Modules, environments and other topics

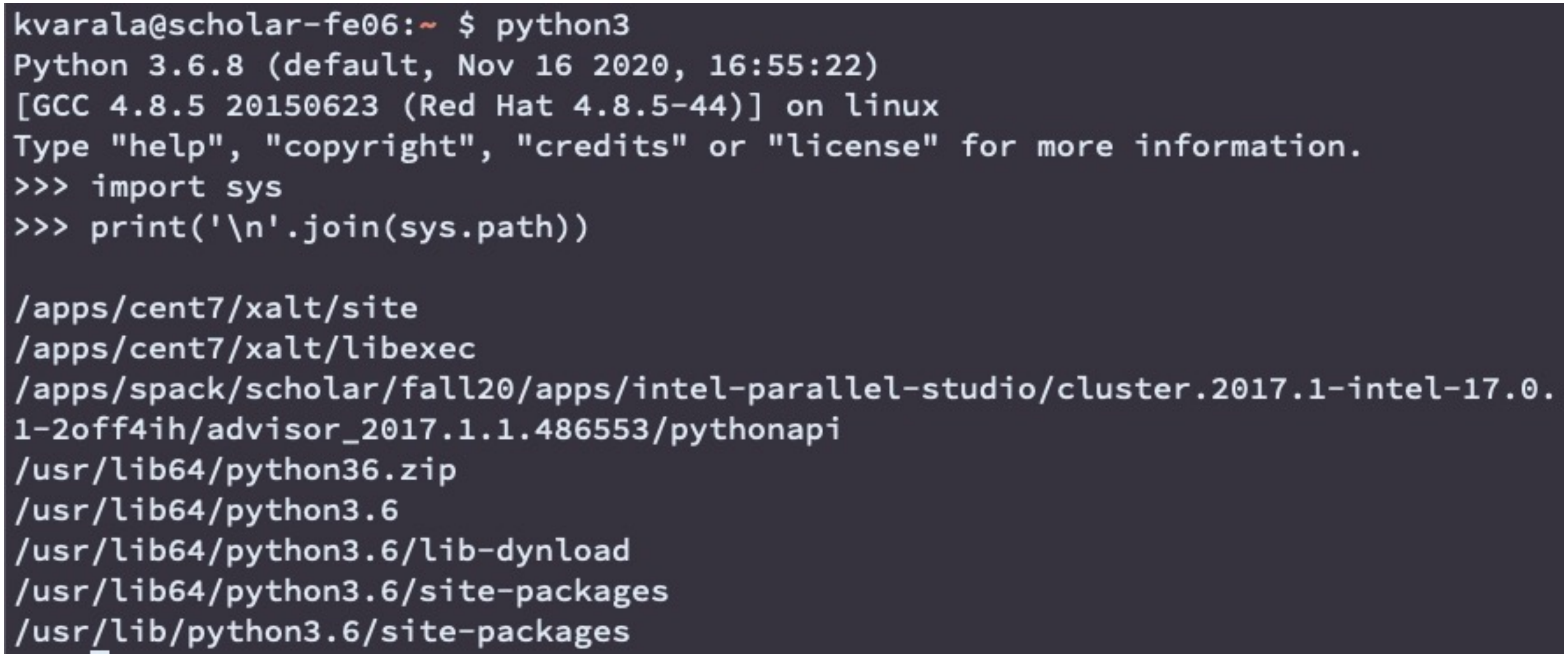
HORT 530

Lecture 14

Modules in Python

- Modules are other scripts that you can include in your script.
- A great way to include code that you or someone else wrote.
- Python has hundreds of standard modules. For example: 'os', 'sys', 'time' etc.
- Non-standard python modules have to be installed before you can use them.
- On servers, ask the system administrator to install them for you OR install them in your home folder.

