Computational Analysis of Physical Systems

About the course

The course - Topics

1. Open-source softwares in science,
   introduction to Python, data types, basic I/O operations
2. Basic constructions in Python (loops, conditions)
3. Operations on arrays and plotting commands
4. Random numbers
5. Functions
6. Matrix operations
7. Python as a MATLAB-like computation tool
8. Tkinter and graphical user interface
9. Object-oriented programming with Python
10. Interaction of C/C++ languages and Python
11. Symbolic computation with Python

The course - Grading

Activity	Quantity	Effect on Grading, %
Midterm Exam	**1**	**20%**
Quizzes	**10**	**20%**
Homeworks	**7**	**20%**
Final Exam	**1**	**40%**

QUIZZES:
You will be free to **cooperate** in quizzes and working with a friend will be encouraged.
Please remember that you will be **responsible individually** from the result
and be expected to explain your answer.

HOMEWORKS:
Homeworks will be assigned on NINOVA. **Belated homeworks will not be accepted**.
You must upload your homework to NINOVA before the deadline.
All homeworks **showing an effort for solution** will be **fully** graded.

The course - References

Textbook	Jaan Kiusalaas**,** *Numerical Methods in Engineering with Python*, Cambridge University Press, New York, 2010.
Other References	-Michael Dawson, *Python Programming for the Absolute Beginner*, Course Technology, Boston, 2010 -Mark Lutz, *Programming Python*, O'Reilly Media, California, 2011 -James Payne, *Beginning Python*, J. Wiley & Sons Inc, Indianapolis, 2010 - Hans Petter Langtangen, *A Primer on Scientific Programming with Python*, Springer, Dordrecht, 2011

Computational Analysis of Physical Systems (Lecture I)

Open-source softwares in science, Introduction to Python,

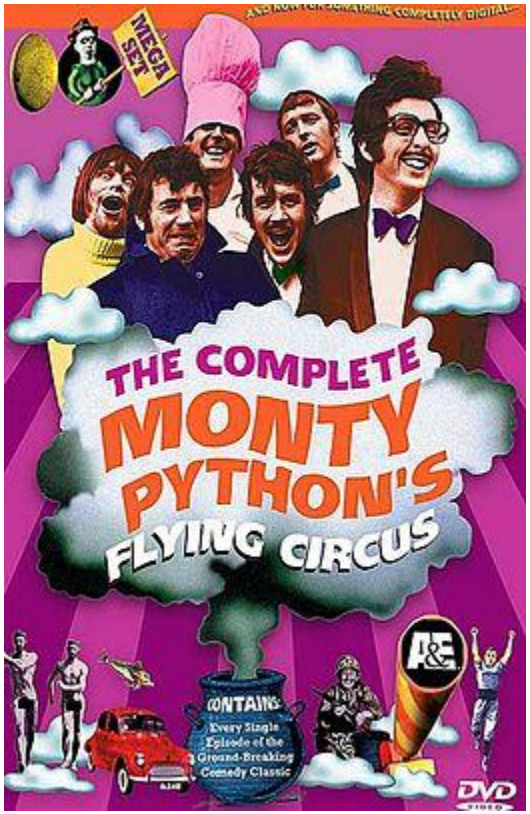
Data types, Basic I/O operations

Open-source softwares in science

- OS: GNU Linux
- Office: openoffice.org, Libreoffice
- Writing scientific documents: LaTeX
- Programming lang.s: C/C++, Fortran, Python, …
- Programming packages: Octave (MATLAB), Maxima (Symbolic mathematics), SAGE (Num/Sym mathematics)
- And many more specialized packages and programs.

