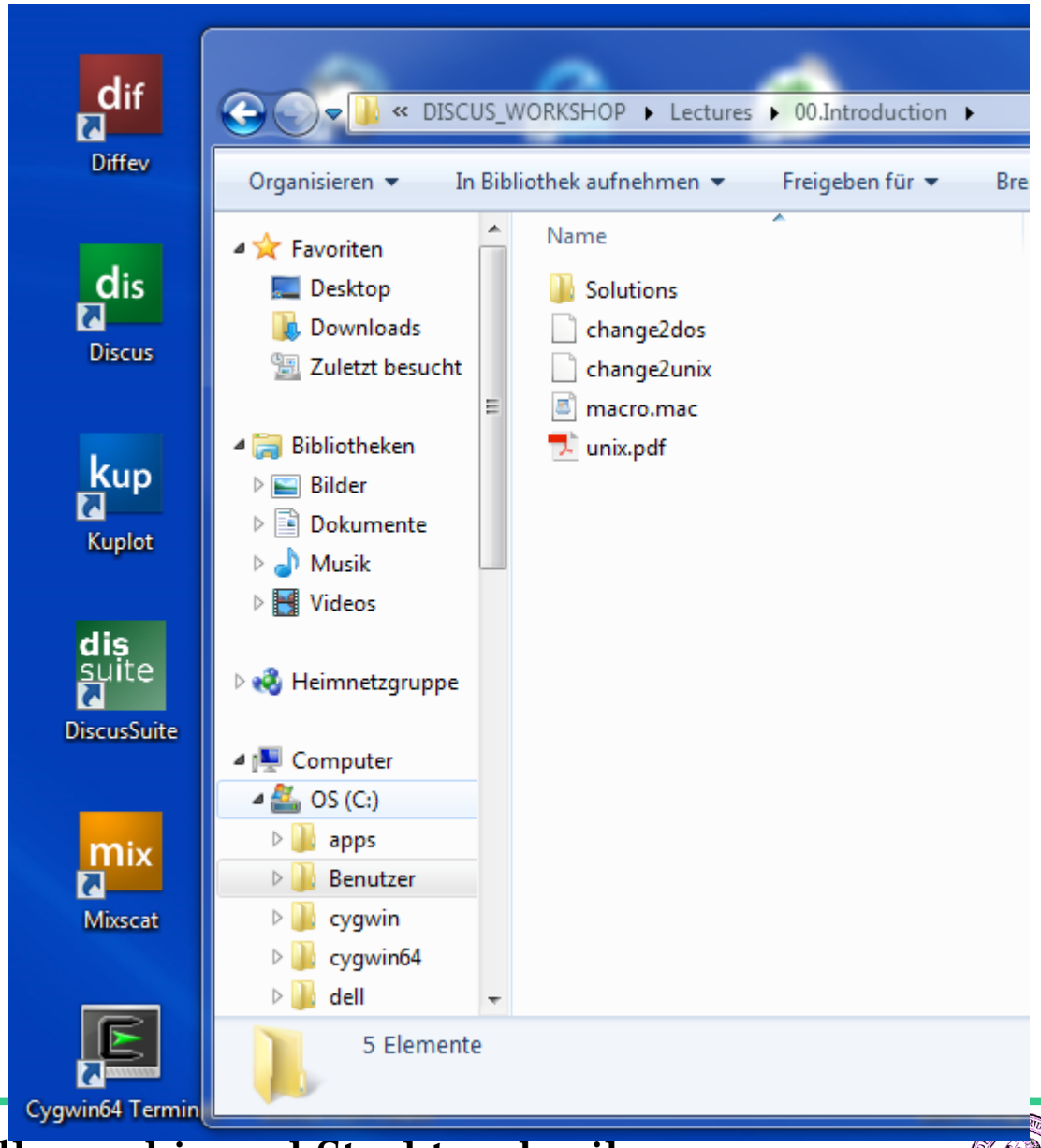
Introduction to syntax

Simulate a crystal structure

Open in Windows Explorer: Lectures\00.Introduction

Start DISCUS_SUITE

You should see:



Simulate a crystal structure

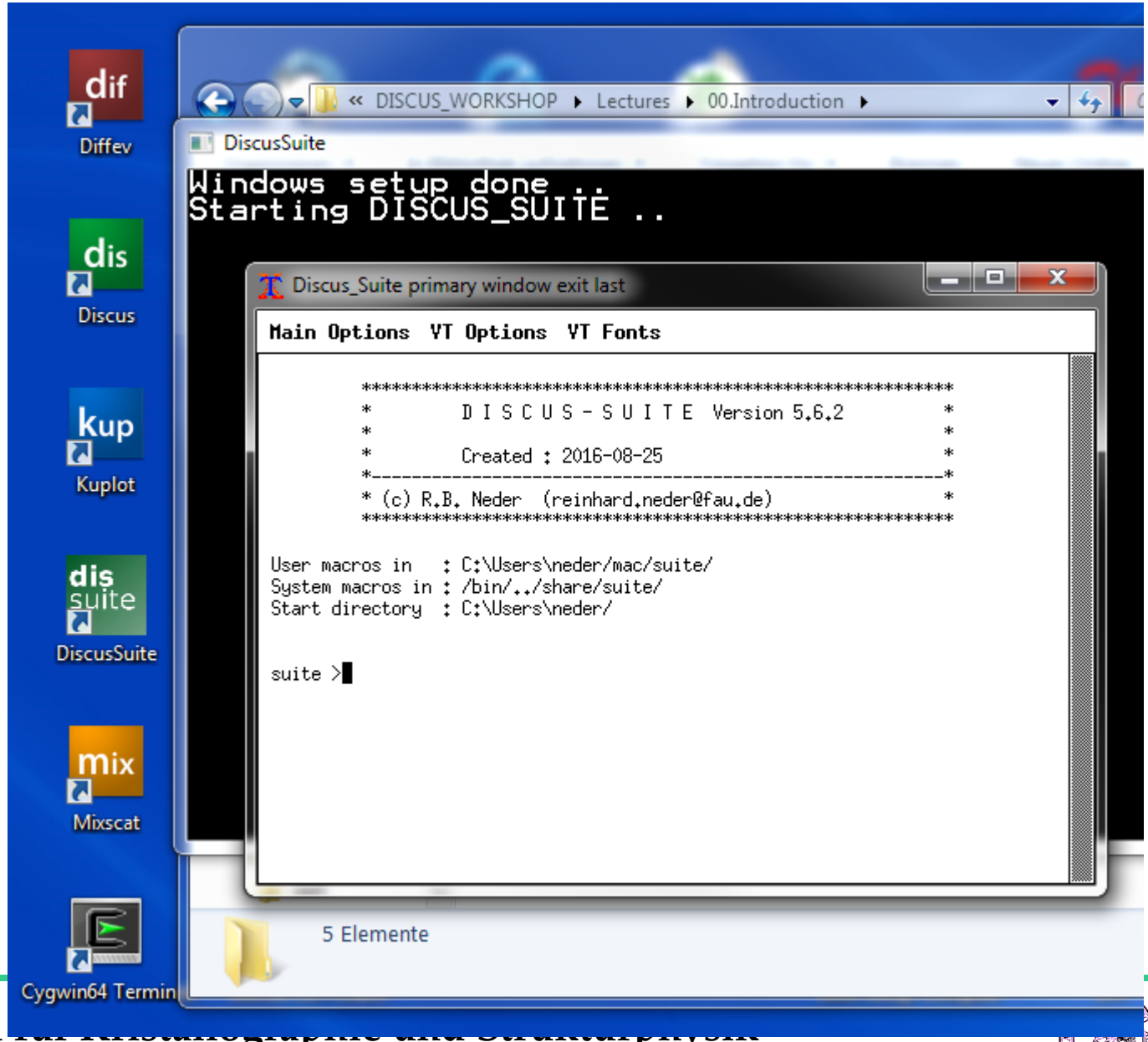
Open in Windows Explorer: Lectures\00.Introduction

Start DISCUS_SUITE

DISCUS_SUITE
needs to be in correct folder

type:

You should see:



Simulate a crystal structure

Open in Windows Explorer: Lectures\00.Introduction

Start DISCUS_SUITE

DISCUS_SUITE
needs to be in correct folder

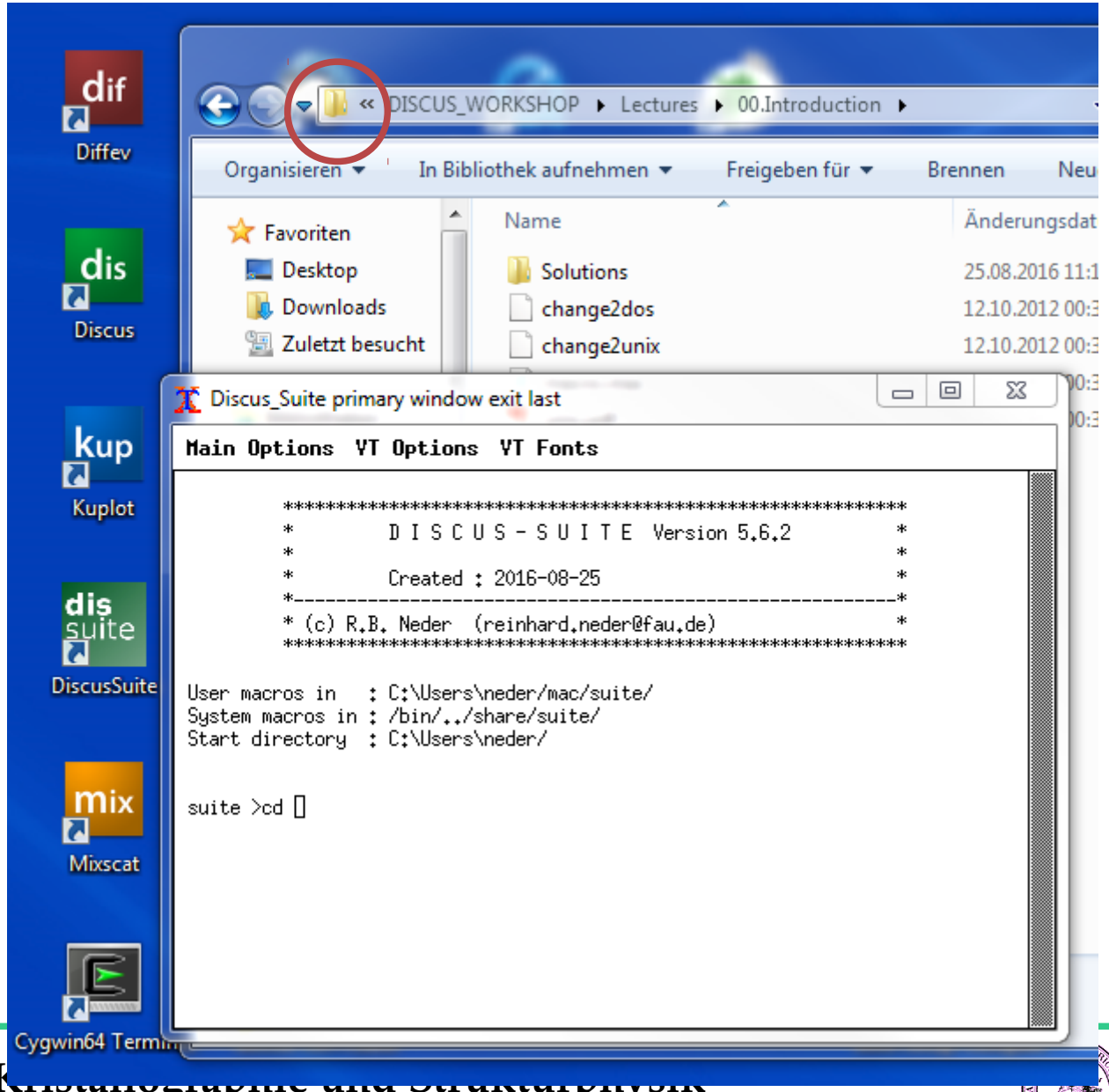
type:

discus > cd

NO RETURN YET !!

