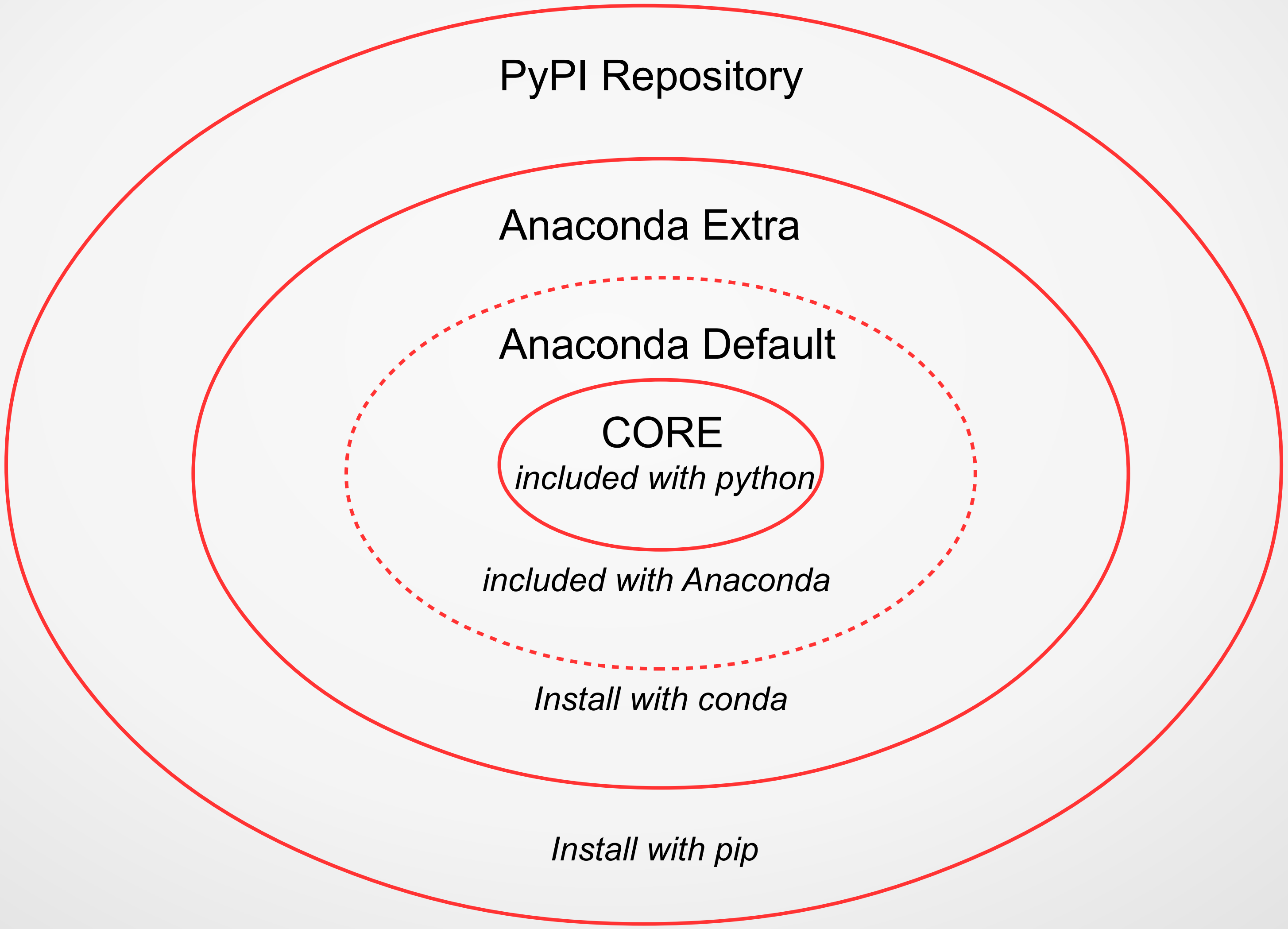
Welcome to Intro to Python @Esri

- Instructor: John Strickler (jstrickler@gmail.com)
- Class time: 8:30 AM - 5:30 PM
- Please make a name tent (tents and Sharpies up front)
- Lunch 12 noon – 1 PM
- WI-FI "esri2018" p/w "2018"
- Requirements
  - Python 3
    - www.anaconda.com
    - www.python.org
  - l**PyCharm Community Edition** *Or your favorite IDE*
    - www.jetbrains.com/pycharm
  - Student files
    - Windows: py3esri_1.0.zip
    - Mac/Linux: py3esri_1.0.tar.tz

Python Modules (using Anaconda)



What Can Python Do?

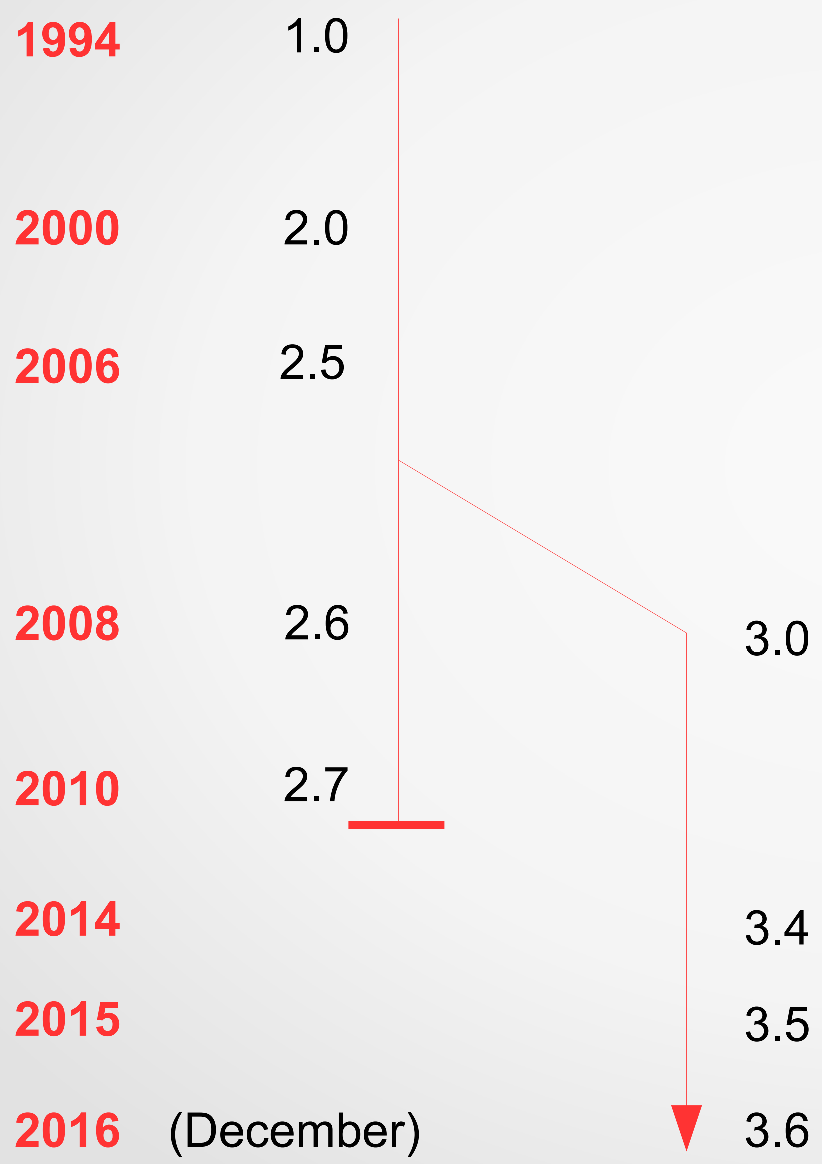
- Web apps
- Web services (REST, SOAP)
- Data mining/web scraping
- Data science
- End-user GUI apps
- System Administration (Windows, Mac, Linux)
- Scientific/Engineering analysis
- Data visualization
- Cloud apps

