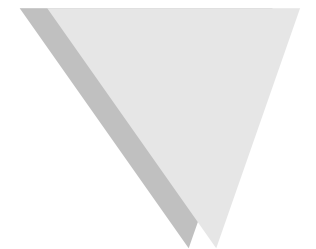
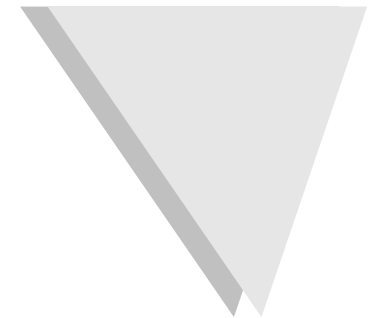
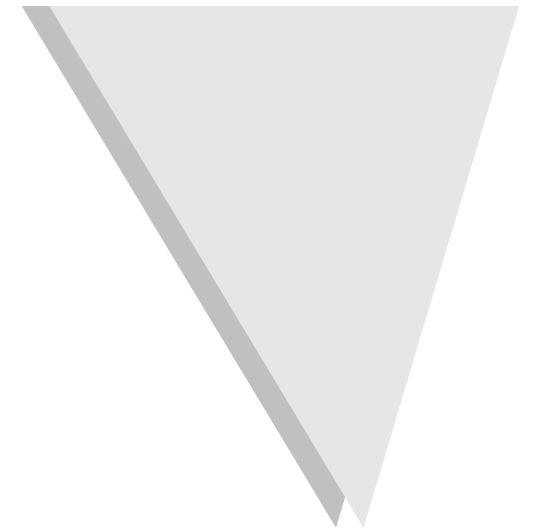


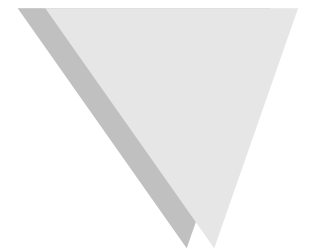
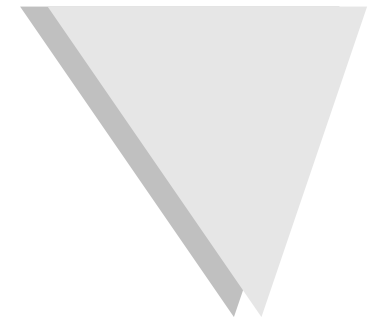
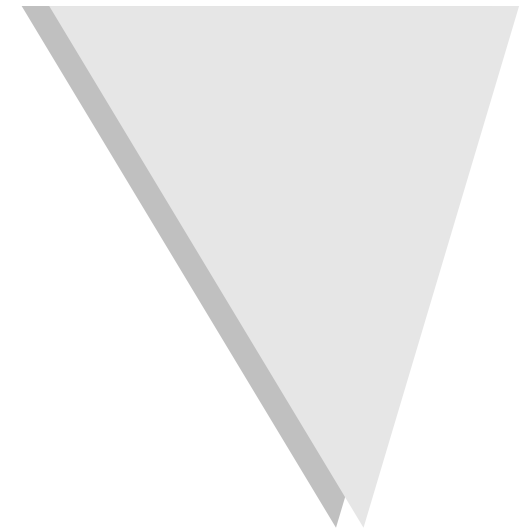
Python for AI

- Introducing Python
- Applications & Platforms
- Comparing to LISP
- Further Links



Introducing Python

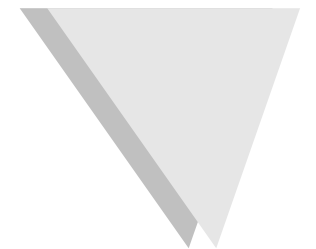
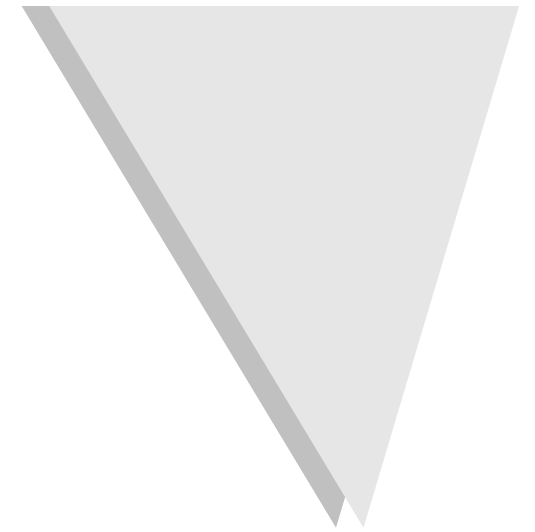
- object-oriented:
 - everything is an object
 - loosely defined interfaces
- dynamically typed:
 - variables have no type – data is typed
 - allows natural polymorphism
- whitespace for indentation:
 - no use of block delimiters
 - readable pseudo code



