Course March 2022

Molecular
Electronic
Structure
Group

Lectures in software design in python

Abel Carreras

Introduction

Why Programming (software) design?

Readability

Understandable
by others (and you)

Maintainability

Fixing bugs

Upgrade to new
environments

Extensibility/flexibility

Add new
features

Do not waste your time!

Introduction

Readability

- You may want to reuse your code in the future
- You may want to share your code
- You will to explain how to use your code to others
- You will not have much time to read your code
- You may not remember how your code works anymore

Make your code readable!

Introduction

Maintainability

- Your OS will become obsolete
- Your Python version will become obsolete
- Your computer will become obsolete
- You will not have time to update your code
- You may not remember how your code works anymore

Make your code intuitive!

Introduction

Extensibility

- You may want to add new features to your code
- These features may conflict with your current structure
- You will not have time to change your code strucuture
- You may not remember how your code works anymore

Make your code flexible!

Introduction

The truth

- Your code (design) is crap: no users (maybe you in recent future)
- Your code is bad: future you is your only user
- Your code is fine: future you is your main user
- Your code is good: future you among some other researchers are users
- Your code is great: many researchers in your field may be users (future you included)

This is also for you!

Introduction

Like writing a paper!

What is a good design?

- **Logical and intuitive**
  People do not like to read manuals (and probably neither do you)
- **Less comments and better code**
  let the code speak for you
- **Divide and conquer**
  Properly organize you code in files, modules, functions, paragraphs,..
- **Explicit better than implicit (but..)**
  Simple and long better than short (compact) and complicated
- **(…) Make use of available good python modules**
  do not reinvent the wheel. Better notation is always nice!

Logical and intuitive

How?

- **Use conventions to write: PEP8**
  - ***variable names:***
    explicit names, lower case, spaces as underscore “_”
  - ***blank lines and spaces:***
    use them to separate logical blocks in your code
- **Aim for a good equation-code correspondence**
  write helper functions if necessary
- **Let the code guide the structure**
  minimize module imports by playing with scopes