Left click on folder icon
Full path will be in blue
Copy with CTRL c
Activate DISCUS_SUITE Window
Paste by SHIFT + middle mouse button

You should see:



Simulate a crystal structure

Open in Windows Explorer: Lectures\00.Introduction

You should see:

Start DISCUS_SUITE

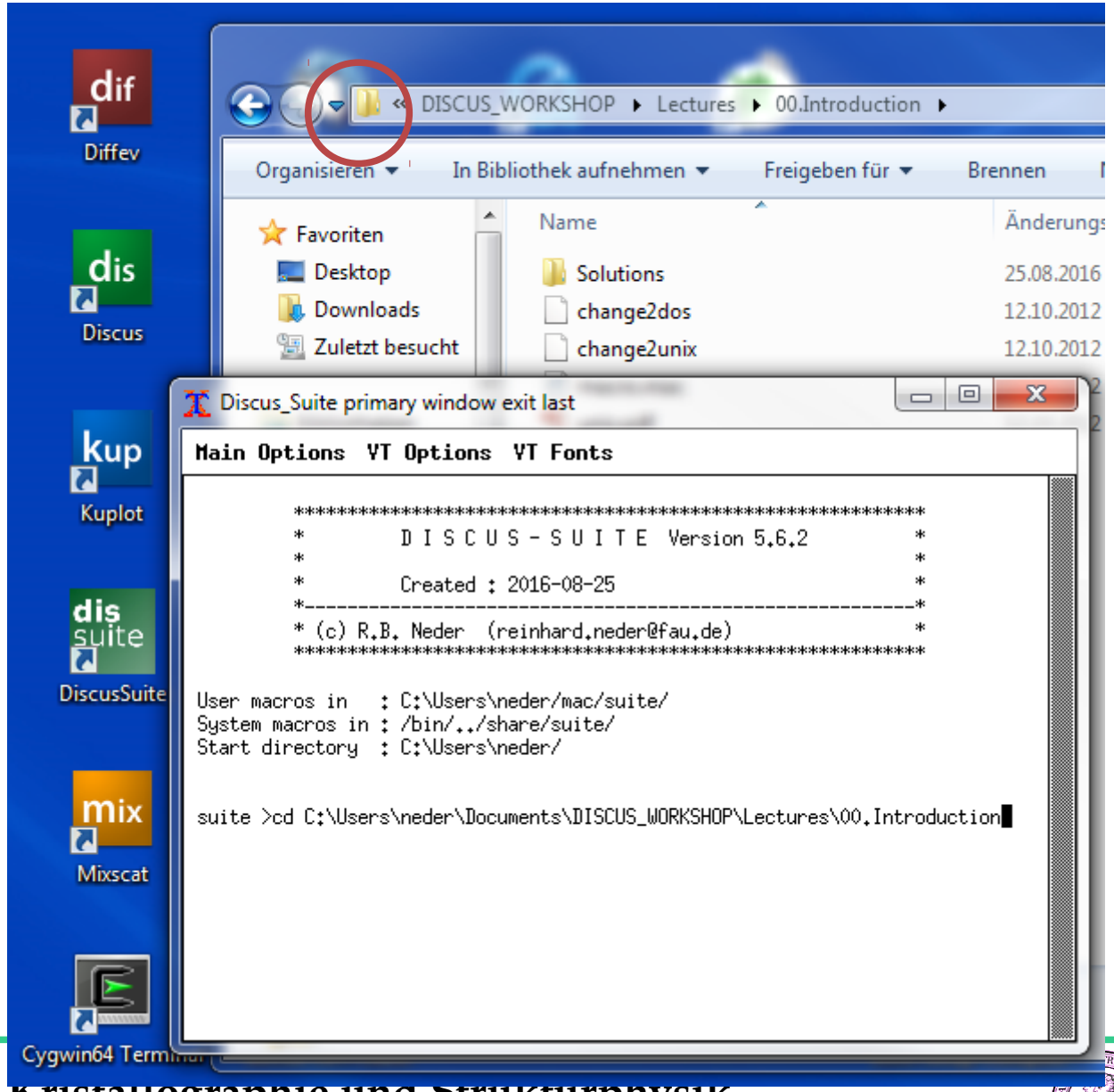
DISCUS_SUITE
needs to be in correct folder

type:

discus > cd

NO RETURN YET !!