Introducing Python

Function example:

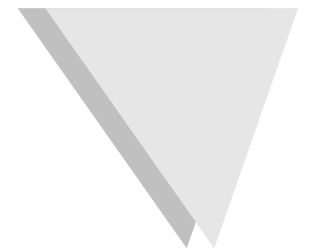
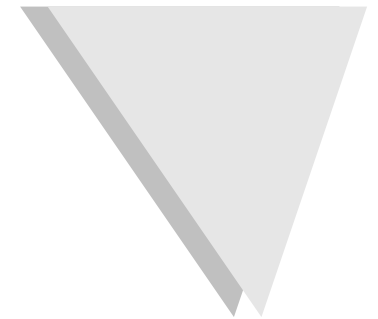
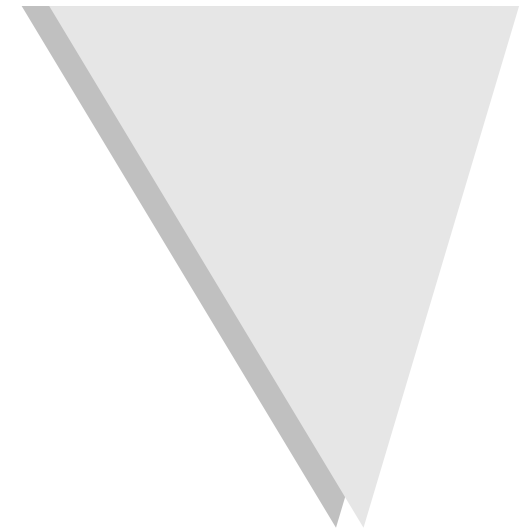
```
def gcd(a, b):    # greatest common divisor
    while b:     # i.e. b > 0 for positives integers
        a, b = b, a % b    # parallel assignment
    return a
```



Introducing Python

Class example:

```
class stack: # simple stack
    def __init__(self):
        self.data = [] # empty list
    def push(self, item):
        self.data.append(item)
    def pop(self):
        return self.data.pop()
```



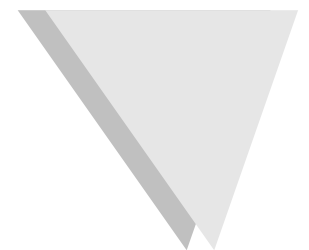
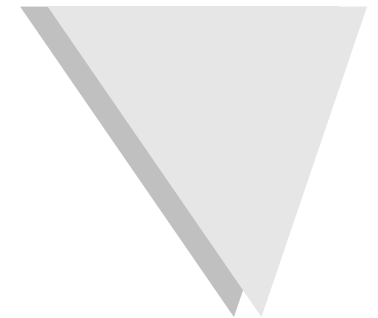
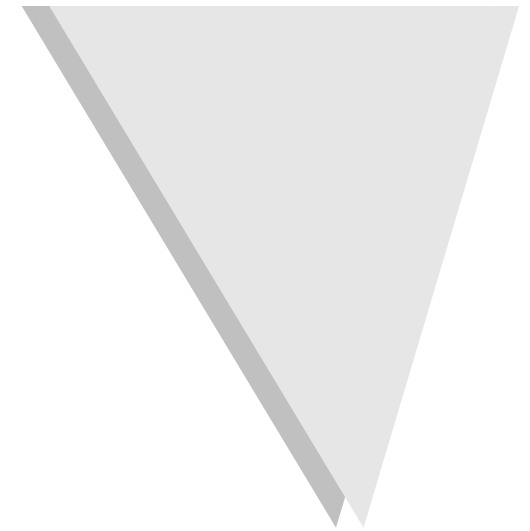
Introducing Python

Data types example:

```
map_to = {0:("a", 42), 1:("b", 0815)} # hash table
for key in map_to.keys(): # iterate through list
    print map_to[key]
print map(lambda x:map_to[x], map_to.keys())
```

Applications & Platforms

- existing applications:
 - ZOPE (open source web–publishing system)
 - mailman (GNU mailing list server)
 - linux installation tool (RedHat distribution)
 - lots of numerical and other scientific code
- applications under development:
 - linux kernel configuration (by Eric S. Raymond)
 - Language for PC–BIOS replacement (Intel)