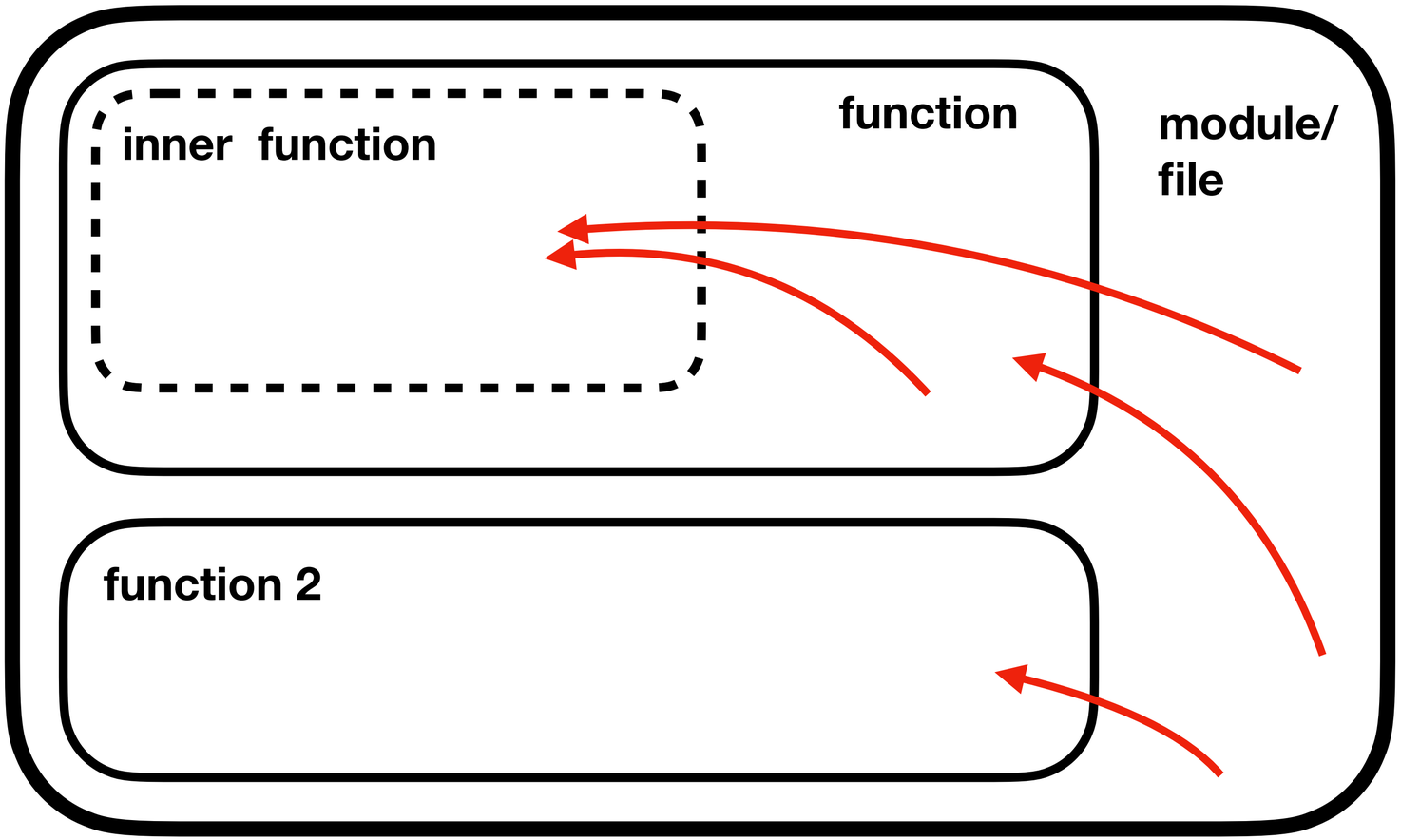
Example 1

Some initial advices

- **Use good code editor (IDE):**
  PyCharm, Visual studio, Spider, etc..
- **Get used to a version control system (VCS) —> git**
  very easy to use through IDE
- **Check the documentation (or StackOverflow)**
  do your research before coding
- **Prioritize standard library**
  over other obscure modules
- **Try to use widely compatible syntax**
  support old versions of Python

