



# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# What is all this about?

# Programming in FORTRAN

## Introduction

# What is FORTRAN?

- 1 Short for FORMula TRANslator
- 2 Compiled language
- 3 Very widely used (even if you don't realise it)
- 4 60 years old, with 'minor' changes every once in a while

# Why learn FORTRAN?

- 1 Its FAST!
- 2 Well structured
- 3 Many good old codes are written in FORTRAN
- 4 Can easily be linked to high level languages like Python

# A little history

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

### Structure and Variables

### Control flow statements

### Arrays

### Subprograms

### Read and writing from/to file

### Fortran to Python interface

### Summary

- The first FORTRAN compiler was released in 1957
- FORTRAN had many releases important ones include F77, F90 and F95
- Until F90 was released FORTRAN operated in fixed form i.e. every command had to be given after six spaces.
- There are some new FORTRAN compilers out there recently released by Intel but have not had a wide acceptance and will not be mentioned any further

# Editor and Compiler

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

## Introduction

## Structure and Variables

## Control flow statements

## Arrays

## Subprograms

## Read and writing from/to file

## Fortran to Python interface

## Summary

## Editor

- Any editor can be used
- For the practical we will use gedit
- A FORTRAN script can be saved by either .f or .f90 or .f95 depending on the version it was written for

## Compiler

- The gfortran compiler (comes with gcc)
- Compile for equivalent releases you want to run.
- example for FORTRAN95: compile: `f95 hello.f95 -o hello`  
run: `./hello`

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# Structure

Program:structure.f90

```
PROGRAM Structure
    [Variable declarations]
    ...
    [code]
    ...
END PROGRAM
```

Every piece of code written in FORTRAN must have a main program to compile. This is equivalent to a main function in C. Each variable used needs to be declared before use.



# Structure

## Example

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

### Program: hello.f90

```
program Hello
    ! This program says hello
    print *, "Hello World!"
end program
```

- The asterisk following the keyword `print` tells the computer that the programmer will not be specifying the exact format of the printed answer. The default format, also called a list-directed format, is used.
- Comments are specified with `!`
- Strings are given between `" "`

# Variable declaration

## Variable types

- INTEGER
- REAL - 6 digits precision
- DOUBLE PRECISION - 15 digits precision
- LOGICAL -takes values TRUE or FALSE
- COMPLEX

## Implicit types

FORTRAN has types auto assigned to variables:

- If the first letter is i, k, l, m or n then by default they are set to type INTEGER, Otherwise they are set to type REAL. This can get confusing and you should turn it off at the start of the program with IMPLICIT NONE

## Example

# Programming in FORTRAN

## Structure and Variables

Program:First program.f90

```
PROGRAM Define
  Real :: a
  INTEGER :: b
  DOUBLE PRECISION :: c
  READ*, a,b,c
  c = a**b
  print*, c
END PROGRAM
```

## Constants

# Programming in FORTRAN

## Structure and Variables

Program:First program.f90

```
PROGRAM Define
  Real,parameter :: a=3
  INTEGER,parameter:: b=4
  DOUBLE PRECISION :: c
  c = a**b
  print*, c
END PROGRAM
```

Between numbers

Relation operators		
FORTRAN 77	FORTRAN 95	Equivalent
.LT.	<	less than
.LE.	<=	less than or equal to
.GT.	>	greater than
.GE.	>=	greater than or equal to
.EQ.	==	equal to
.NE.	/=	not equal to

# Logical expressions

## Between Logicals

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

A logical expression can be either TRUE or FALSE

Relation operators		
Operator	Equivalent bool	Priority
.AND.	and	left to right
.OR.	or	left to right
.NOT.	not	right to left
.EQV.	equivalent to	left to right
.NEQV.	not equivalent to	left to right

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements**
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# Control flow statements

## IF statement

### Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

#### Introduction

#### Structure and Variables

#### Control flow statements

#### Arrays

#### Subprograms

#### Read and writing from/to file

#### Fortran to Python interface

#### Summary

### IF structure

```
IF ('Logical expression') THEN
    ...
ELSE IF ('Logical expression') THEN
    ...
ELSE
    ...
END IF
```



# Control flow statements

## IF statement: example

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

sorting.f90

```
PROGRAM sort
  IMPLICIT NONE
  INTEGER :: a, b
  READ*, a
  READ*, b
  IF (a==b) THEN
    PRINT*, "a=b"
  ELSE IF (a>b) THEN
    PRINT*, "a>b"
  ELSE
    PRINT*, "a<b"
  END IF
END PROGRAM
```