Advantages of Python

- Readable
- Multi-paradigm
- Modular
- Exceptions
- Standard library
- Extensible and embeddable

Disadvantages of Python

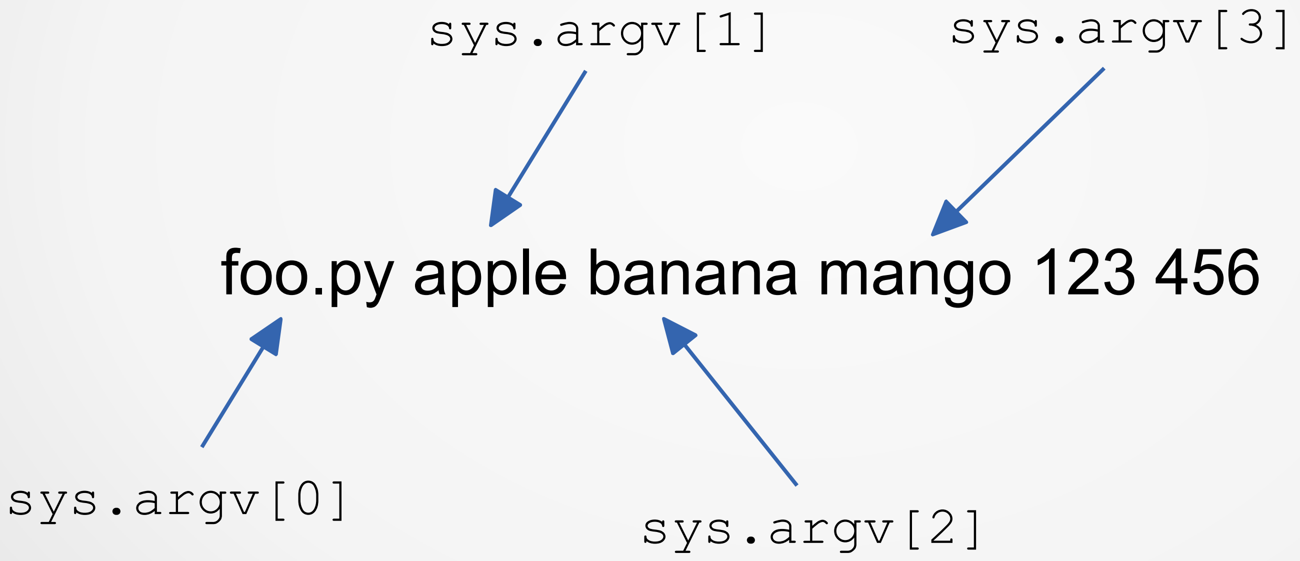
Python Evolution



String literals

- Single-delimited
  - 'spam\n' "spam\n"
- Triple-delimited
  - '''spam\n''' """spam\n"""
- Raw
  - r'spam\n'

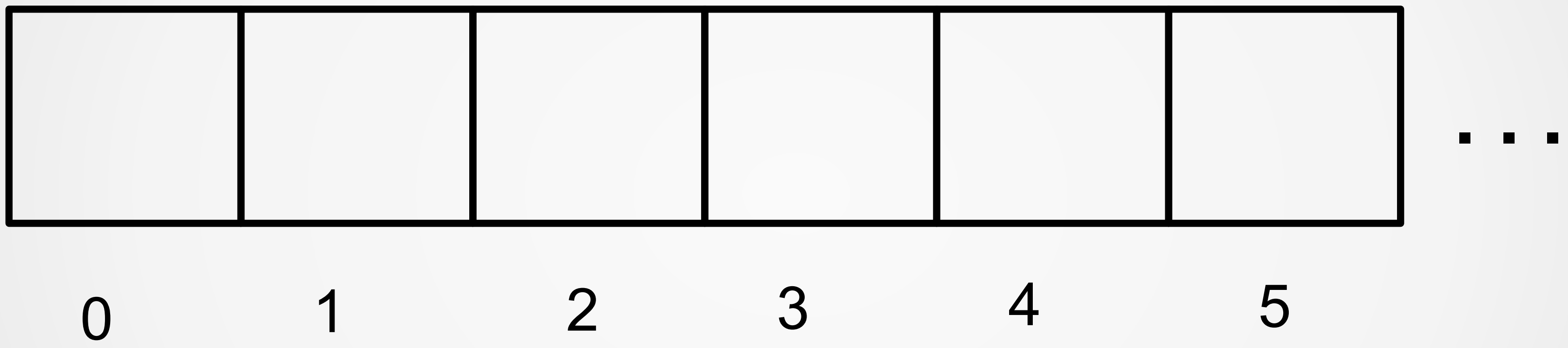
Command Line Parameters



Indenting blocks

```
Block statement:
●●●●Statement
●●●●Statement
●●●●Nested Block Statement:
●●●●●●●●Statement
●●●●●●●●Statement
●●●●Statement
●●●●Statement

Statement
```

