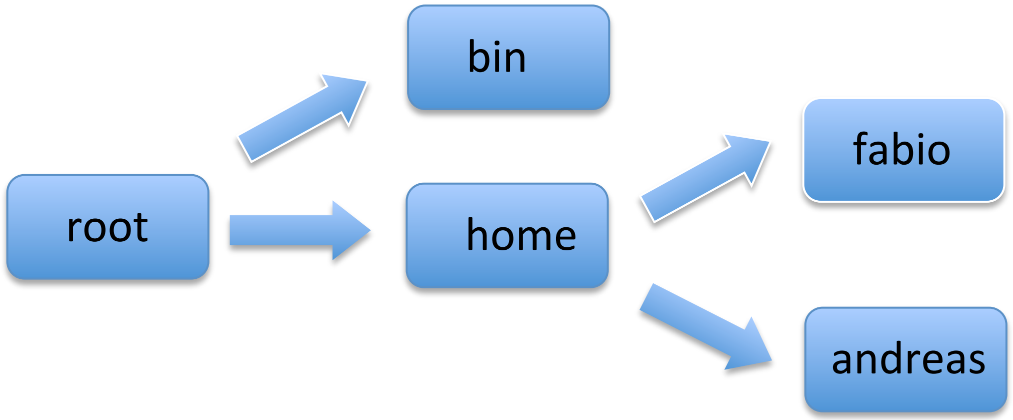
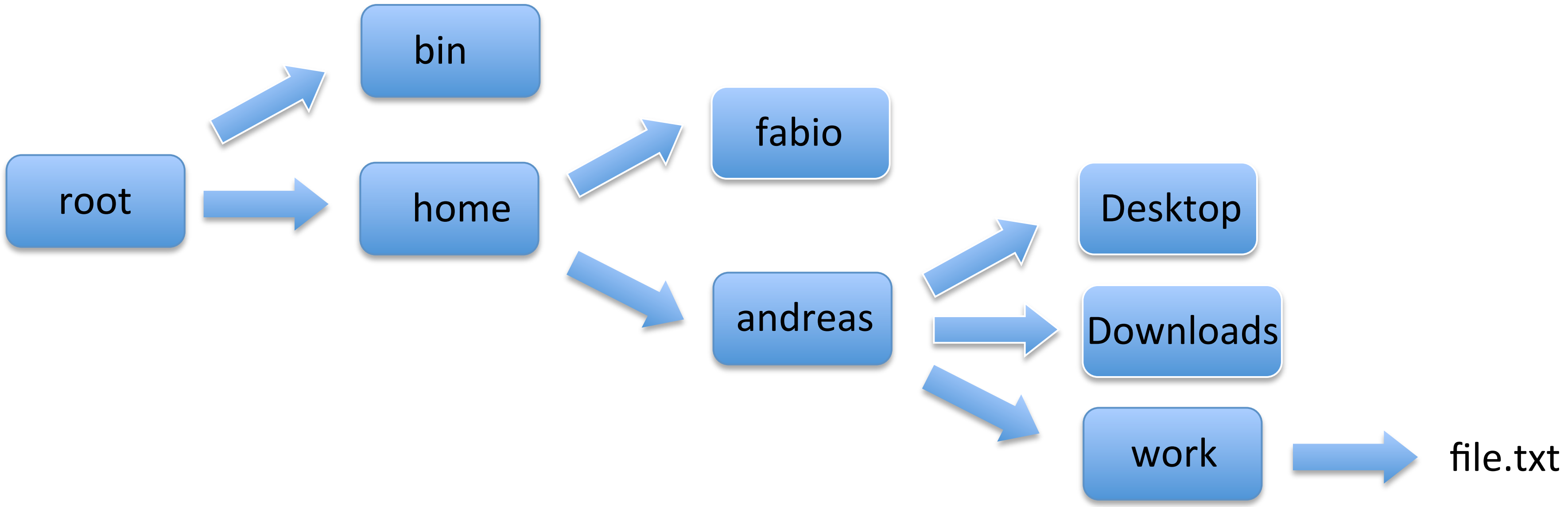
Python course 2013
Commandline