## DO statement

## Control flow statements

```
DO i= start,fin,step
  ...
END DO
```

- i will firstly take the value of start
- at each loop step will be added to i
- until  $i \geq end$
- Equivalent to for loops

# Control flow statements

DO statement: example

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

loop1.f90

```
PROGRAM loop
  IMPLICIT NONE
  INTEGER:: a
  DO a=1,5,3 ! from 1 to 5 in steps of 3
    PRINT*, "a=", a
  END DO
END PROGRAM
```

This will print to the output:

a= 1

a= 4

## DO WHILE statement

## Control flow statements

```
DO WHILE ('Logical expression')
```

END DO

- The statements within the loop will run until the logical expression is false
- Equivalent to while loops

# Control flow statements

## DO WHILE statement: example

loop2.f90

```
PROGRAM loop_while
  IMPLICIT NONE
  INTEGER:: a
  a=1
  DO WHILE (a<=5)
    PRINT*, "a=", a
    a=a+3
  END DO
END PROGRAM
```

This will print to the output:

a= 1

a= 4

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

**Arrays**

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays**
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# Arrays in FORTRAN

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

### Structure and Variables

### Control flow statements

### Arrays

### Subprograms

### Read and writing from/to file

### Fortran to Python interface

### Summary

## Some Definitions

- FORTRAN arrays start from 1
- Support colon notation
- Like in C you should initialise the array before use

# Arrays

## Declare Arrays

### Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

### Structure and Variables

### Control flow statements

### Arrays

### Subprograms

### Read and writing from/to file

### Fortran to Python interface

### Summary

- A type declaration statement to declare the type and the number of the elements contained in the array.
- An array can be of any type: real, integer, logical, or double precision

### Declaration statement

```
REAL, DIMENSION(16):: voltage  
INTEGER, DIMENSION(2,4):: index  
LOGICAL, DIMENSION(2):: temp  
DOUBLE PRECISION, DIMENSION(3,2):: length
```



## Initialise Arrays

# Programming in FORTRAN

## Introduction

## Arrays

## Initialise arrays with assignment statement

```
REAL, DIMENSION(10) :: array1
DO i=1,10
    array1(i)=0.0
END DO
!or
array=(/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,/)

```

## Initialise Arrays

# Programming in FORTRAN

## Arrays

## Initialise arrays in type declaration statements

```
INTEGER :: i,j
REAL, DIMENSION(100) :: array1=1.
INTEGER, DIMENSION(5) :: array2=&
(/1, 2, 3, 4, 5 /)
INTEGER, DIMENSION(5) :: array3=(/(i, i=1, 5)/)
```

# Arrays

## 2-D or Rank-2 Arrays

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

### Initialise Rank-2 arrays with assignment statement

```
INTEGER, DIMENSION(4, 3) :: array1
DO i = 1,4
    DO j = 1,3
        array1(i, j)=j
    END DO
END DO
!OR
DO j = 1,3
    array1(:, j)=j
END DO
!OR
array1 = &
RESHAPE((/1,1,1,1,1,2,2,2,2,2,3,3,3,3 /), (/4,3/))
```

## 2-D or Rank-2 Arrays

## 2-D or Rank-2 Arrays

# Programming in FORTRAN

## Arrays

## Initialise Rank-2 arrays in type declaration statements

```
INTEGER, DIMENSION(4, 3) :: array1(4, 3)=&  
RESHAPE((/1,1,1,1,2,2,2,2,3,3,3,3/), (/4,3/))
```

# Arrays

## Array using Tips

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

If two arrays are the same shape, you can use them in ordinary arithmetic operations and the operation will be applied on an element-by-element basis.

### Whole Array Operation

```
REAL, DIMENSION(4) :: a=(/1., 2., 3., 4./)
REAL, DIMENSION(4) :: b=(/5., 6., 7., 8./), c
DO i=1,4
    c(i)=a(i)+b(i)
END DO
!OR
c(:)=a(:)+b(:)
!Or even
c=a+b
```

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms**
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary

# Subprograms

## Functions

### Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

#### Introduction

#### Structure and Variables

#### Control flow statements

#### Arrays

#### Subprograms

#### Read and writing from/to file

#### Fortran to Python interface

#### Summary

- A function takes in a number of arguments and returns one value
- The value that is returned has the name of the function.
- A function can be an INTEGER, REAL, COMPLEX, LOGICAL, DOUBLE PRECISION
- FORTRAN has some functions inbuilt, ex SIN(x)
- when a function is used, it needs to be declared in the main program as an external function  
Example: integer, external :: fun