Example 2

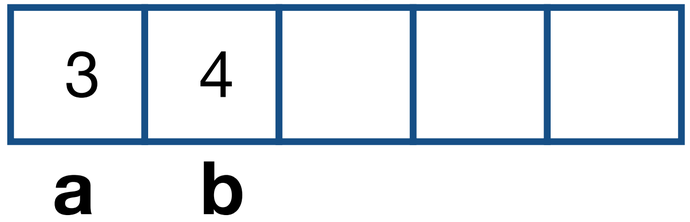
Variable Scope



Example 3

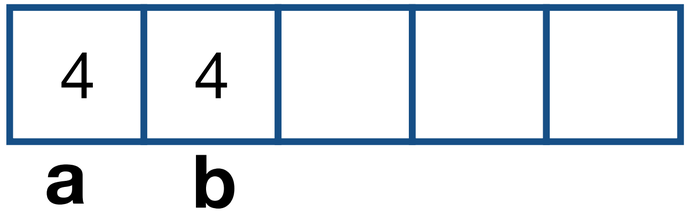
Fortran/C vs Python

RAM Memory




```
a=3
b=4
```

RAM Memory




```
a=3
b=4
a=b
```

Fortran/C vs Python

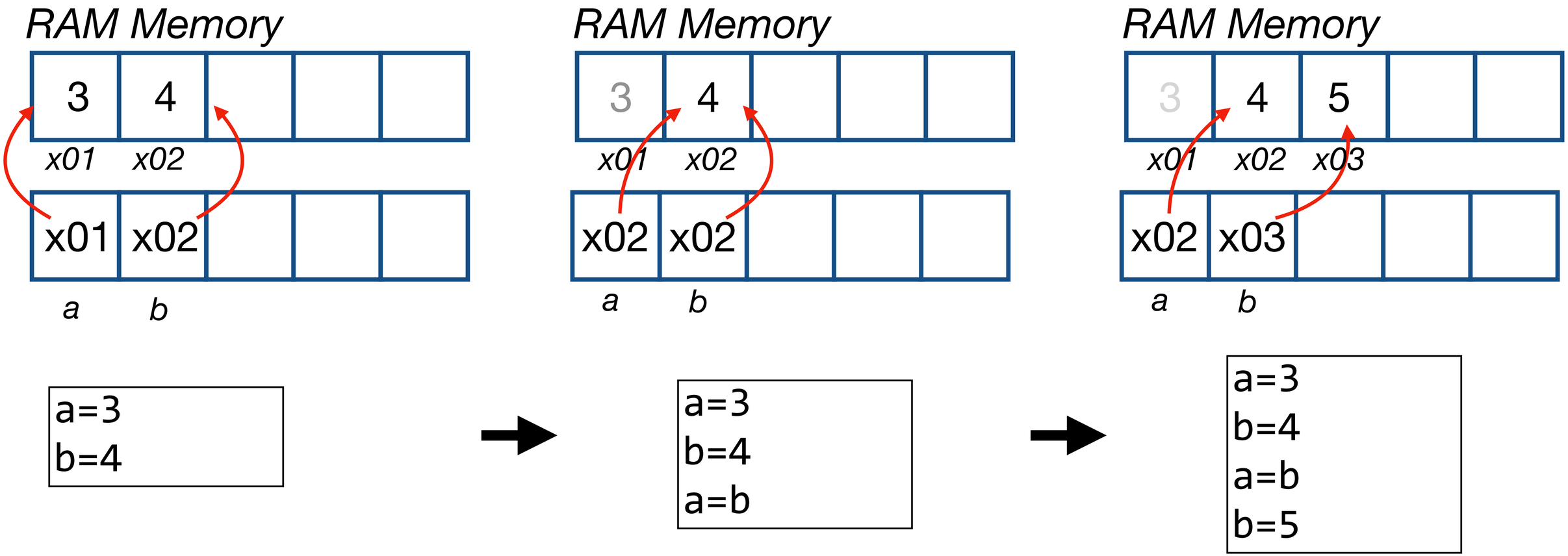
FORTRAN/C

- Small set of instructions
- Simple instructions very flexible
- Short documentation, lots of creativity
- Fast and insecure
- Compiled
- Not so nice syntax
- Fewer external libraries and not so easy to use

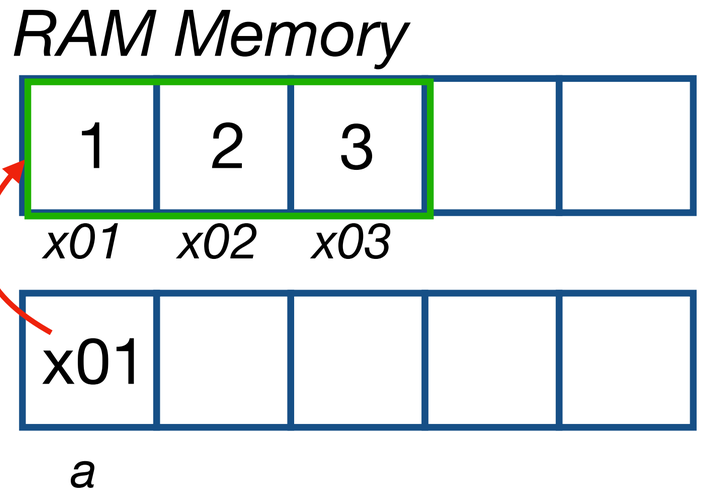
Python

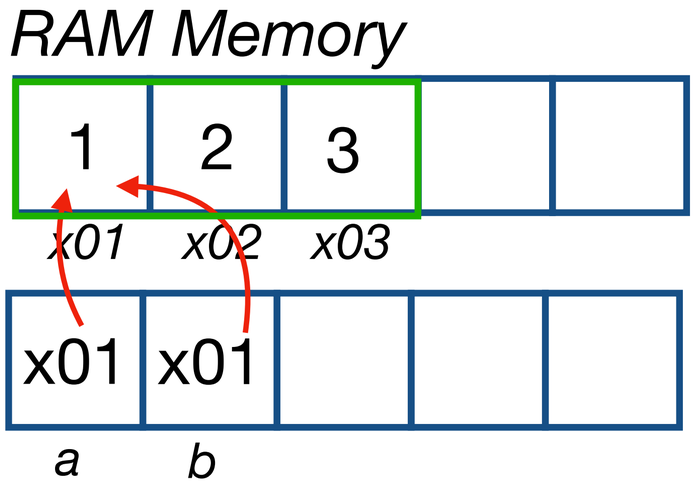
- Large set of instructions
- Complex instructions and very specific
- Long documentation, lots of research (and creativity)
- Slow and safe
- Interpreted
- Nice syntax
- Lots of external libraries easy to use

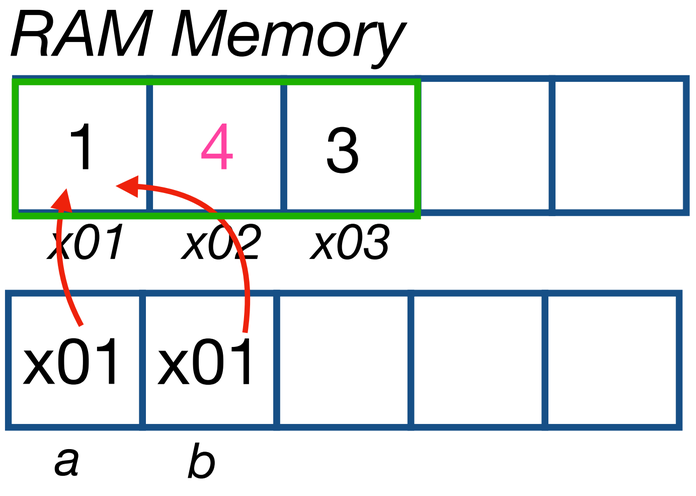
Fortran/C vs Python



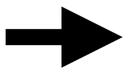
Fortran/C vs Python



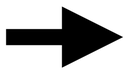





```
a=[1, 2, 3]
```



```
a=[1, 2, 3]
b=a
```



```
a=[1, 2, 3]
b=a
b[1] = 4
```