Andreas Weller

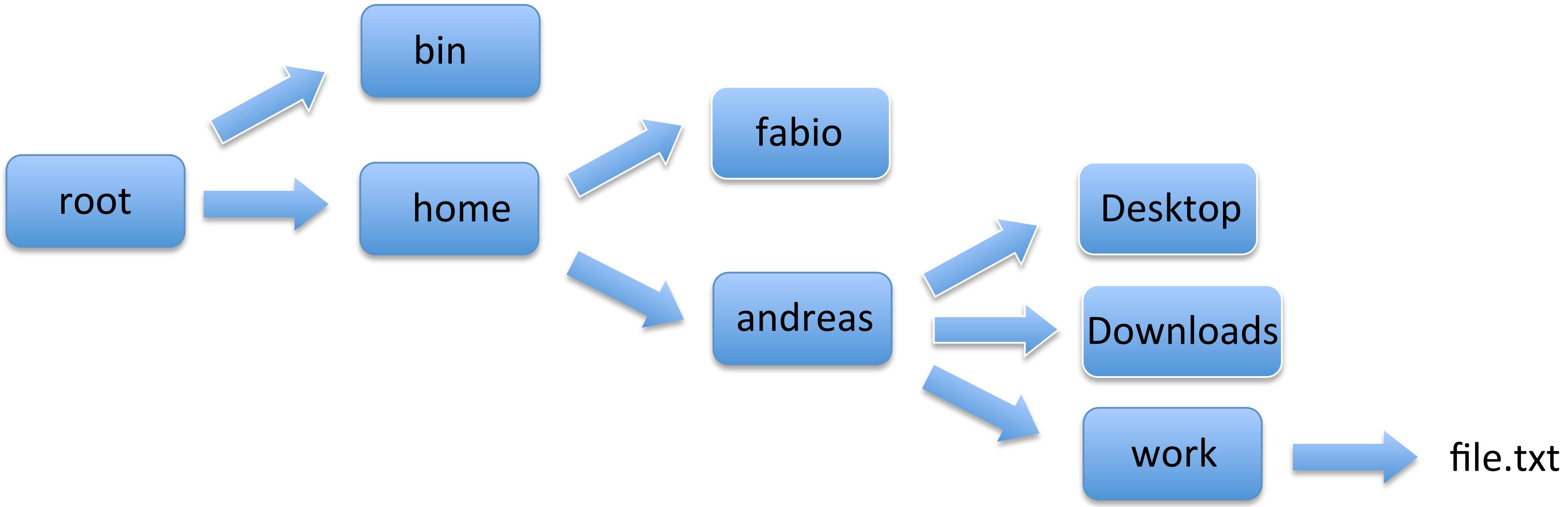
File system



File system


root
bin
home
fabio
andreas
Desktop
Downloads
work
file.txt

File system

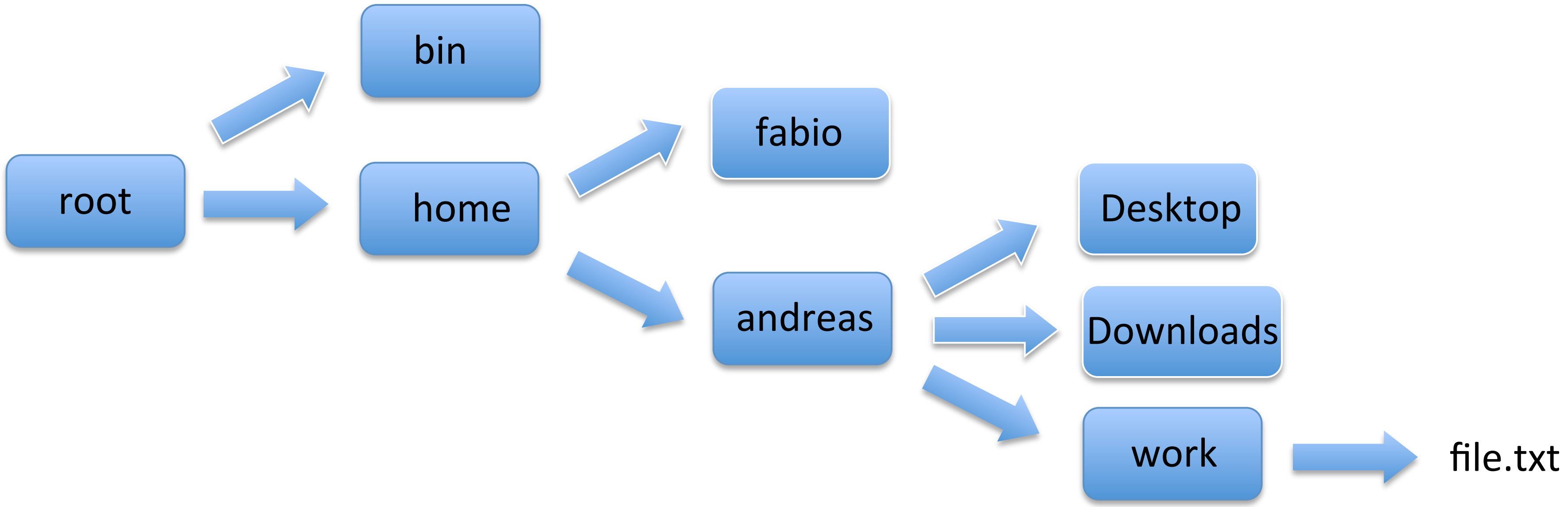


absolute path:

/home/andreas/work/file.txt

~/work/file.txt

File system

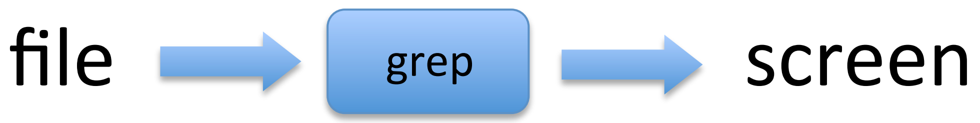


absolute path: /home/andreas/work/file.txt

~/work/file.txt