The module search path



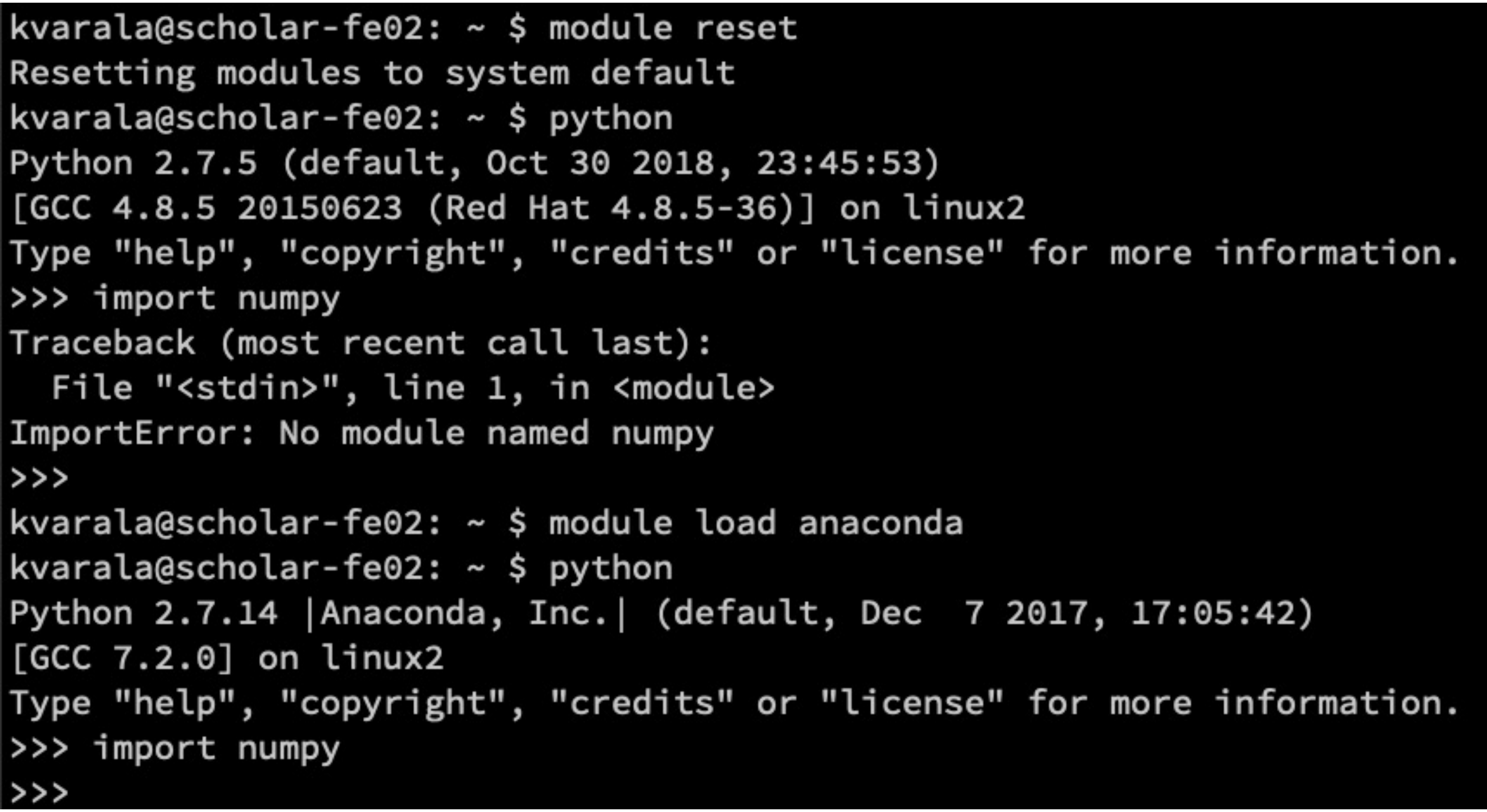
```
kvarala@scholar-fe06:~ $ python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> print('\n'.join(sys.path))

/apps/cent7/xalt/site
/apps/cent7/xalt/libexec
/apps/spack/scholar/fall20/apps/intel-parallel-studio/cluster.2017.1-intel-17.0.
1-2off4ih/advisor_2017.1.1.486553/pythonapi
/usr/lib64/python36.zip
/usr/lib64/python3.6
/usr/lib64/python3.6/lib-dynload
/usr/lib64/python3.6/site-packages
/usr/lib/python3.6/site-packages
```

- Python searches for modules in the locations specified by sys.path.
- sys.path tells you the locations AND order of them.
- You can install packages anywhere and add that location to the PYTHONPATH environment variable.