Sequences



Slices

```
0 W 1 O 2 M 3 B 4 A 5 T 6
```

```
s = "WOMBAT"

s[0:3]        first 3 characters  "WOM"
s[:3]         same, using default start of 0 "WOM"
s[1:4]        s[1] through s[3] "OMB"
s[3:6]        s[3] through end  "BAT"
s[3:len(s)]   s[3] through end  "BAT"
s[3:]         s[3] through end, using default end "BAT"
```

Lists vs Tuples

Lists

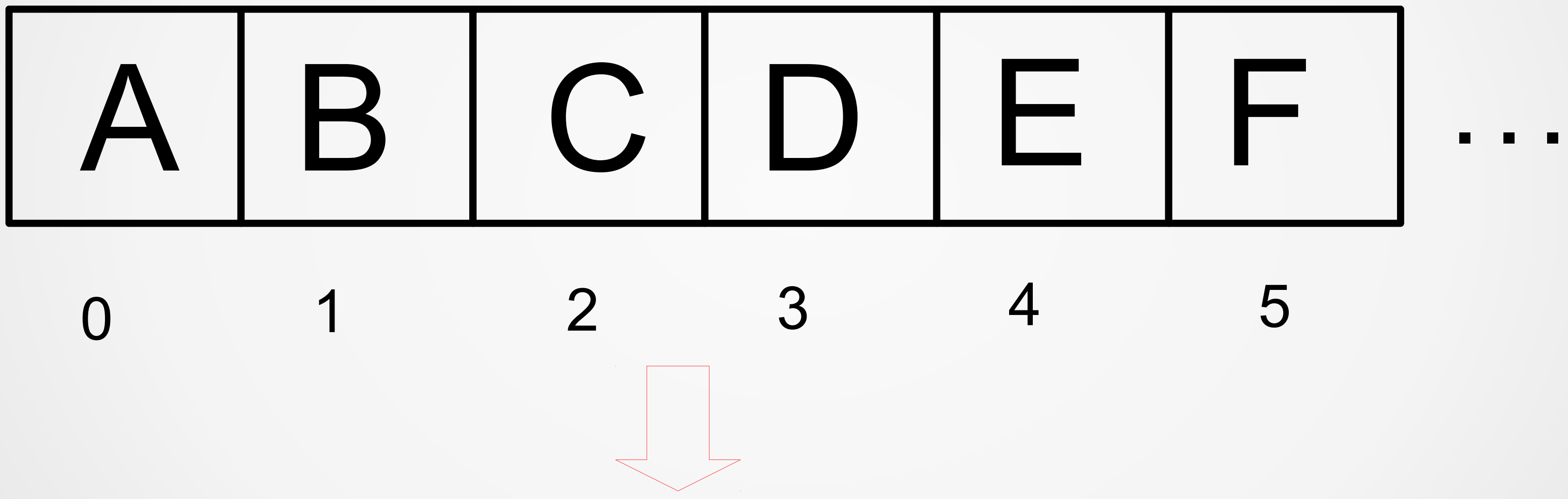
- Dynamic Array
- Mutable/unhashable
- Order doesn't matter
- Designed for looping
- Think "ARRAY"

Tuples

- Collection of related fields
- Immutable/hashable
- Order matters
- Designed for unpacking
- Think "STRUCT" or "RECORD"

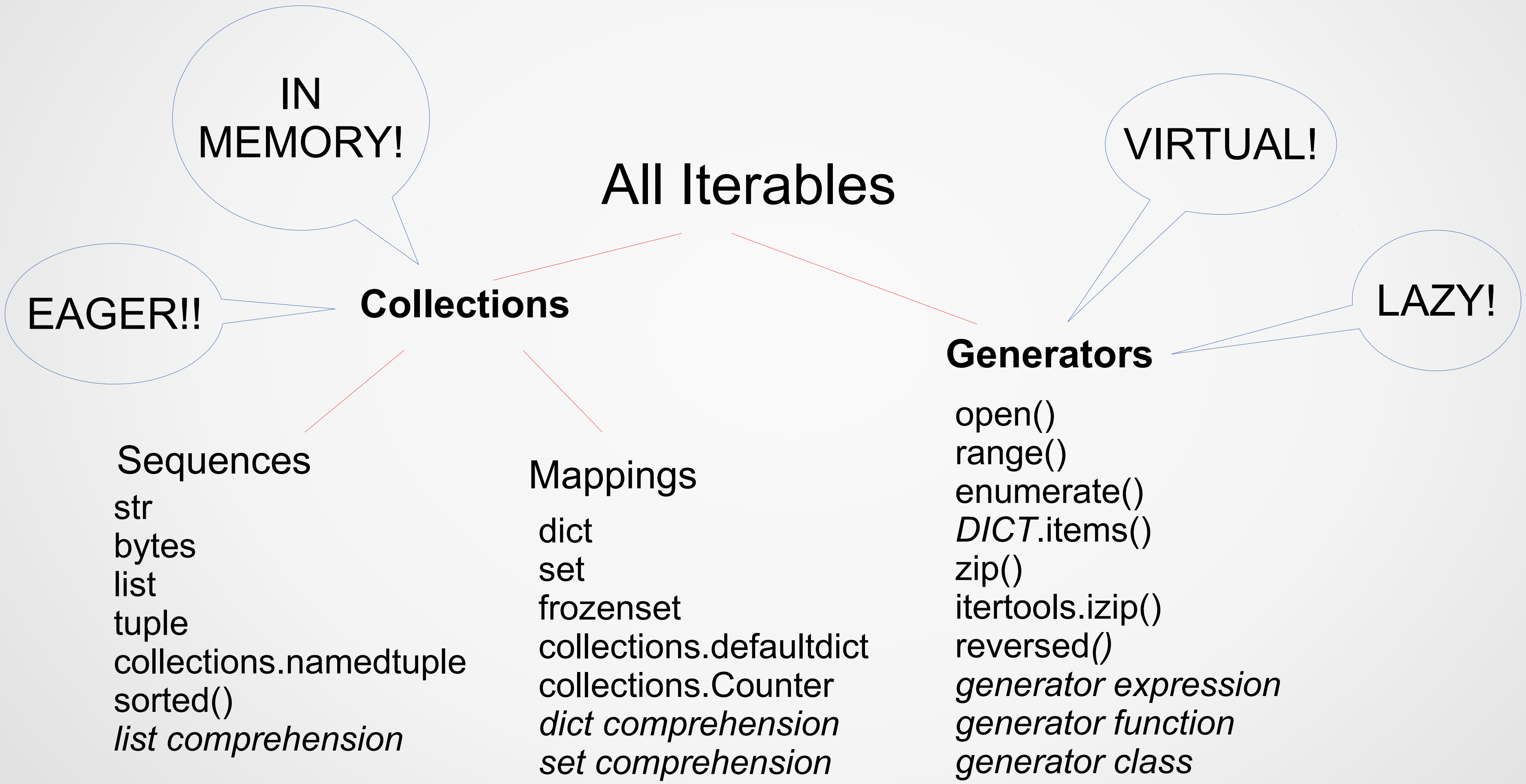
Myth #1: tuples are just read-only lists
Myth #2: tuples are faster than lists
Myth #3: tuples use less memory than lists (*slightly* true)

