

# Python Tutorial

## Part I

Greg (mastergreg) Liras and John (nemo) Giannelos

foss.ntua

December 17, 2011

# Outline

## Introduction to Python

What is Python?

Features

Why Python?

Dos and Don'ts

## Python Standard Types

Arithmetic

Strings

Data Structures



# What is Python?

*Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*
- ▶ is *Dynamic*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*
- ▶ is *Dynamic*
- ▶ is *Portable*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*
- ▶ is *Dynamic*
- ▶ is *Portable*
- ▶ is *Object Oriented*



# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*
- ▶ is *Dynamic*
- ▶ is *Portable*
- ▶ is *Object Oriented*
- ▶ has *Vast Libraries*

# Python Tutorial

In a few words, Python,

- ▶ is is *Scripting Language*
- ▶ is *Strongly Typed*
- ▶ is *Dynamic*
- ▶ is *Portable*
- ▶ is *Object Oriented*
- ▶ has *Vast Libraries*
- ▶ is *Simple and non-obtrusive*

# Why?

- ▶ It is easy to remember

# Why?

- ▶ It is easy to remember
- ▶ You can develop rapidly

# Why?

- ▶ It is easy to remember
- ▶ You can develop rapidly
- ▶ Interface with C libraries

# Must and Must Not

- ▶ Search first code less

# Must and Must Not

- ▶ Search first code less
- ▶ Import only what you need

# Must and Must Not

- ▶ Search first code less
- ▶ Import only what you need
- ▶ Run pychecker on your code



# Outline

## Introduction to Python

What is Python?

Features

Why Python?

Dos and Don'ts

## Python Standard Types

Arithmetic

Strings

Data Structures

# Numeric types

- ▶ int (up to  $10^{308}$ !!!!)

# Numeric types

- ▶ int (up to  $10^{308}$ !!!!)
- ▶ float (53 bits precision)

# Numeric types

- ▶ int (up to  $10^{308}$ !!!!)
- ▶ float (53 bits precision)
- ▶ complex ( $1 + 2j$ )

# Operators

► + (add)

# Operators

- ▶ + (add)
- ▶ - (subtract)

# Operators

- ▶ + (add)
- ▶ - (subtract)
- ▶ \* (multiply)

# Operators

- ▶ + (add)
- ▶ - (subtract)
- ▶ \* (multiply)
- ▶ / (divide)





# Operators

- ▶ + (add)
- ▶ - (subtract)
- ▶ \* (multiply)
- ▶ / (divide)
- ▶ % (modulo)

# Operators

- ▶ + (add)
- ▶ - (subtract)
- ▶ \* (multiply)
- ▶ / (divide)
- ▶ % (modulo)
- ▶ = (assign)

# Strings

- ▶ Strings are not lists! Strings are immutable!

# Strings

- ▶ Strings are not lists! Strings are immutable!
- ▶ Simple concatenation:

```
>>> 'Hello' + 'World'  
'HelloWorld'
```

# Strings

- ▶ Strings are not lists! Strings are immutable!

- ▶ Simple concatenation:

```
>>> 'Hello' + 'World'  
'HelloWorld'
```

- ▶ Slicing:



# Strings

- ▶ Strings are not lists! Strings are immutable!

- ▶ Simple concatenation:

```
>>> 'Hello' + 'World'
'HelloWorld'
```

- ▶ Slicing:

```
▶ >>> 'HelloWorld'[0]
'H'
```



# Strings

- ▶ Strings are not lists! Strings are immutable!

- ▶ Simple concatenation:

```
>>> 'Hello' + 'World'
'HelloWorld'
```

- ▶ Slicing:

```
▶ >>> 'HelloWorld'[0]
'H'
```

```
▶ >>> 'HelloWorld'[6:]
'orld'
```



# Strings

- ▶ Strings are not lists! Strings are immutable!

- ▶ Simple concatenation:

```
>>> 'Hello' + 'World'
'HelloWorld'
```

- ▶ Slicing:

```
▶ >>> 'HelloWorld'[0]
'H'
```

```
▶ >>> 'HelloWorld'[6:]
'orld'
```

- ▶ Unicode Strings:

```
>>> ur'Hello\u0020World !'
u'Hello World !'
```



# Lists

```
▶ >>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
```



# Lists

```
▶ >>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
```

▶ Negative indices:

```
>>> a[-2]
100
```



# Lists

► `>>> a = ['spam', 'eggs', 100, 1234]`

`>>> a`

`['spam', 'eggs', 100, 1234]`

► Negative indices:

`>>> a[-2]`

`100`

► Concatenation:

`>>> a[:2] + ['bacon', 2*2]`

`['spam', 'eggs', 'bacon', 4]`



# Lists

```
▶ >>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
```

▶ Negative indices:

```
>>> a[-2]
100
```

▶ Concatenation:

```
>>> a[:2] + ['bacon', 2*2]
['spam', 'eggs', 'bacon', 4]
```

▶ Comprehension:

```
for i in a:
    print i
```

# Tuples

- ▶ Immutable (just as strings)

# Tuples

- ▶ Immutable (just as strings)
- ▶ Indexed

# Tuples

- ▶ Immutable (just as strings)
- ▶ Indexed
- ▶ Nested

# Sets

*A set is an unordered collection with no duplicate elements.*



# Sets

*A set is an unordered collection with no duplicate elements.*

```
▶ >>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```



# Sets

*A set is an unordered collection with no duplicate elements.*

- ▶ 

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```
- ▶ Operators:

# Sets

*A set is an unordered collection with no duplicate elements.*

- ▶ 

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```
- ▶ Operators:
  - ▶  $a - b$  (in  $a$  but not in  $b$ )



# Sets

*A set is an unordered collection with no duplicate elements.*

```
▶ >>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```

▶ Operators:

- ▶  $a - b$  (in  $a$  but not in  $b$ )
- ▶  $a | b$  (in  $a$  or in  $b$ )



# Sets

*A set is an unordered collection with no duplicate elements.*

```
▶ >>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```

▶ Operators:

- ▶  $a - b$  (in  $a$  but not in  $b$ )
- ▶  $a | b$  (in  $a$  or in  $b$ )
- ▶  $a \& b$  (in  $a$  and in  $b$ )

# Sets

*A set is an unordered collection with no duplicate elements.*

```
▶ >>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> set(basket)  
set(['orange', 'pear', 'apple', 'banana'])
```

▶ Operators:

- ▶  $a - b$  (in  $a$  but not in  $b$ )
- ▶  $a | b$  (in  $a$  or in  $b$ )
- ▶  $a \& b$  (in  $a$  and in  $b$ )
- ▶  $a \wedge b$  (in  $a$  or  $b$  but not in both)