Left click on folder icon
Full path will be in blue
Copy with CTRL c
Activate DISCUS_SUITE Window
Paste by SHIFT + middle mouse button



Simulate a crystal structure

Open in Windows Explorer: Lectures\00.Introduction

Start DISCUS_SUITE

DISCUS_SUITE
needs to be in correct folder

type:

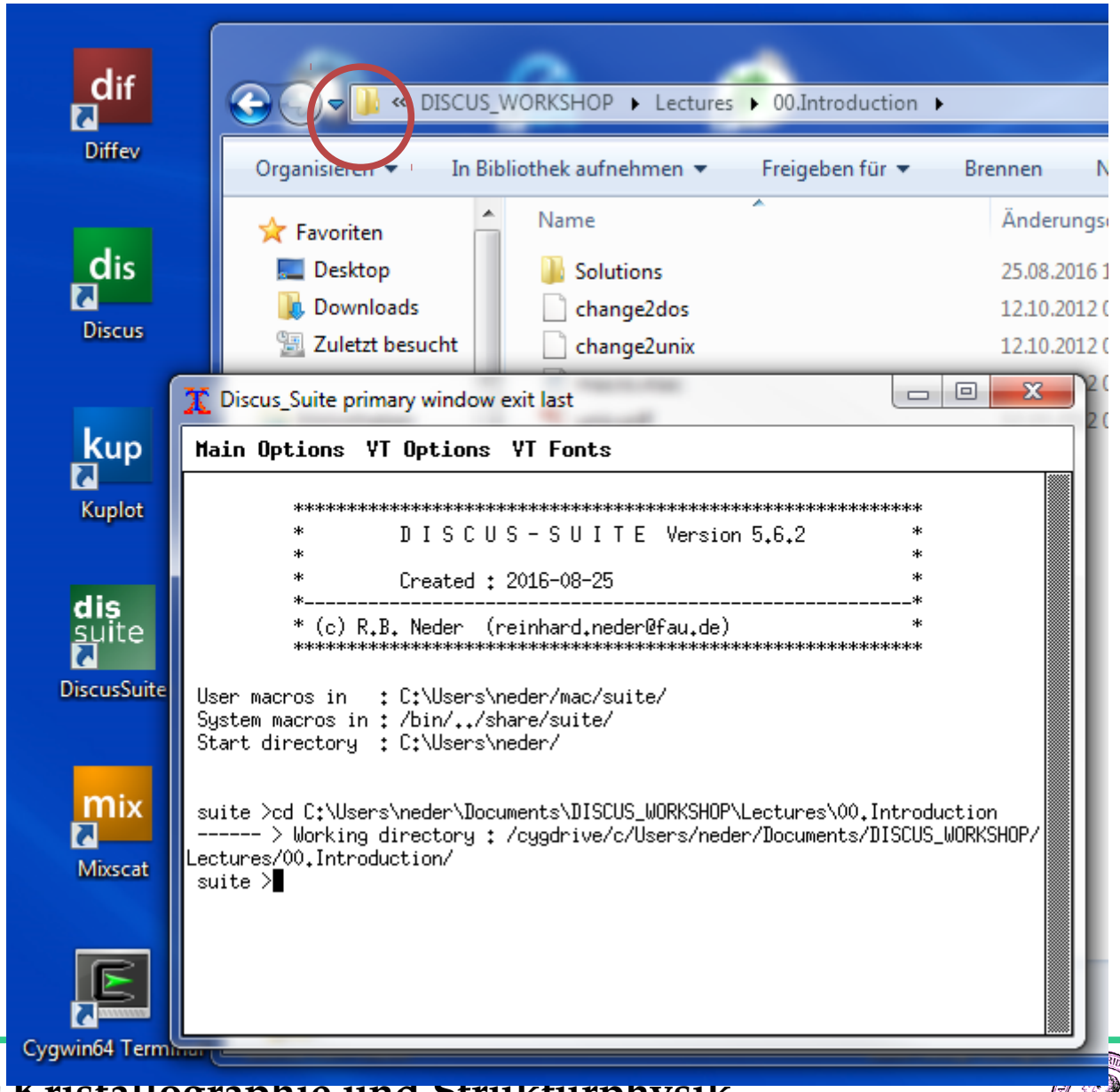
discus > cd

NO RETURN YET !!