enumerate()



(0, A), (1, B), (2, C), (3, D), (4, E), (5, F)...

Iterables



All Iterables

Collections

Sequences

str
bytes
list
tuple
collections.namedtuple
sorted()
list comprehension

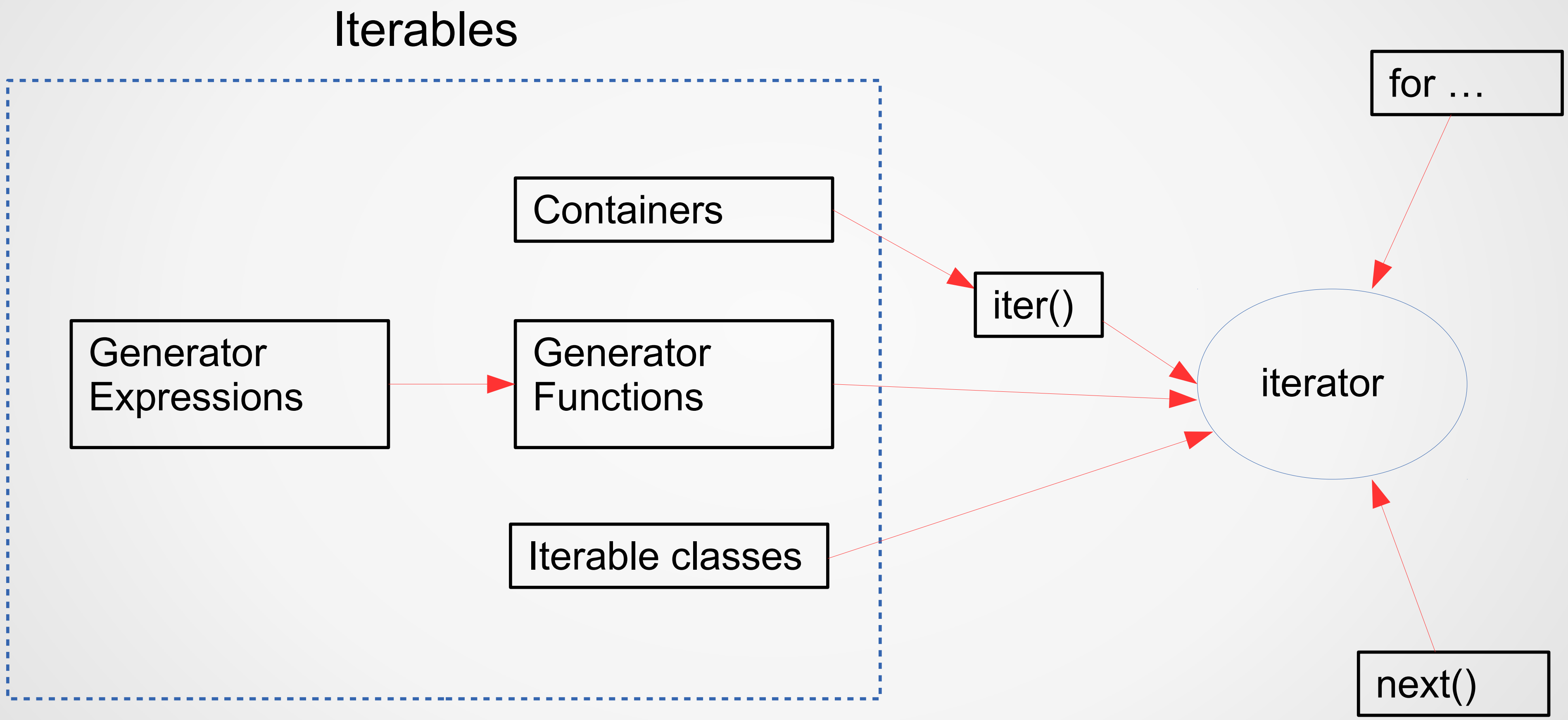
Mappings

dict
set
frozenset
collections.defaultdict
collections.Counter
dict comprehension
set comprehension