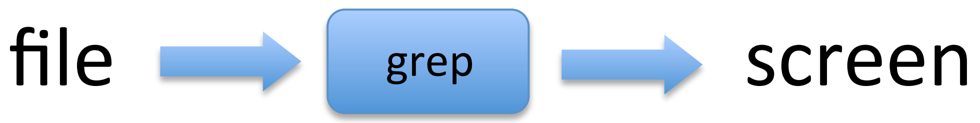
relative path from home folder: work/file.txt

Piping

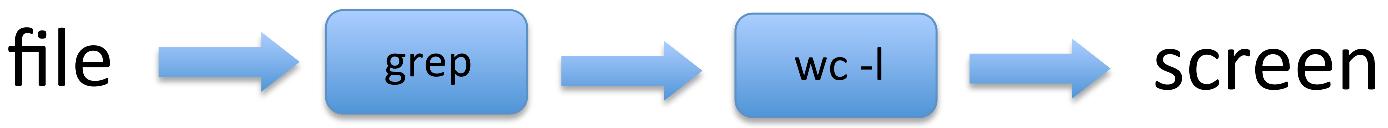


grep term file

Piping

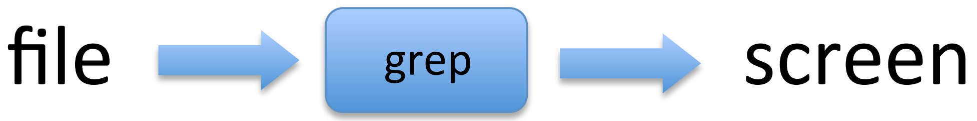


grep term file

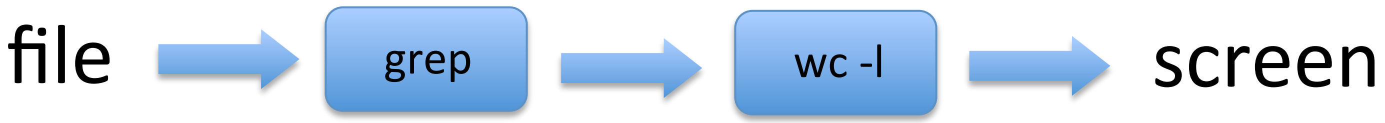


grep term file | wc –l

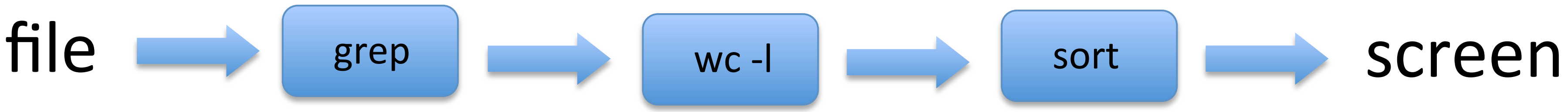
Piping



```
grep term file
```



```
grep term file | wc –l
```