Module version clashes

- Since import does not support any version number, only one version of a module can exist at a time.
- You may need to use different versions of the same module with different scripts.
- Keeping track of all the version numbers for each of your scripts gets tedious.
- Python's solution is to use "environments" which bind a version of python to a set of modules using the PYTHONPATH and other environment variables.

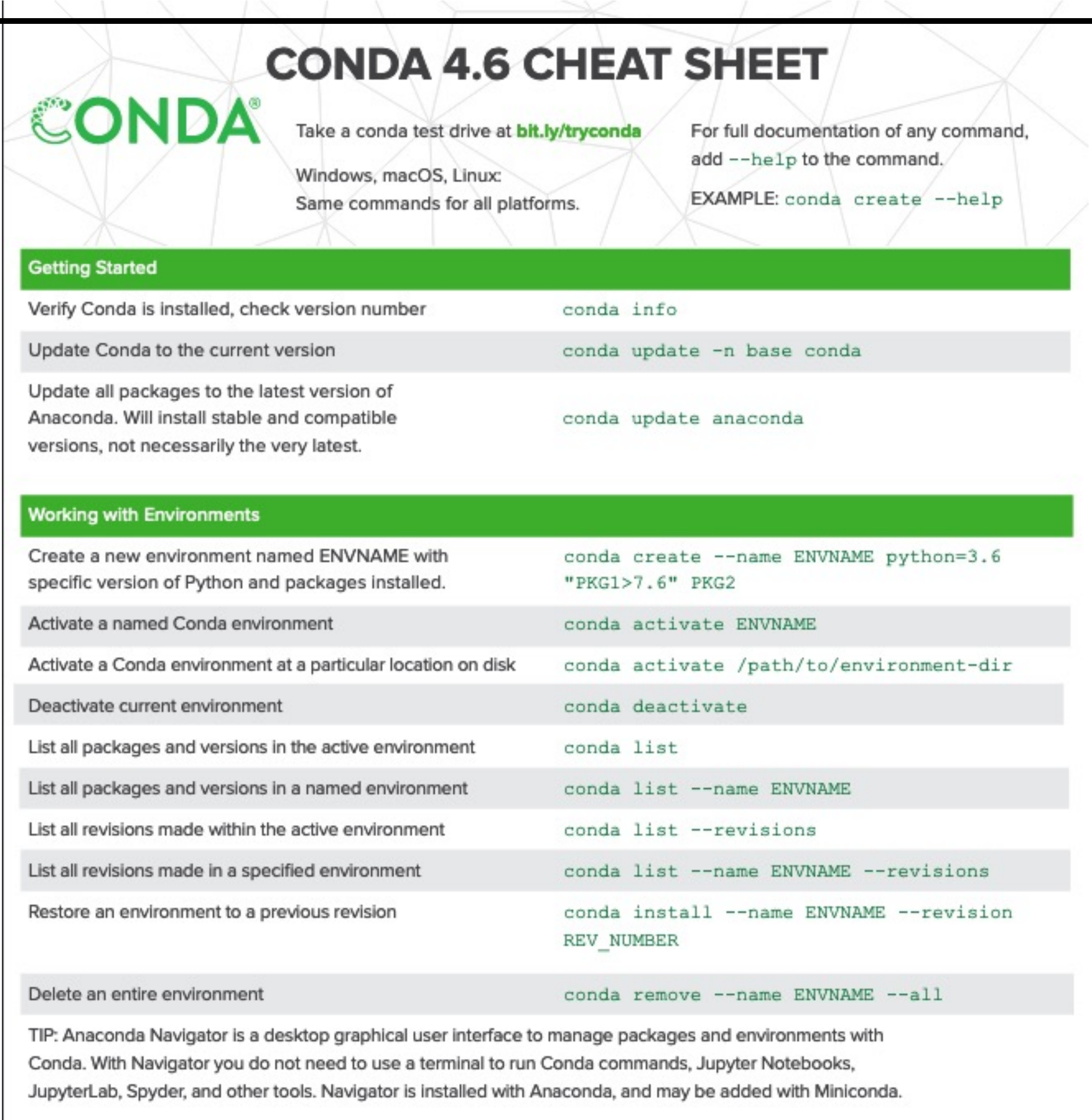
Python environment managers



```
kvarala@scholar-fe02: ~ $ module reset
Resetting modules to system default
kvarala@scholar-fe02: ~ $ python
Python 2.7.5 (default, Oct 30 2018, 23:45:53)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named numpy
>>>
kvarala@scholar-fe02: ~ $ module load anaconda
kvarala@scholar-fe02: ~ $ python
Python 2.7.14 |Anaconda, Inc.| (default, Dec  7 2017, 17:05:42)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

Anaconda

https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html



CONDA 4.6 CHEAT SHEET

CONDA®

Take a conda test drive at **bit.ly/tryconda**

Windows, macOS, Linux:
Same commands for all platforms.

For full documentation of any command, add `--help` to the command.

EXAMPLE: `conda create --help`

Getting Started	
Verify Conda is installed, check version number	`conda info`
Update Conda to the current version	`conda update -n base conda`
Update all packages to the latest version of Anaconda. Will install stable and compatible versions, not necessarily the very latest.	`conda update anaconda`

Working with Environments	
Create a new environment named ENVNAME with specific version of Python and packages installed.	`conda create --name ENVNAME python=3.6 "PKG1>7.6" PKG2`
Activate a named Conda environment	`conda activate ENVNAME`
Activate a Conda environment at a particular location on disk	`conda activate /path/to/environment-dir`
Deactivate current environment	`conda deactivate`
List all packages and versions in the active environment	`conda list`