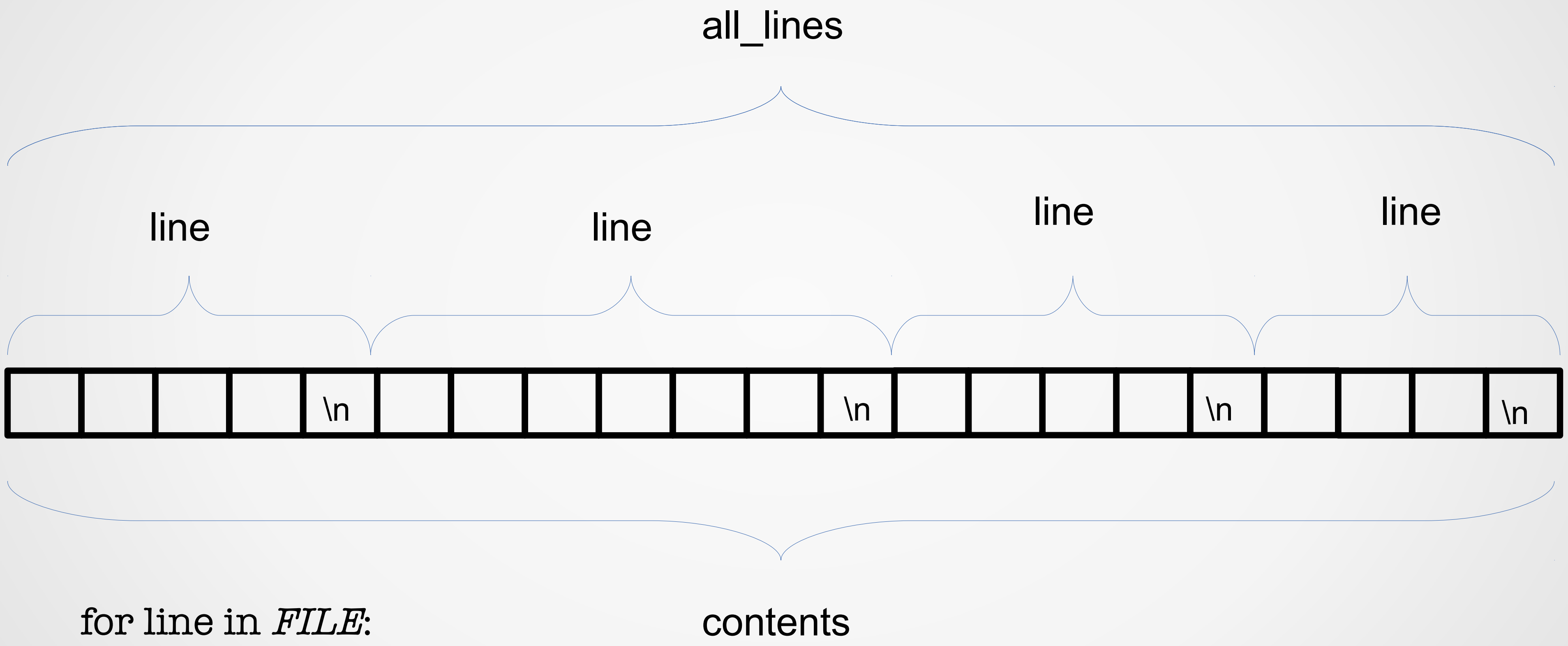
Generators

open()
range()
enumerate()
DICT.items()
zip()
itertools.izip()
reversed*()*
generator expression
generator function
generator class

Iterables and iterators



Reading text files



```
for line in FILE:
    pass
```

```
contents = FILE.read()
```

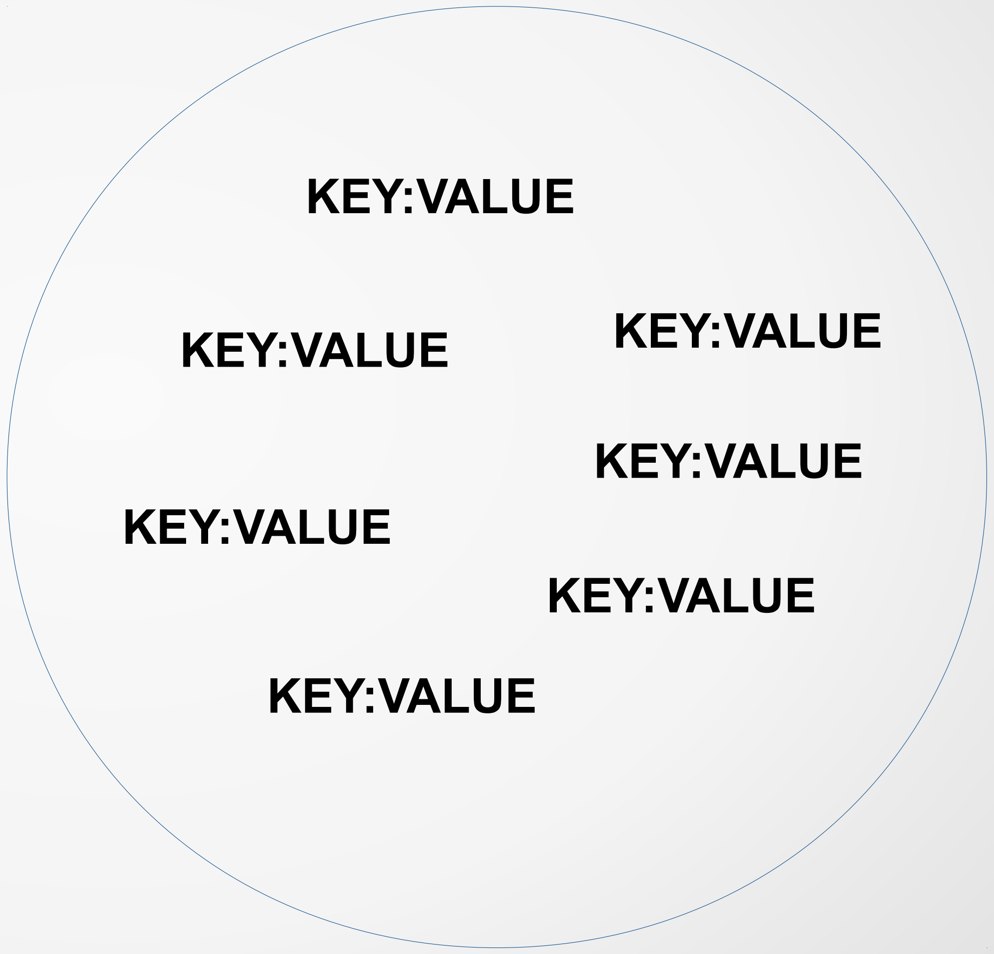
```
all_lines = FILE.readlines()
```

What do these words mean?