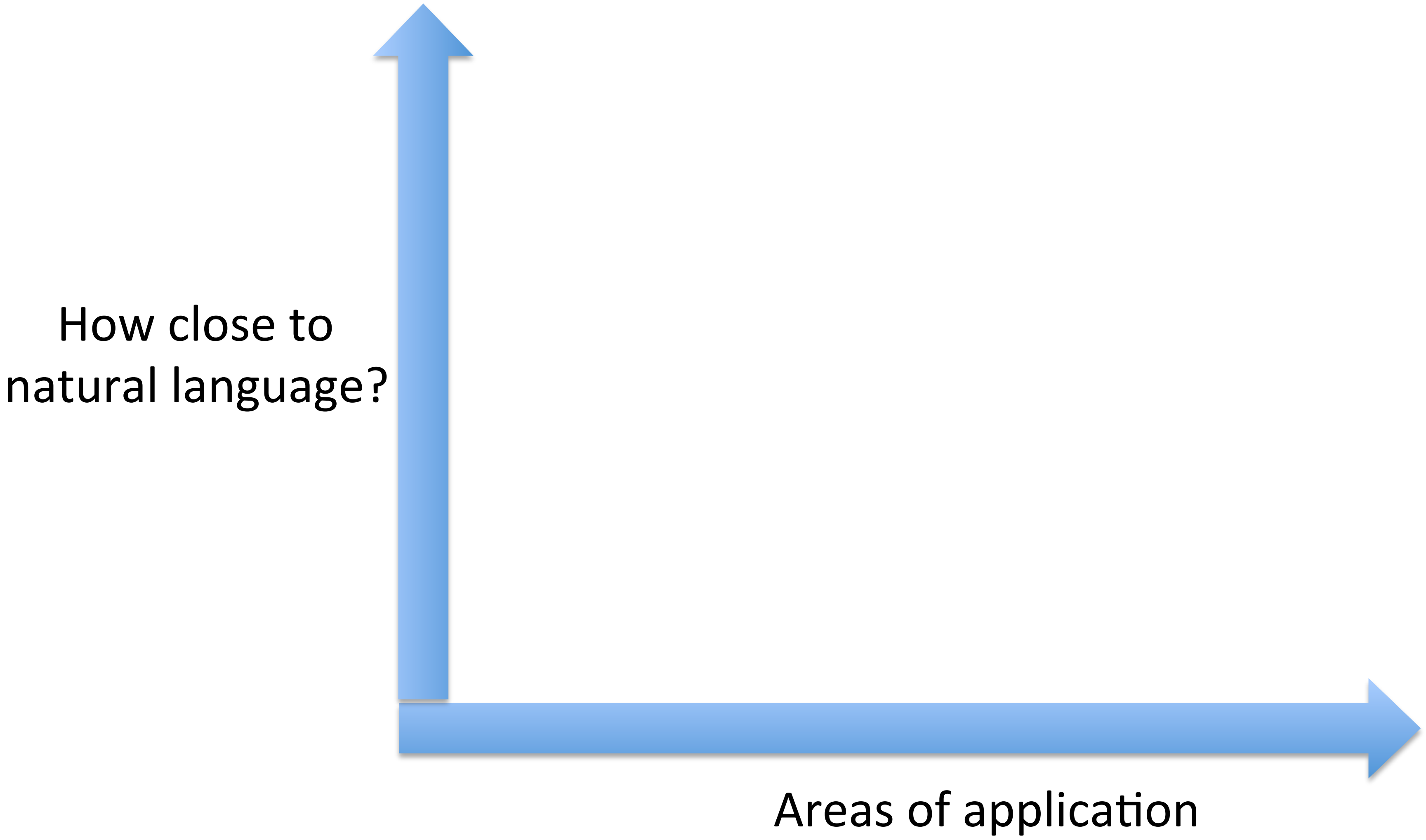


```
grep term file | wc –l | sort
```