List all packages and versions in a named environment	`conda list --name ENVNAME`
List all revisions made within the active environment	`conda list --revisions`
List all revisions made in a specified environment	`conda list --name ENVNAME --revisions`
Restore an environment to a previous revision	`conda install --name ENVNAME --revision REV_NUMBER`
Delete an entire environment	`conda remove --name ENVNAME --all`

TIP: Anaconda Navigator is a desktop graphical user interface to manage packages and environments with Conda. With Navigator you do not need to use a terminal to run Conda commands, Jupyter Notebooks, JupyterLab, Spyder, and other tools. Navigator is installed with Anaconda, and may be added with Miniconda.

Creating anaconda environments

- conda create --name py310 python=3.10
- conda activate py310
- conda install numpy scipy

Regular expressions in Python

- The regular expression module in Python is called 're'
- Match = re.search(pat, string)
- Match.group() contains results of the search.
- Search always finds the leftmost and largest match for the pattern.
- re.findall() allows finding all matches, not just the leftmost.
- re.sub(pat,replacement, string) allows replacing a pattern.

Special characters

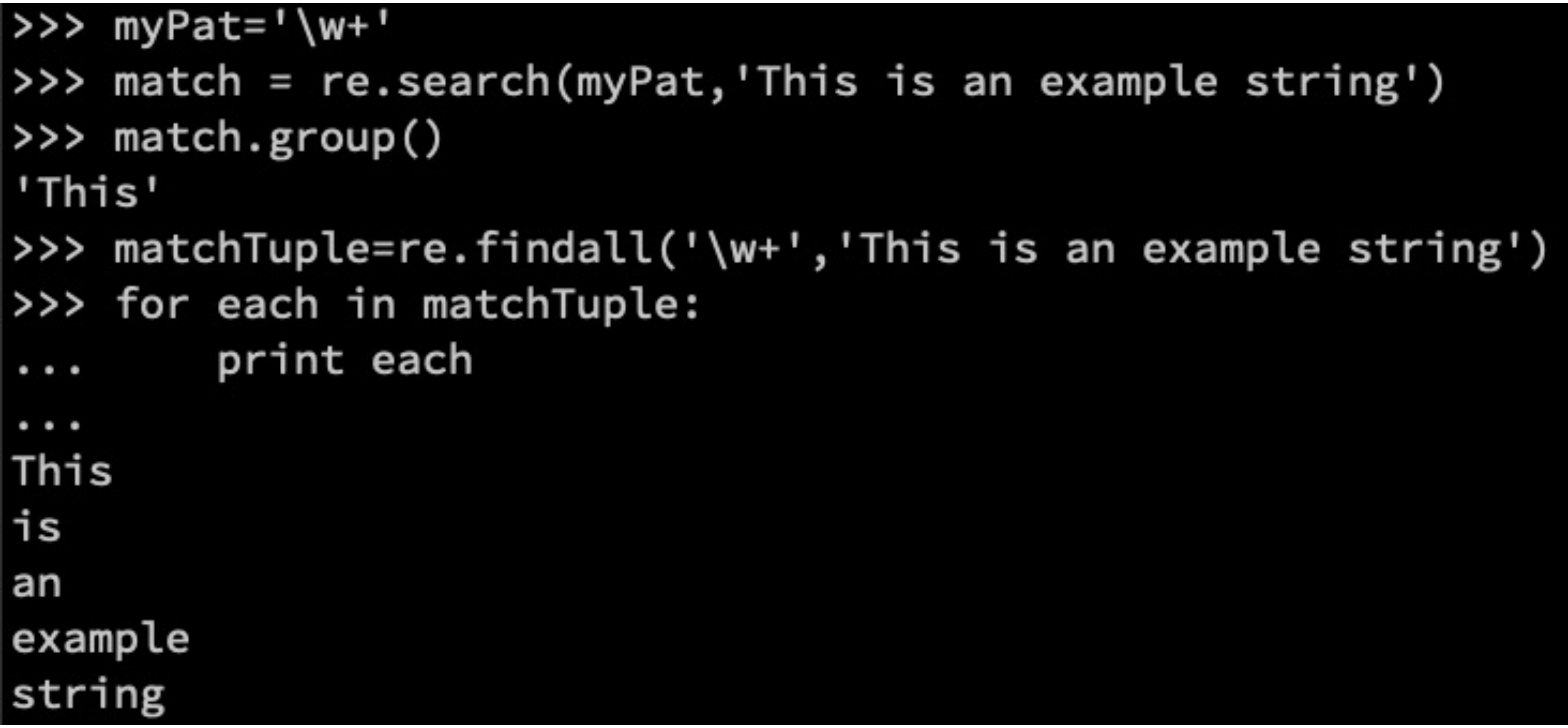
- \d digit, i.e., 0-9
- \D anything except a digit
- \w word (includes letter, digit, underscore)
- \W any character that is not included in word
- ^ Start of line
- $ End of line
- Examples: /\d\d$/

Character classes

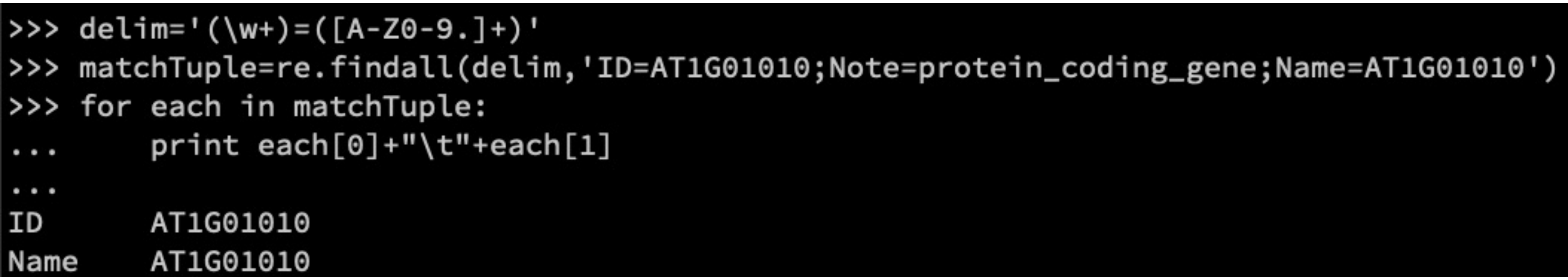
- [A-Z]   matches all upper-case letters
- [a-z]   matches all lower-case letters
- [0-9]   matches all digits
- [tnr][hd]   matches th or nd or rd
- Character class can be negated by using ^ as the first character in the class.
- [^0-9]   matches all characters that are not a number

Quantifiers

- * zero or more matches
- + one or more matches
- ? zero or one matches
- {2} exactly 2 matches
- {2,10} at least 2, maximum of 10 matches
- {,10} 0-10 matches