- formication
- ramiferous

Dictionary

- Key/value pairs
- Keys ordered (3.6+)
- Keys not ordered (<3.6)
- Keys are unique
- Use .items() to loop through k/v pairs

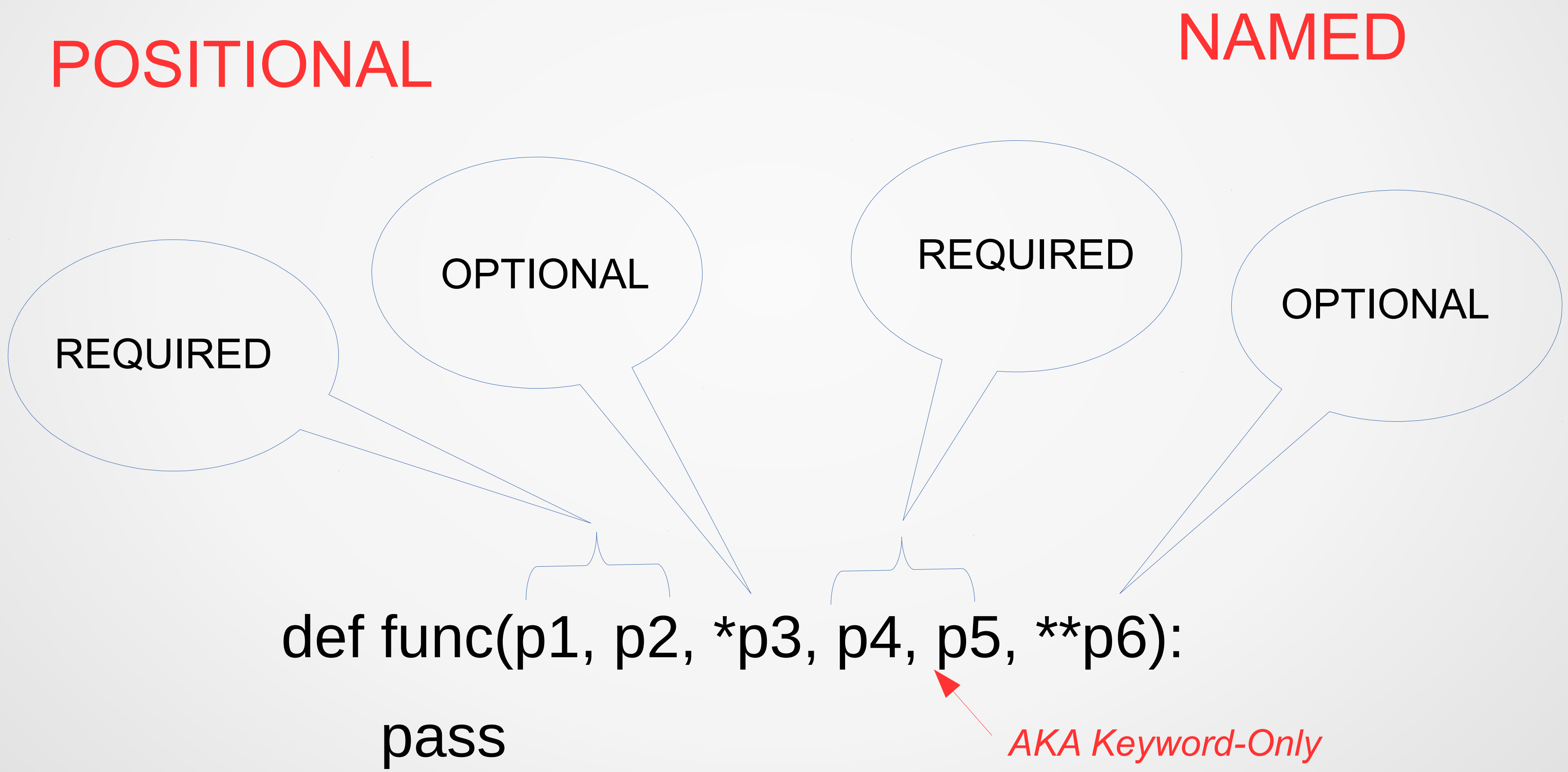


dict.items()

A	B	C	D	E	F	*keys*
100	200	300	400	500	600	*values*

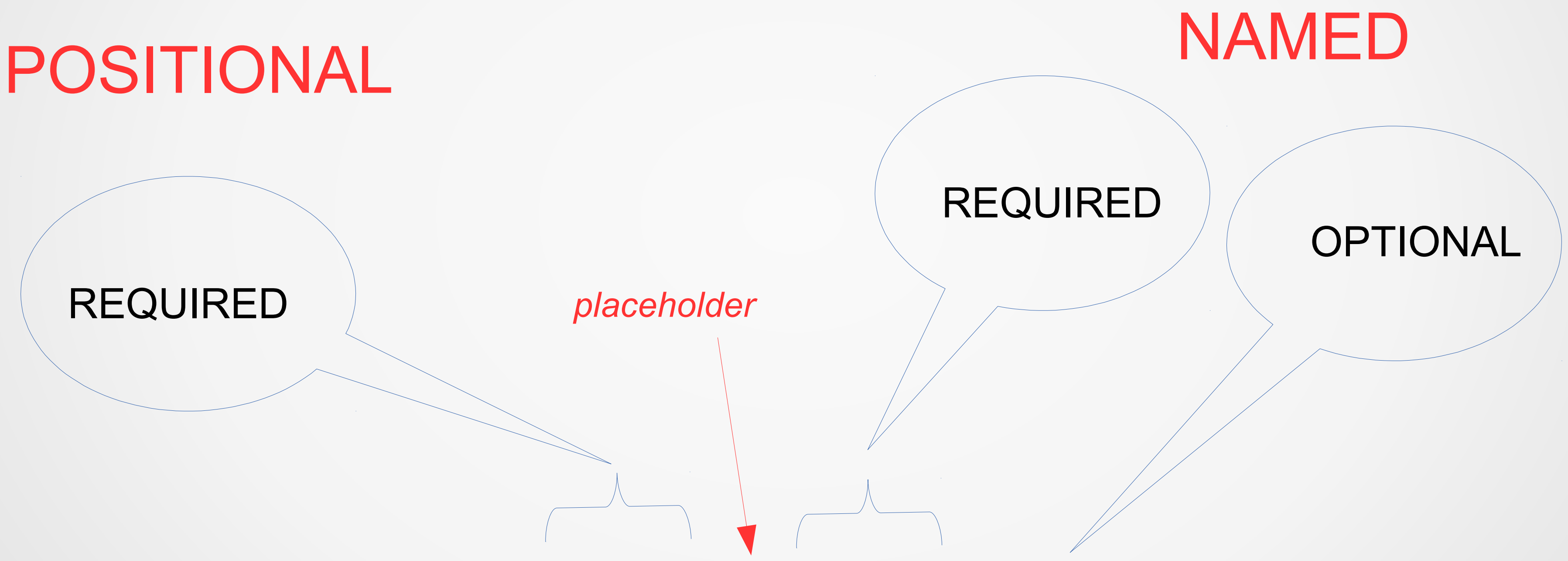
(A, 100), (B, 200), (C, 300), (D, 400), (E, 500), (F, 600) ...