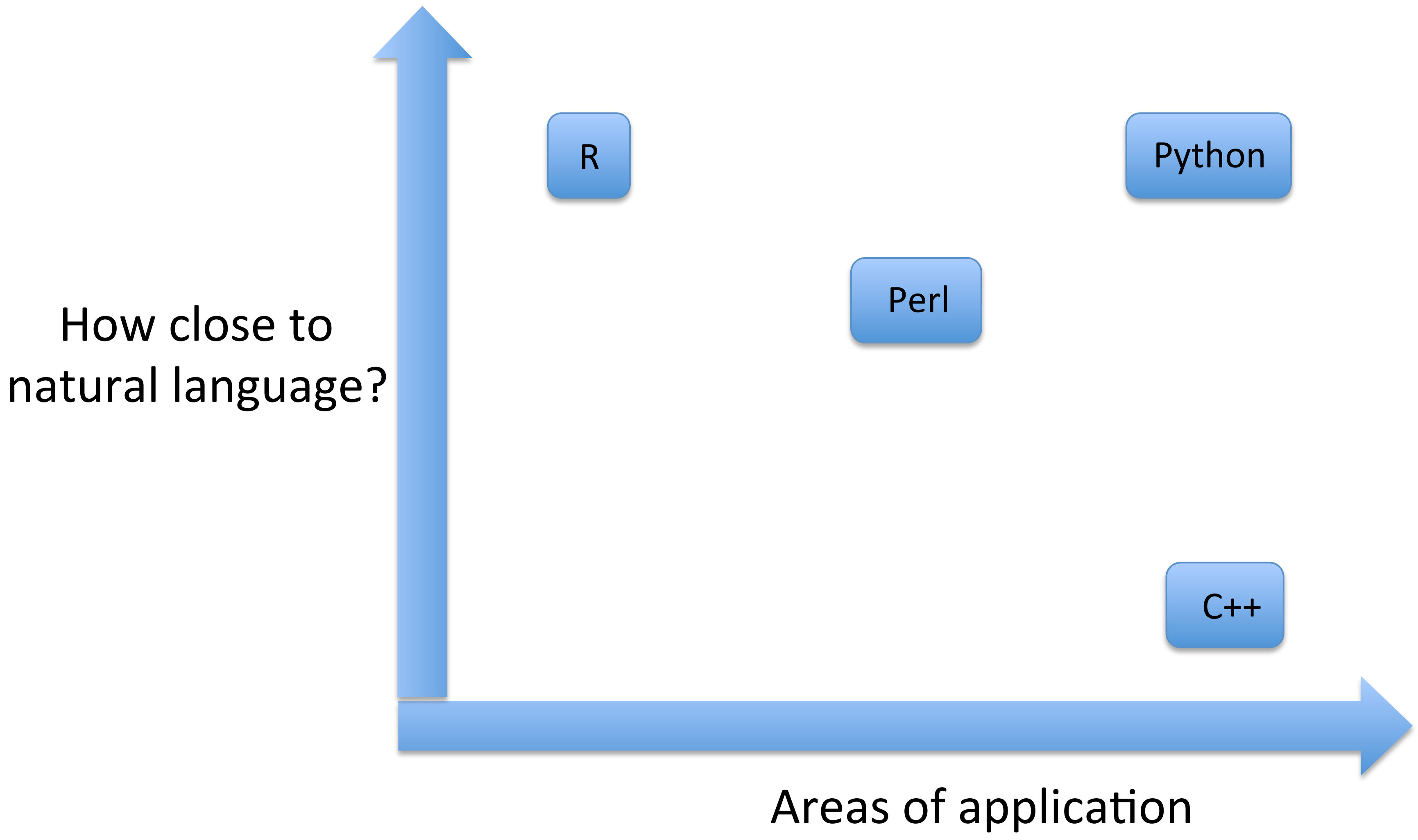
Python

Andreas Weller

Why Python?