```
>>> myPat='\w+'
>>> match = re.search(myPat,'This is an example string')
>>> match.group()
'This'
>>> matchTuple=re.findall('\w+','This is an example string')
>>> for each in matchTuple:
...     print each
...
This
is
an
example
string
```

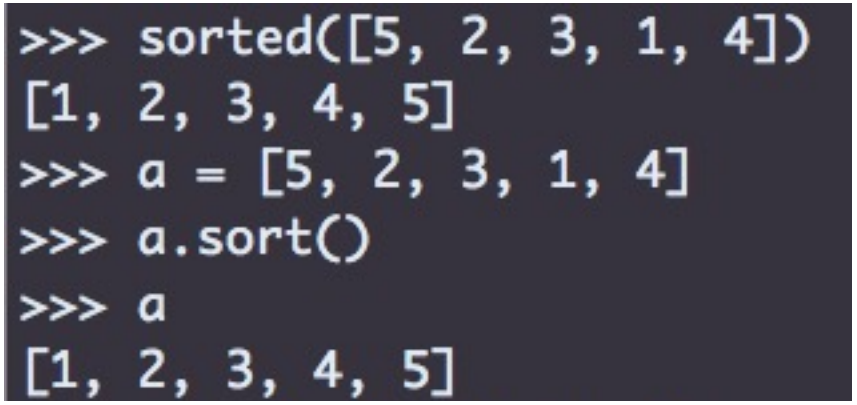