Function parameters



```
def func(p1, p2, *p3, p4, p5, **p6):
    pass
```

Function parameters, cont'd



```
def func(p1, p2, *, p4, p5, **p6):
    pass
```

Parameter passing

~~Passing by reference~~

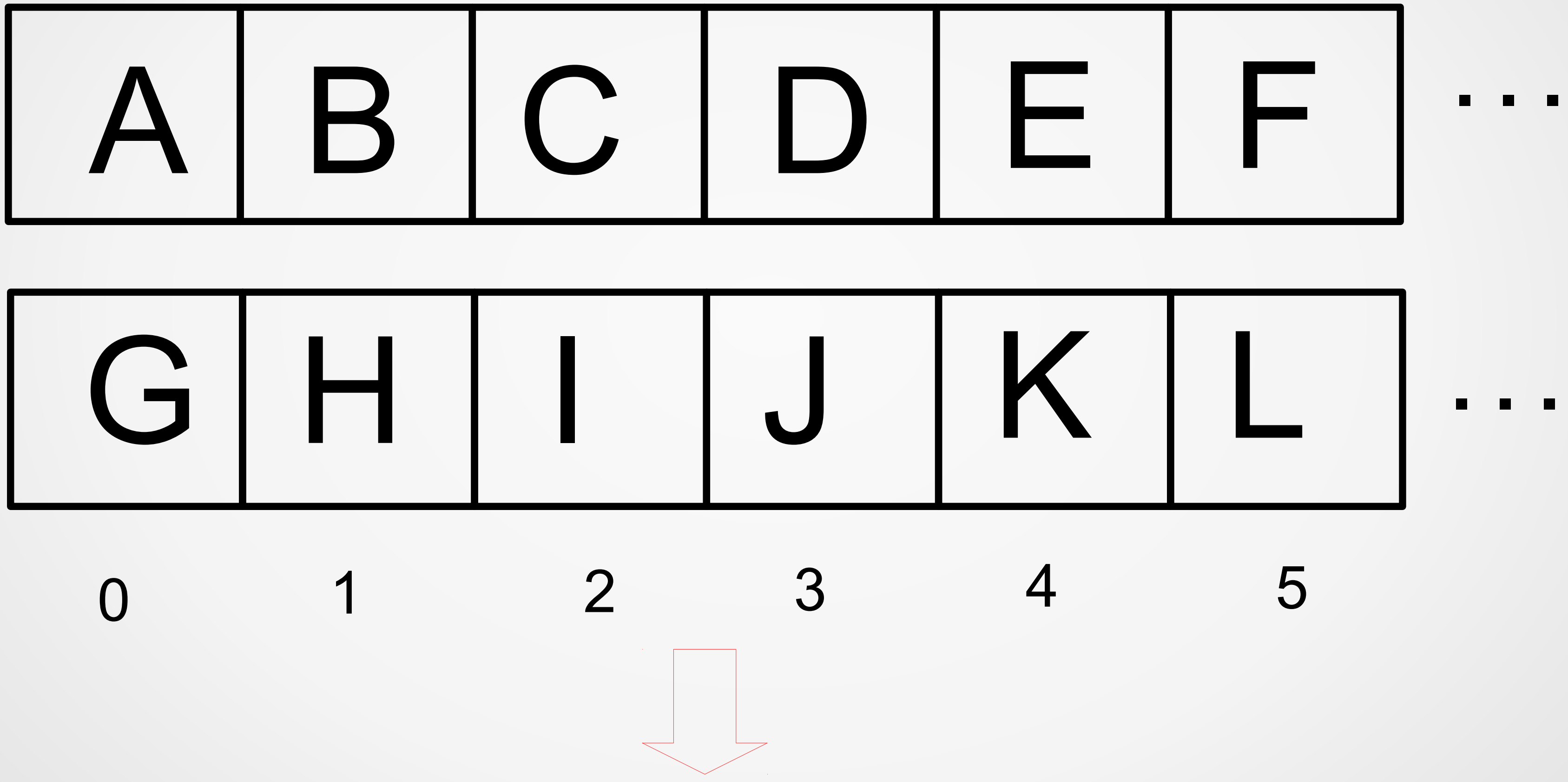
~~Passing by value~~

✓ Passing by sharing

- Read-only reference is passed
- Mutables may be changed via reference
- Immutables may not be changed

```
def spam(x, y):
    x = 5
    y.append('ham')

foo = 17
bar = ['toast', 'jam']

spam(foo, bar)
```

zip()



(A, G), (B,H), (C, I), (D, J), (E, K), (F, L)...