# Subprograms

## Functions: structure

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

## Function

```
'Type' FUNCTION NAME(args)
      VARIABLE DECLARATIONS
      ...
      name = ...
END FUNCTION NAME
```





# Subprograms

## Subroutines

### Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

### Structure and Variables

### Control flow statements

### Arrays

### Subprograms

### Read and writing from/to file

### Fortran to Python interface

### Summary

- A Subroutine is equivalent to void functions in C.
- Input arguments, makes calculations upon them and returns them
- Specify the intent of the arguments in the declaration part within the subroutine
- The subroutine is used by using the CALL command in the main program function
- The intent of each variable should to be stated at the subroutine's variable declarations
- these can be
  - intent(in)
  - intent(out)
  - intent(inout)

# Subprograms

## Subroutine: Structure

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

### Subroutine

```
SUBROUTINE NAME(args)
    VARIABLE DECLARATIONS
    ...
END FUNCTION NAME
```

# Subprograms

## Subroutines: Example

Programming  
in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

## Subroutine

```
PROGRAM CALCULATION
  IMPLICIT NONE
  EXTERNAL :: trig
  REAL x, y, z
  x= 7; y = 9
  CALL trig(x,y,z)
END PROGRAM

SUBROUTINE trig(x,y,z)
  IMPLICIT NONE
  REAL, INTENT(IN):: x, y
  REAL, INTENT(OUT):: z
  z = sin(x) * tan(y)
END SUBROUTINE trig
```

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file**
- 7 Fortran to Python interface
- 8 Summary

# Opening files

# Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

## Introduction

Read and  
writing  
from/to file

- To read and write to/from a file it needs to be opened
- The action that is going to be taken needs to be stated
- Opening needs to have a unit associated with it
- Units can be any integer except 0,5 and 6
- The file should be closed after work is done

# Open and Read

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

### Introduction

### Structure and Variables

### Control flow statements

### Arrays

### Subprograms

### Read and writing from/to file

### Fortran to Python interface

### Summary

## Initialise arrays with FORTRAN READ statements

Assume a file initial.dat contains the values:

1 1 1 1 2 2 2 2 3 3 3 3

```
INTEGER, DIMENSION(4,3) :: array1  
OPEN (7, file='initial.dat', action='read')  
READ (7, *) array1  
CLOSE(7)
```

## Open and write

## Initialise arrays with FORTRAN READ statements

Assume a file initial.dat contains the values:

1 1 1 1 2 2 2 2 3 3 3 3

```
INTEGER, DIMENSION(4,3) :: array1
OPEN (7, file='initial.dat', action='write')
WRITE (7, *) array1
CLOSE(7)
```



# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface**
- 8 Summary

# Motivation

# Programming in FORTRAN

## Introduction

# Fortran to Python interface

There are many codes that are already written in FORTRAN and may needed to be run as part of a code in a high level language.

- MEX files with Matlab (not covered)
- f2py in python

f2py allows the conversion of FORTRAN functions and subroutines in to python modules which can be imported and called when needed.

- Command line
- Comes with Numpy
- Allows values to be passed back to Python and called again
- Supports Numpy arrays
- runs from terminal
- can support C++ conversion

In general when the compiler is not specified f2py will try and compile FORTRAN77 failing that it will try FORTRAN90 and finally it will try C++

## How to use

Suppose there is a file `Process.f` which contains functions and subroutines

To create the Python module:

- `f2py -c process.f -m modulename`
- This can then be called from python as:
  - `import modulename`
  - Use the functions and subroutines when needed as you would in Python
  - example: `modulename.function(10.)`
- There are various options and tweaks to `f2py` that can be found by typing `f2py` in terminal

# Outline

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- 1 Introduction
- 2 Structure and Variables
- 3 Control flow statements
- 4 Arrays
- 5 Subprograms
- 6 Read and writing from/to file
- 7 Fortran to Python interface
- 8 Summary**

# Summary

## Programming in FORTRAN

Ioannis  
Begleris  
Hao Wang

Introduction

Structure and  
Variables

Control flow  
statements

Arrays

Subprograms

Read and  
writing  
from/to file

Fortran to  
Python  
interface

Summary

- Structure of FORTRAN
- Variables
- Control flow statements
- Subprograms
- Reading and writing from/to a file
- FORTRAN to Python interface