Left click on folder icon
Full path will be in blue
Copy with CTRL c
Activate DISCUS_SUITE Window
Paste by SHIFT + middle mouse button

HIT RETURN

You should see:



Simulate a crystal structure

Open in Windows Explorer:

Lectures\00.Introduction

You should see:

Start DISCUS_SUITE

DISCUS_SUITE
needs to be in correct folder

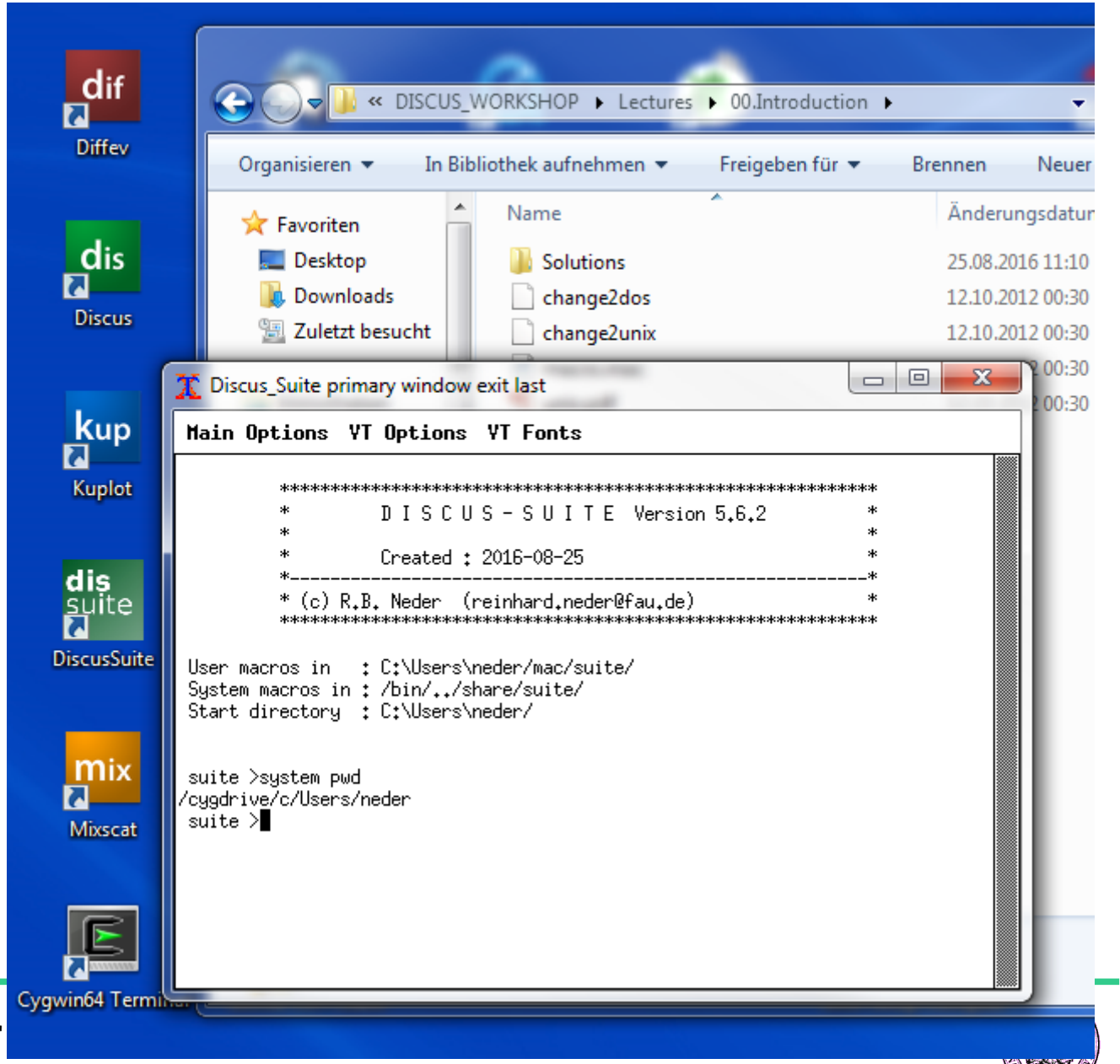
type:

discus > system pwd

Default folder is:
C:\Users\your_user_name

Within this folder you have:
Downloads
Documents
Etc. etc

Change to a folder via
cd path_to new_folder



DISCUS, main commands

to start: Type ***discus*** at the main discus_suite prompt

help	Enters help menu
help <command>	gives help on specific command leave help by simple return
exit	ends DISCUS section Warning, nothing is saved automatically, DISCUS does NOT ask
@macro	executes commands listed in file „macro.mac“
learn <filename.mac>	Start to learn everything you type into a macro called <filename.mac>
lend	stops the learn process the macro may now be used as: @<filename>
manual section:<name>	Opens the manual on section <name> suite, discus, diffev, kuplot, package

DISCUS, main commands

Syntax of DISCUS commands:

command
just a command name, which may be abbreviated

help
echo
fourier

command parameter, parameter

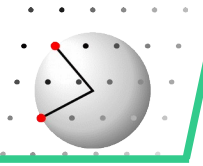
a command name followed by at least one blank
and a list of parameters, all separated by comma

evaluate 4+5
insert si, 0, 0, 0 , 1.0

command parameter, parameter # The rest of the line is a comment
command parameter, parameter ! The rest of the line is a comment

text following a '#' is treated as comment and ignored

text following a '!' is treated as comment and ignored



The command language

Variables

SUITE
DISCUS
KUPLOT
DIFFEV

```
i[0] = 3
i[500] = i[0] + 2
```

i[0] to i[500] is an array of integer variables

```
r[0] = 3.1415
r[500] = -1.4e5
```

r[0] to r[500] is an array of real variables

```
variable real, pi
variable real, length,10.0
variable integer, number

pi = 3.1415
length = length * 2
number = 4
```

further variables may be defined by the user and used just as the predefined variables

---

DISCUS

n[1]	Total number of atoms in the crystal
n[2]	number of different atom types
x[index]	x position of atom no. index
y[index]	y position of atom no. nndex
z[index]	z position of atom no. index

SUITE
DISCUS
KUPLOT
DIFFEV

```
i[0] = 3
i[500] = i[0] + 2**4

r[0] = 3.1415
r[ i[0] ] = cos(r[0])

evaluate acosd(60.00)

variable real, pi
pi = 4*atan(1.0)

evaluate 1./(6.+4.)
```

Variables and fixed numbers may be used in arithmetic operations

the usual hierachy of operation applies

( ) brackets may be used to change the sequence of operations

r[0] to r[500] is an array of real variables

evaluate just calculates the result of an expression and displays the result on the screen

Homework 1

start a learn process into a file called homework1.mac

define an integer valued variable called „product“

store into this variable the result of the product of numbers 1, 2, 3, 4, 5

evaluate the result of the variable

Stop the learn process

Execute the macro homework1.mac to verify your commands

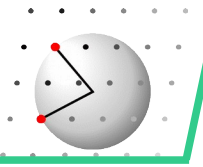
At the discus prompt type: @homework1

Homework 1 solution

```
variable  integer,  product

product  = 1 * 2 * 3 * 4 * 5

evaluate product
```



DISCUS KUPLOT DIFFEV	sin(arg) sind(arg) cos(arg) cosd(arg) tan(arg) tand(arg)	usual trigonometric functions argument in radian or in degrees cos**d**(arg)
	asin(arg) ...	inverse trigonometric functions
	sqrt(arg)	square root of argument
	exp(arg)	exponential function e^{arg}
	ln(arg)	natural logarithm
	abs(arg)	absolute value
	int(arg)	conversion to integer 1.6 ==> 1
	nint(arg)	conversion to next integer 1.6 ==> 2
	frac(arg)	fractional part 1.6 ==> 0.6
	min(arg1, arg2)	minimum of the two arguments (also max)
	mod(arg1, arg2)	modulo function rest after integer division arg1 - int(arg1/arg2)
	ran(arg)	uniform random number
	gran(arg)	Gaussian distributed random number
	logn(arg,arg)	lognormaly distributed random number
	pois(arg)	Poison distributed random number