Introduction to Python-1

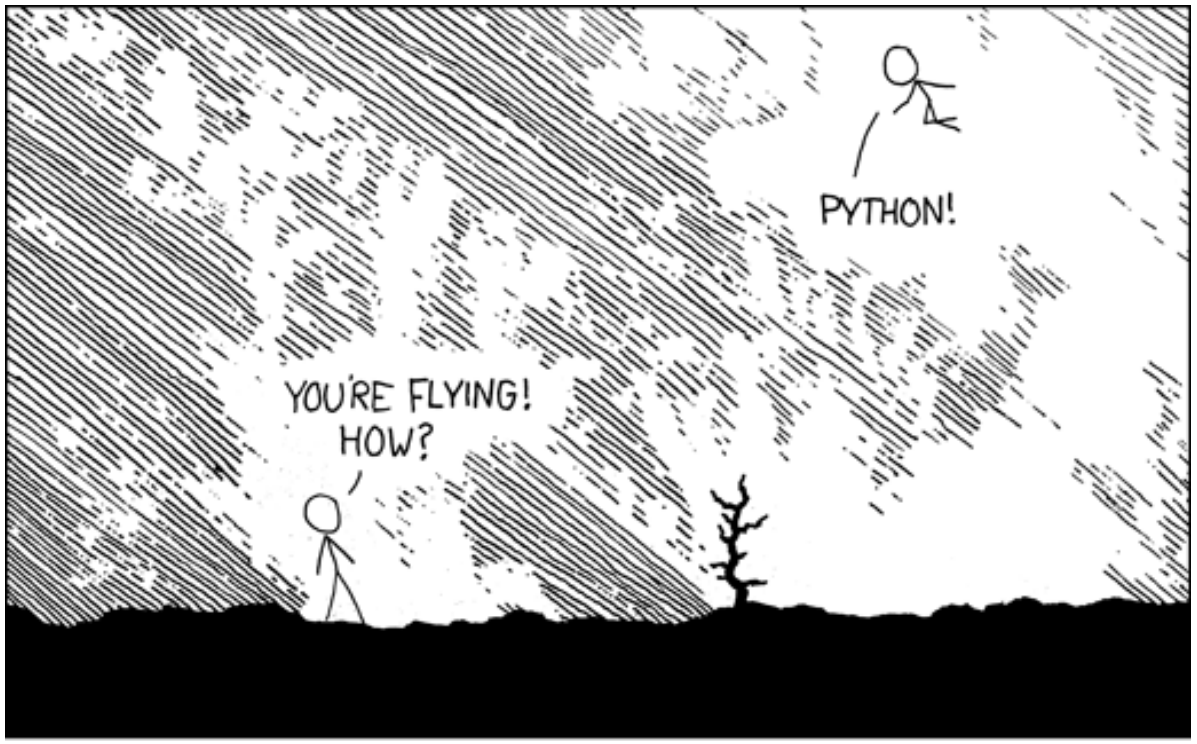
- The name: “Monty Python's Flying Circus”



Introduction to Python-2

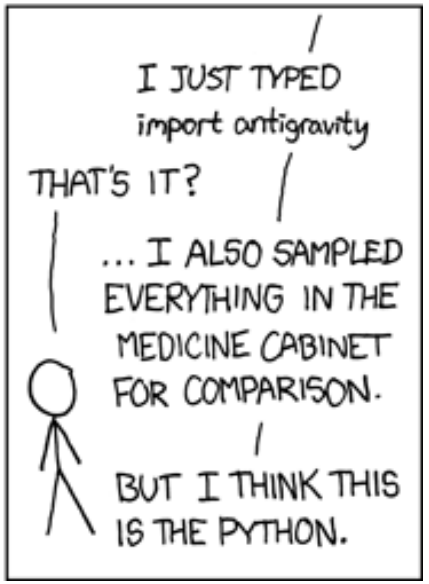
- Created by Guido Van Rossum in the early 1990s

- Python is a high-level, open-source programming language that can be used for a wide variety of programming tasks via an enormous support of freely available modules.


PYTHON!
YOU'RE FLYING! HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"


I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?


I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Introduction to Python-3

Who uses Python?

Everybody including:

- Coders of Linux components
- NASA (used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System)
- Google (to implement many components of its Web Crawler and Search Engine)
- Coders who want to write smaller and readable codes

Introduction to Python-4

How to get Python?

[Linux (!), Windows, Mac OS, etc.]

http://www.python.org/download/

Download Python 2.7.x instead of newer Python 3.x.x

(Some existing very useful third-party modules are not yet compatible with Python 3.x.x)

Introduction to Python-5

Windows:

http://code.google.com/p/pyscripter/
http://code.google.com/p/pythonxy/

Linux:

http://eric-ide.python-projects.org/

Introduction to Python-6

- Documentation, Getting Help:

  http://www.python.org/doc/
  http://www.tutorialspoint.com/python/
  http://www.istihza.com/ (in Turkish)
  http://www.python.org/community/lists/
  http://www.istihza.com/forum/ (in Turkish)

Introduction to Python-7

ONLINE INTERPRETER:

Python anywhere

Numeric data types

There are four distinct numeric types:

Integers

Long integers

Floating point numbers

Complex numbers have a real and imaginary part, which are each a floating point number. To extract these parts from a complex number *z*, use z.real and z.imag.

Arithmetic operators (a=10, b=20)

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0
**	Exponent - Performs exponential (power) calculation on operators	a**b will give 10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 is equal to 4 and 9.0//2.0 is equal to 4.0

Vectors and Matrices

- It is easy to define vectors and matrices in Python:

myvector=[1,2,3,4]

mymatrix = [[1, 2, 3, 0], [4, 5, 6, 0], [7, 8, 9, 0]]

The “hello world” program in C++

```
#include <iostream.h>
int main()
{
    cout << "Hello World!";
    return 0;
}
```

The “hello world” program and basic importing

Coding the “hello world” program is simple (one line):

```
print “hello world”
```

Try,

```
import antigravity
```

(needs internet connection)

Basic I/O operations-1

“print” command:

```
x=3
print x
print "your number is",x
```

Basic I/O operations-2

“input” command:

```
x=input("Enter x")
```

```
z = raw_input("What is your name?")
```

