inflammation_0 == 0 and max_inflammation_20 == 20:
        print("Suspicious looking data!")
    elif numpy.sum(numpy.min(data,axis=0)) == 0:
        print("minima add up to 0")
    else:
        print("Seems Ok!")

for f in filenames:
    analyze(f)
    detect_problems(f)

help(analyze)
```

feedback day2 afternoon

- (red)
- - need more time to recap
- - exercises maybe after a break (freshness factor)
- - exercises maybe doing together once, then on your own (len was not mentioned)
- - examples were a bit far off
- - can you tell us more about refactoring?
- - difficult to distinguish lists versus strings
- (green)
- - I learned cool tricks
- - starting to feel the application of python
- - good note to structure code
- - got an overview what can be done with python
- - really cool how to combine different programs
- - just need more practise with loops
-

```
def outer (variable):
    result = (variable[0][0]) + (variable[-1][0])
    return result
print outer (["Byungho", "Tasinoivai", "Lee"])
BL

numbers = [1.5,2.3,0.7,-0.001,4.4]
total = 0.0
for num in numbers:
    assert num > 0., "input "+str(num)+" is negative. Stopping loop."
    total = total + num

print("sum is",total)




x1, y1

def normalize_rectangle(coordinates):
    """ normalize rectangle described by 4-integer tuple <coordinates>, so that it is at the
origin
    and 1 unit long along its longest axis
    input parameter <coordinates> is expected to be of the form (x0, y0, x1, y1)
    """
    #let's check the pre-conditions
    assert len(coordinates) == 4, 'Rectangles must contain 4 coordinates'
    x0, y0, x1, y1 = coordinates

    assert x0 < x1, "Invalid x coordinates"
    assert y0 < y1, "Invalid y coordinates"

    dx = x1 - x0
    dy = y1 - y0
    if dx > dy:
        #rectangle is rather wide
        scaled = float(dx) / dy
        upper_x, upper_y = 1.0, scaled
    else:
        #rectangel is rather tall
        scaled = float(dx) / dy
        upper_x, upper_y = scaled, 1.0

    #let's check the post-conditions
    assert 0 < upper_x <= 1.0, "calculated upper x coordinate failed"
    assert 0 < upper_y <= 1.0, "calculated upper y coordinate failed"

    return (0,0,upper_x,upper_y)

#Test-driven development:
# (red) write a failing test
# (green) add code that makes the test succeed
# (refactor) restructure the code to your liking WITHOUT breaking the succeeding tests

def range_overlap(intervals):
    """ return common overlap among a set of (low, high] ranges
    <intervals> : list of tuples where each tuple has 2 entries (low, high)"""
    lowest = 0.
    highest = 1.0
    for (low, high) in intervals:
        lowest = max(lowest, low)
        highest = min(highest, high)

    return (lowest,highest)


assert range_overlap([ (0., 1.) ]) == (0., 1.)
assert range_overlap([ (2,3), (2,4) ]) == (2,3)
assert range_overlap([ (0,1), (0,2), (-1,1) ]) == (0,1)


#Test-driven development:
# (red) write a failing test
# (green) add code that makes the test succeed
# (refactor) restructure the code to your liking WITHOUT breaking the succeeding tests

def range_overlap(intervals):
    """ return common overlap among a set of (low, high] ranges
    <intervals> : list of tuples where each tuple has 2 entries (low, high)"""
    low_ends = []
    for (low, _) in intervals:
        low_ends.append(low)
    hi_ends = []
    for (_, hi) in intervals:
        hi_ends.append(hi)
    lowest = min(low_ends)
    highest = max(hi_ends)
    for (low, high) in intervals:
        lowest = max(lowest, low)
        highest = min(highest, high)

    return (lowest,highest)


assert range_overlap([ (0., 1.) ]) == (0., 1.)
assert range_overlap([ (2,3), (2,4) ]) == (2,3)
assert range_overlap([ (0,1), (0,2), (-1,1) ]) == (0,1)
```

```
>>> 3//0.1
29.0
>>> 3/0.1
30.0
```

Why??

----

feedback day3 morning

- (red)
- - examples take too long, wish we can cover more examples
- - still unclear about how to do TDD, the best way to write assertions? (just need practise)
- - more practical examples
- - pretty complicated for me (homework?)
- - too fast, too much typing for defensive programming (don't make people copy code, but explain step-by-step)
- (green)
- - very nice access to defensive programming
- - 1st time TDD!
- - very insightful TDD, noone ever has taught me this
- - it looks easy until I do it myself
- - very useful, mostly TDD
- - defensive programming, assert
- - liked the exercises, +1
- - liked finding errors
- - very clear explanation
- - nice explanation
- - improved practising
-
-
-

```
import sys
import numpy

def print_means(filename):
    data = numpy.loadtxt(filename, delimiter=',')
    for m in numpy.mean(data,axis=1):
        print(m)

#print(sys.version)

def main():
    script = sys.argv[0]
    action = sys.argv[1]

    if action != '--min' and action != '--mean':
        print("usage: python "+script+" <--min|--mean> [file ...]")
        sys.exit(1)
    filenames = sys.argv[2:]

    for fname in filenames:
        print(fname)
        data = numpy.loadtxt(fname, delimiter=',')
        values = None
        if action == '--min':
            values = numpy.min(data,axis=1)
        elif action == '--mean':
            values = numpy.mean(data,axis=1)

        for m in values:
            print(m)

if __name__ == '__main__':
    main()
```

https://www.gnu.org/software/diffutils/manual/diffutils.html < --- search for 'Myers'
http://dx.doi.org/10.1007/BF01840446

GIT reference:

http://swcarpentry.github.io/git-novice/

feedback day3 afternoon

- (red)
- - more explanations of the motivation or overview before jumping into examples
- - more practical examples
- - fatal push/pull, +1
- - confusion when running python from the command line
- - intro/motivation what a repo is
- - explain what advantages a repo has over manual versioning
- - more seperation between beginner & advanced level
-
- (green)

- - very clearly explained
- - good level of exercise and theory, +1
- - good interaction
- - nice fast comprehensive crash course on git
- - saved time and activating energy
- - finally understood so many command line things and great tricks, +1
- - very informative, +2
- - good sweets, +1
- - course could be longer
- - git is super useful, +4
- - great course, +2
- - please run a more advanced one
- - how to add collaborators (train collaboration with examples?)
- - nice to see the workflow