DISCUS
KUPLOT
DIFFEV

```
if( condition ) then
      block1
elseif( condition2) then
      block2
...
else
      block3
endif
```

if condition 1 is true then
statements in block 1 are executed
otherwise, if condition 2 is true then
statements in block 2 are executed

otherwise, (no condition was true)
statements in block 3 are executed
final line of an if block

condition:
expression operator expression

conditions are a logical comparison or
an arithmetic comparison

operators:

<	.lt.
=	.le.
==	.eq.
>=	.ge.
>	.gt.

less than; less than or equal; equal;
greater than or equal; greater than

.and.
.or.
.not.

EXAMPLE:

```
variable real,value
value = ...
...

if( 3.14 > asin(value) .or. value == 0.0) then
```

DISCUS
KUPLOT
DIFFEV

```
do counter= start,end,increment
      block1
enddo
```

Loop with fixed number of cycles

```
do while(logical condition)
      block2
enddo
```

loop is executed as long as the condition is true

```
do
      block3
enddo until (condition)
```

loop is executed until the condition is true

EXAMPLE:

```
do i[0]=1,10,1
  evaluate i[0]
enddo
```

EXAMPLE:

```
i[0]=1
do while (i[0] < 11)
  evaluate i[0]
  i[0] = i[0] + 1
enddo
```

DISCUS
KUPLOT
DIFFEV

```
variable integer, counter
variable integer, start,  1
variable integer, ende, 10

do counter = start, ende
  evaluate counter
enddo
```

A macro file contains DISCUS commands, typed as explicitely needed here in file loop.mac

```
@loop.mac
```

the „@“ starts reading the commands from the macro file

MODIFICATION:

```
@loop.mac 1,10

@loop.mac 2, 99
```

```
do counter=$1,$2
  evaluate counter
enddo
```

string „$1“ is replaced by the first parameter

```
@trigo.mac cosd, 10

@trigo.mac sind, 60
```

File trigo.mac:

```
evaluate $1($2)
```

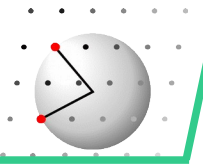
Homework 2

Create a macro that

takes one parameter, a number n

calculates the product 1 * 2 * … * n and stores this into a variable

evaluates the value of the variable



Homework 2

Create a macro that

takes one number as parameter n

calculates the product 1 * 2 * … * n and stores this into a variable

evaluates the value of the variable

```
variable  integer,  product
variable  integer,  loop
variable  integer,  start
variable  integer,  finish
product  =  1
start    =  1
finish    = $1
do loop = start, finish
  product  = product  * loop
enddo

evaluate product
```

Homework 3

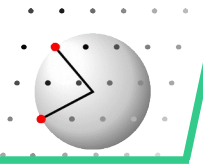
write a macro that

takes one parameter, a real valued number

if the number is larger than zero calculate the square root

if the number is less than zero calculate the square

display the number and the result

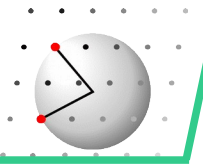


Homework 3

```
variable real, result

if ( $1 > 0.0) then
    result = sqrt( $1)
elseif ($1 < 0.0) then
    result = $1**2
else
    result  = 0.0
endif

evaluate $1
evaluate result
```



DISCUS KUPLOT DIFFEV	date(0) fdate(0)	date in format CCYYMMDDhhmmss.ss Date in format Day Mon DD hh:mm:ss CCYY
	fmodt('name')	File modification date
	getcwd()	Current directory / folder
	getenv('name')	Get environment variable
	length(<string>)	Length of character string
	index(<string>, <sub> [,"BACK]")	Position of strign <sub> in <string>
	isvar(<string>)	Is <string> a user variable Yes/No
	isexp(<string>)	Is <string> a valid expression Yes/No

Exercise

Run the macro „loop.mac“ with a single parameter,
an arbitrary whole number
suite> @loop.mac 40

Run macro with HUGE number
suite> @loop.mac 123456789

Interrupt with a CTRL-c