Assigning variables - 1

```
counter = 100 # An integer assignment
miles = 1000.0 # A floating point
name = "John" # A string
print counter
print miles
print name
```

Assigning variables - 2

- Multiple assignment:

```
a = b = c = 1
```

or

```
a, b, c = 1, 2.0, "john“
```

- You can also delete the reference to a number object by using the **del** statement:

```
del var
```

or

```
del var_a, var_b
```

A simple I/O example - 1

```
yourname = raw_input("What is your name?")

print "Hello",yourname
```

A simple I/O example – 2

```
x=input("Enter x")
y=input("Enter y")

z=x+y

print "The addition of",x,"and",y,"is",z,"."
```

A simple I/O example – 3

```
x=3

y=5

z1=x+y+3
print z1

z2=x+y+3.0
print z2

z3=x/y
print z3

z4=float(x)/y
print z4

z5=x/float(y)
print z5
```

A simple I/O example - 4

```
import datetime

now = datetime.datetime.now()

currentyear=now.year

myname=raw_input("What is your name? ")
mybirth=input("What is your year of birth? ")
yearsold=currentyear-mybirth

print "Dear",myname,", your age is",yearsold
```

Reserved words - 1

And	Exec	Not
Assert	Finally	Or
Break	For	Pass
Class	From	Print
Continue	Global	Raise
Def	if	Return
Del	import	Try
Elif	in	While
Else	is	With
Except	lambda	Yield

Reserved words - 2

```
# input is not a reserved word

input=input("Enter input")

print input
```

Mathematical functions - 1

Function	Returns (description)
abs(x)	The absolute value of x: the (positive) distance between x and zero.
ceil(x)	The ceiling of x: the smallest integer not less than x
cmp(x, y)	-1 if x < y, 0 if x == y, or 1 if x > y
exp(x)	The exponential of x: e^x
fabs(x)	The absolute value of x.
floor(x)	The floor of x: the largest integer not greater than x
log(x)	The natural logarithm of x, for x> 0
log10(x)	The base-10 logarithm of x for x> 0 .

Mathematical functions - 2

max(x1, x2,...)	The largest of its arguments: the value closest to positive infinity
min(x1, x2,...)	The smallest of its arguments: the value closest to negative infinity
modf(x)	The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float.
pow(x, y)	The value of x**y.
round(x [,n])	**x** rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0.
sqrt(x)	The square root of x for x > 0

Trigonometric functions

Function	Description
acos(x)	Return the arc cosine of x, in radians.
asin(x)	Return the arc sine of x, in radians.
atan(x)	Return the arc tangent of x, in radians.
atan2(y, x)	Return atan(y / x), in radians.
cos(x)	Return the cosine of x radians.
hypot(x, y)	Return the Euclidean norm, sqrt(x*x + y*y).
sin(x)	Return the sine of x radians.
tan(x)	Return the tangent of x radians.
degrees(x)	Converts angle x from radians to degrees.
radians(x)	Converts angle x from degrees to radians.

Multiline statements

- line continuation character (\):

```
total = item_one + \
        item_two + \
        item_three
```

- Statements contained within the [], {}, or () brackets do not need to use the line continuation character.

```
days = ['Monday', 'Tuesday', 'Wednesday',
         'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes can be used to span the string across multiple lines. For example, all the following are legal:

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""
```

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line

x=3; y=5; z1=x+y+3; print z1

Python Strings

Strings in Python are identified as a contiguous set of characters in between quotation marks.

The plus (+) sign is the string concatenation operator, and the asterisk (*) is the repetition operator.

```
str = 'Hello World!'

print str # Prints complete string
print str[0] # Prints first character of the string
print str[2:5] # Prints characters starting from 3rd to 5th
print str[2:] # Prints string starting from 3rd character
print str * 2 # Prints string two times
print str + "TEST" # Prints concatenated string
```

Python Lists

A list contains items separated by commas and enclosed within square brackets ([]).

All the items belonging to a list can be of different data type.

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list # Prints complete list
print list[0] # Prints first element of the list
print list[1:3] # Prints elements starting from 2nd till 3rd
print list[2:] # Prints elements starting from 3rd element
print tinylist * 2 # Prints tinylist two times
print list + tinylist # Prints concatenated lists
```

Python Tuples - 1

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]), and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as **read-only** lists.

Python Tuples - 2

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tinytuple = (123, 'john')

print tuple # Prints complete tuple
print tuple[0] # Prints first element of the tuple
print tuple[1:3] # Prints elements starting from 2nd till 3rd
print tuple[2:] # Prints elements starting from 3rd element
print tinytuple * 2 # Prints tinytuple two times
print tuple + tinytuple # Prints concatenated tuples
```

Python Dictionary - 1

- Python's dictionaries are kind of hash table type.
- They consist of key-value pairs.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Python Dictionary - 2

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print dict['one'] # Prints value for 'one' key
print dict[2] # Prints value for 2 key
print tinydict # Prints complete dictionary
print tinydict.keys() # Prints all the keys print
tinydict.values() # Prints all the values
```