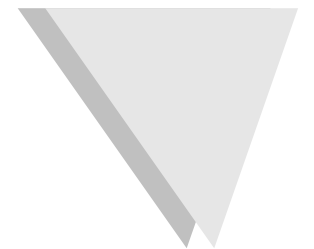
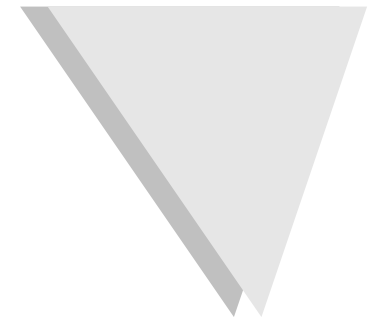
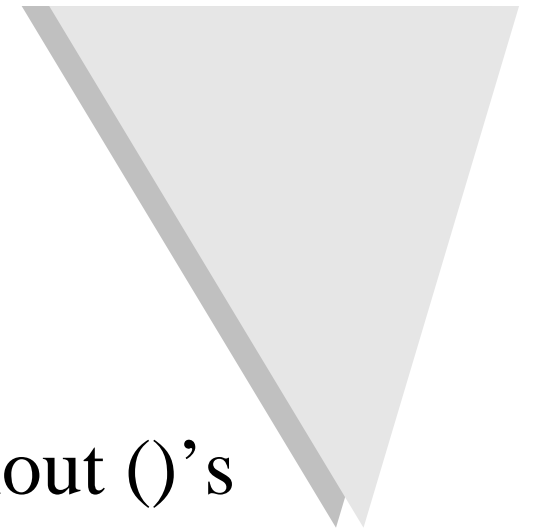


Applications & Platforms

- supported platforms:
 - Unix, Windows, Mac, BeOS, DOS, OS/2, VMS, Cray ...
- existing implementations:
 - CPython (written in C, bindings to C/C++, Fortran etc.)
 - JPython (written in Java, compiles to Java byte code)
- implementations under development:
 - Python# (part of the Microsoft .NET project)

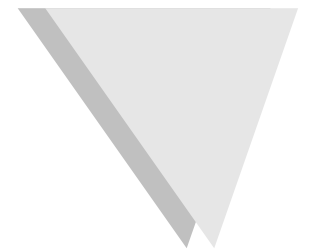
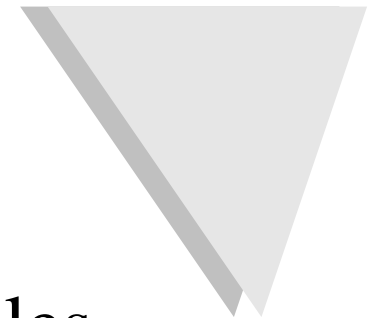
Comparing to LISP

- Python can be seen as a LISP dialect without ()'s
- syntax complexity trade-off:
 - harder to parse for computers
 - easier to read (and write) for humans
- uses infix notation instead of prefix notation:
 - e.g.: $7 + 5$ instead of $(+ 7 5)$
- syntactic sugar for list operations:
 - e.g. indexing and slicing for lists



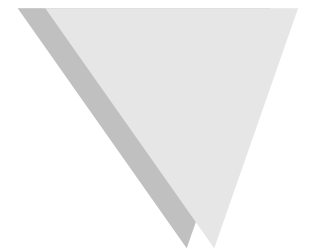
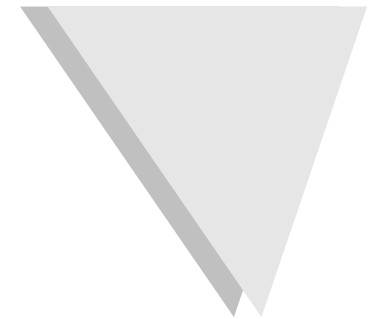
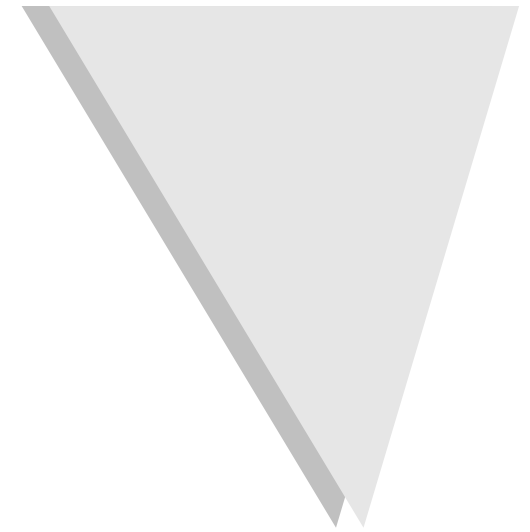
Comparing to LISP

- distinguishes between statements and expressions
- availability of map/filter/reduce
- lambda only for expressions
- lists are resizable arrays $\Rightarrow O(1)$ for access
- explicit self to reference object member variables
- explicit return for returning function values
- 1 namespace for functions and variables
- all namespaces are normal hash tables



Comparing to LISP

- Python's advantages compared to LISP:
 - higher readability
 - more expressive syntax
 - easier to learn
- Python's disadvantages:
 - typically slower execution speed
 - more complex syntax



Further Links

- Documentation at the Python language web-site:
 - <http://www.python.org/doc>
- Python for Lisp Programmers (by Peter Norvig)
 - <http://www.norvig.com/python.-lisp.html>
- Python community mailing list:
 - python-list@python.org
- Python development at SourceForge:
 - <http://python.sourceforge.net>