```
>>> delim='(\w+)=([A-Z0-9.]+)'
>>> matchTuple=re.findall(delim,'ID=AT1G01010;Note=protein_coding_gene;Name=AT1G01010')
>>> for each in matchTuple:
...     print each[0]+"\t"+each[1]
...
ID      AT1G01010
Name    AT1G01010
```

Sorting a list of values

- Sorting is the process of ordering items in an increasing (or decreasing) order based on their value.
- Lists in Python can be sorted in two ways:
  - list.sort() function
  - sorted() function
- sorted is a general function that will accept any iterable item, such as dictionaries, tuples etc.

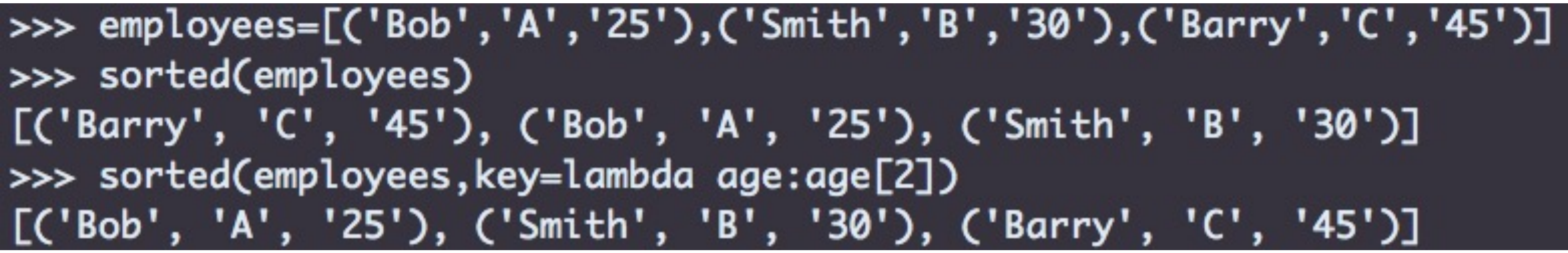
Sorting a list of values



```
>>> sorted([5, 2, 3, 1, 4])
[1, 2, 3, 4, 5]
>>> a = [5, 2, 3, 1, 4]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