Why Python?


How close to natural language?
R
Python
Perl
C++
Areas of application

Course overview

Python	Analogy	Example
Data types	Basic materials	Stone

Course overview

Python	Analogy	Example
Data types	Basic materials	Stone
Methods/ Functions	Associated verbs	Stone cutting

Course overview

Python	Analogy	Example
Data types	Basic materials	Stone
Methods/ Functions	Associated verbs	Stone cutting
Data structures	House structures	Stone wall

Course overview

Python	Analogy	Example
Data types	Basic materials	Stone
Methods/ Functions	Associated verbs	Stone cutting
Data structures	House structures	Stone wall
Control Flow	Coworkers	“Build me a wall from these stones!”

Data types: Numbers

- Find out the type of object:
  - *type(object)*

Data types: Numbers

- Find out the type of object:
  - *type(object)*

- 2 types of numbers:
  - integer: whole number
  - float: number with decimal

Next task!

Data types: Booleans

- 2 states: True or False
- Answer of a question

Data types: Booleans

- 2 states: True or False
- Answer of a question

- Identity:

  x in y, x not in y

  e.g. “Is beer in fridge”
- Comparisons:

  >, >=, <=, !=, ==

  e.g. “1 > 2”

Next task!

Variables

- a name for an object

Variables

- a name for an object

- assigned with =

```
a = 2
b = 3
a + b == 5
```

Data types: Strings

- String of letters without inherent meaning
- Always written in “”

Data types: Strings

- String of letters without inherent meaning
- Always written in “”
- “100” != 100!
- concatenate with +

  “And” + “reas” == “Andreas”

Next task!

Slicing

- Ways of data retrieval from an iterable
- Def Iterable: anything made of smaller parts

Slicing

- Ways of data retrieval from an iterable
- Def Iterable: anything made of smaller parts

- 0-based

A	B	C	D	E
0	1	2	3	4

- Uses [ ] brackets
- Pick item [2] or range [1:3]

Next task!

Functions and Methods

- Verbs of Python

- Functions stand alone

  type(“andreas”) == string

Functions and Methods

- Verbs of Python

- Functions stand alone
  type(“andreas”) == string
- Methods are tied to a data type
  “andreas”.upper() == “ANDREAS”

- Round brackets are always needed!

Easy scripting

- a script is a text file with ending “.py”

Easy scripting

- a script is a text file with ending ".py"

- execution with on bash with

```
python scriptname.py
```

Next task!

Structures: Lists

- array of objects
- object order is static

- written in []

  fruits = [“apples”, “oranges”, pineapple]

Structures: Lists

- array of objects
- object order is static

- written in []

  fruits = [“apples”, “oranges”, pineapple]

- sliceable

  fruits[0] == apples

Next task!

Structures: Dictionarys

- mapping of key:value pairs

- written in {}

grades = {“Tom”:”A+”, “Jim”:”C”, “Jane”:”A+”}

Structures: Dictionarys

- mapping of key:value pairs

- written in {}

  grades = {“Tom”:”A+”, “Jim”:”C”, “Jane”:”A+”}
- retrieval by dict[key]

  grades[“Tom”] == “A+”

- keys are unique, values not
- object order is random

Next task!

Flow control: For - loops

- "Do something to all elements in y"

  English:

  "Clean all fruits in the basket!"

Flow control: For - loops

- “Do something to all elements in y”

  English:

  “Clean all fruits in the basket!”

  Python:

```
For fruit in basket:
    clean fruit
```

- continues “action” until iterable is through
- “something” is defined on-the-fly
- the “action” is always indented with a TAB

Next task!

File I/O

- Files are traversed <u>once</u> and <u>row-by-row</u>
- Rows are an iterable

File I/O

- Files are traversed <u>once</u> and <u>row-by-row</u>
- Rows are an iterable

```
with open(filename) as myfile:
    for row in myfile:
        print row
```

- “myfile” and “row” are defined on-the-fly

Next task!

Modules

- module = optional Python code

Modules

- module = optional Python code
- can be imported to be used in your script

```
import scipy
scipy.mean(list_of_numbers)
```

Next task!