- Remember to capture the output of sorted in a new object

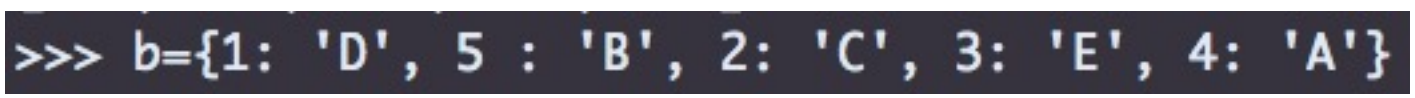
Sorting tuples



```
>>> employees=[('Bob','A','25'),('Smith','B','30'),('Barry','C','45')]
>>> sorted(employees)
[('Barry', 'C', '45'), ('Bob', 'A', '25'), ('Smith', 'B', '30')]
>>> sorted(employees,key=lambda age:age[2])
[('Bob', 'A', '25'), ('Smith', 'B', '30'), ('Barry', 'C', '45')]
```

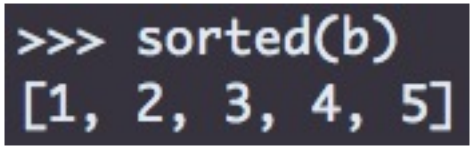
- Sorting tuples by default will sort them by the first element in the tuple. In this example, that is the name of the employee.
- The sorted() function can be used to sort based on a different element by using the ‘key’ argument.
- In this example, the lambda construct is used to generate an inline functions that simply returns the element at index 2.

Sorting Dictionaries



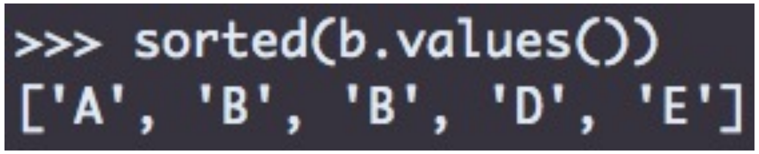
```
>>> b={1: 'D', 5 : 'B', 2: 'C', 3: 'E', 4: 'A'}
```

- Sort by keys :



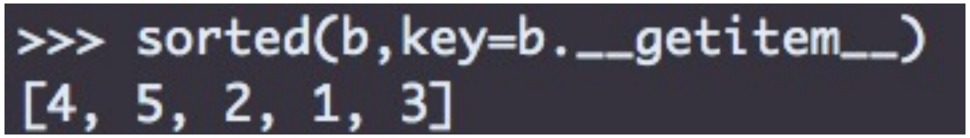
```
>>> sorted(b)
[1, 2, 3, 4, 5]
```

- Sort by values :



```
>>> sorted(b.values())
['A', 'B', 'B', 'D', 'E']
```

- Sort keys by values :



```
>>> sorted(b,key=b.__getitem__)
[4, 5, 2, 1, 3]
```

Using pandas

```
import pandas as pd
import numpy as np

expMat =pd.read_csv("GSE49418_series_matrix.txt",sep='\t',header=0,index_col=0)
newInd = (np.arange(len(expMat.columns)) // 3) + 1
expAvg =expMat.groupby(newInd, axis=1).sum()
expAvg.rename(columns={1:"WT-CK",2:"MT-CK",3:"WT-Dry",4:"MT-Dry"},inplace=True)
expAvg = expAvg.div(3)
filtMat =expMat[expAvg["WT-CK"]>5]
filtMat2 =expMat[(expAvg["WT-CK"]>5) & (expAvg["MT-CK"]>5)]
print(expMat.shape)
print(filtMat.shape)
print(filtMat2.shape)
print(expMat.corr())
print(filtMat.corr())
print(filtMat2.corr())
```

Final project specifications

- The code for this project MUST be written in the Python language and has to run on the Purdue Scholar cluster.
- The project submission HAS to include:
  - Input data set, student's code (on Scholar) and
  - All instructions required to execute the code (Readme.txt on Scholar OR in an email).
- Project submission deadline is 12 PM on the May 7th, 2022.
- Final project is worth 30% of your grade.