Sorting

- Numbers

  n, n, n, …

- Strings

  "$C_1C_2C_3$", "$C_1C_2C_3$", "$C_1C_2C_3$", …

- Iterables

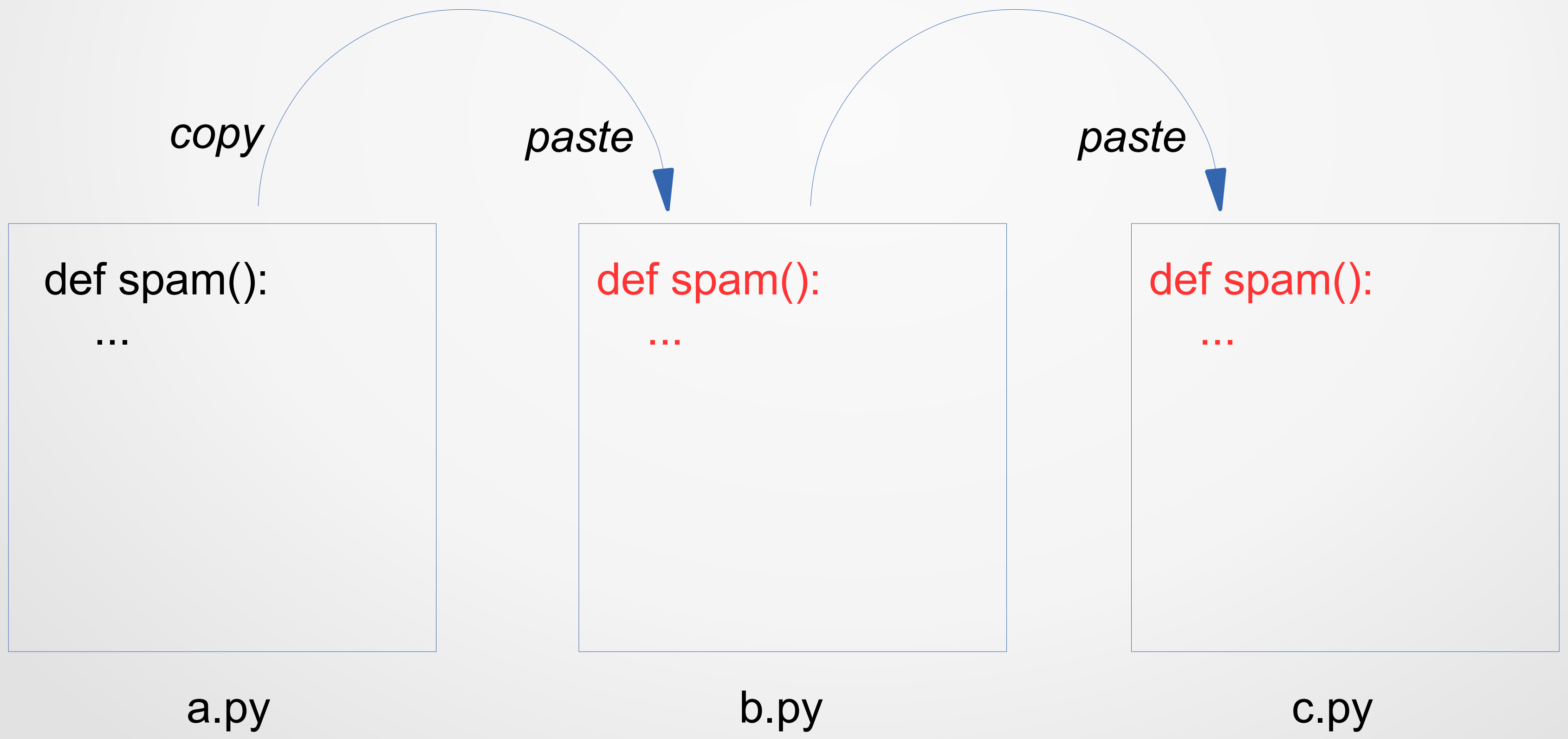
  [O_1, O_2, O_3], [O_1, O_2, O_3], [O_1, O_2, O_3], …

- ***dict***.items()

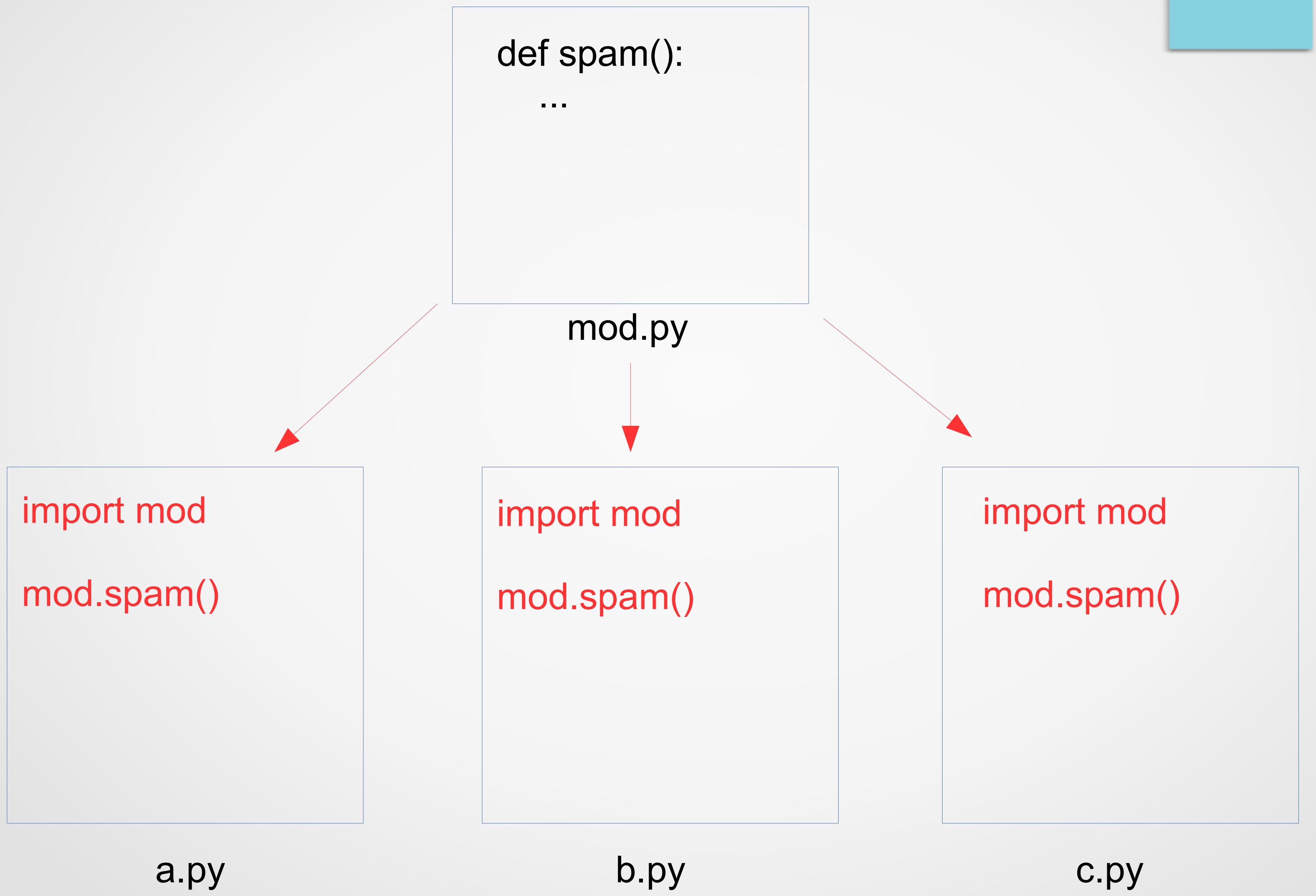
  (key, value), (key, value), (key, value), ...

Copying and pasting functions

DON'T DO THIS!!



Using a module



```
def spam():
    ...
```

mod.py

```
import mod

mod.spam()
```

a.py

```
import mod

mod.spam()
```

b.py

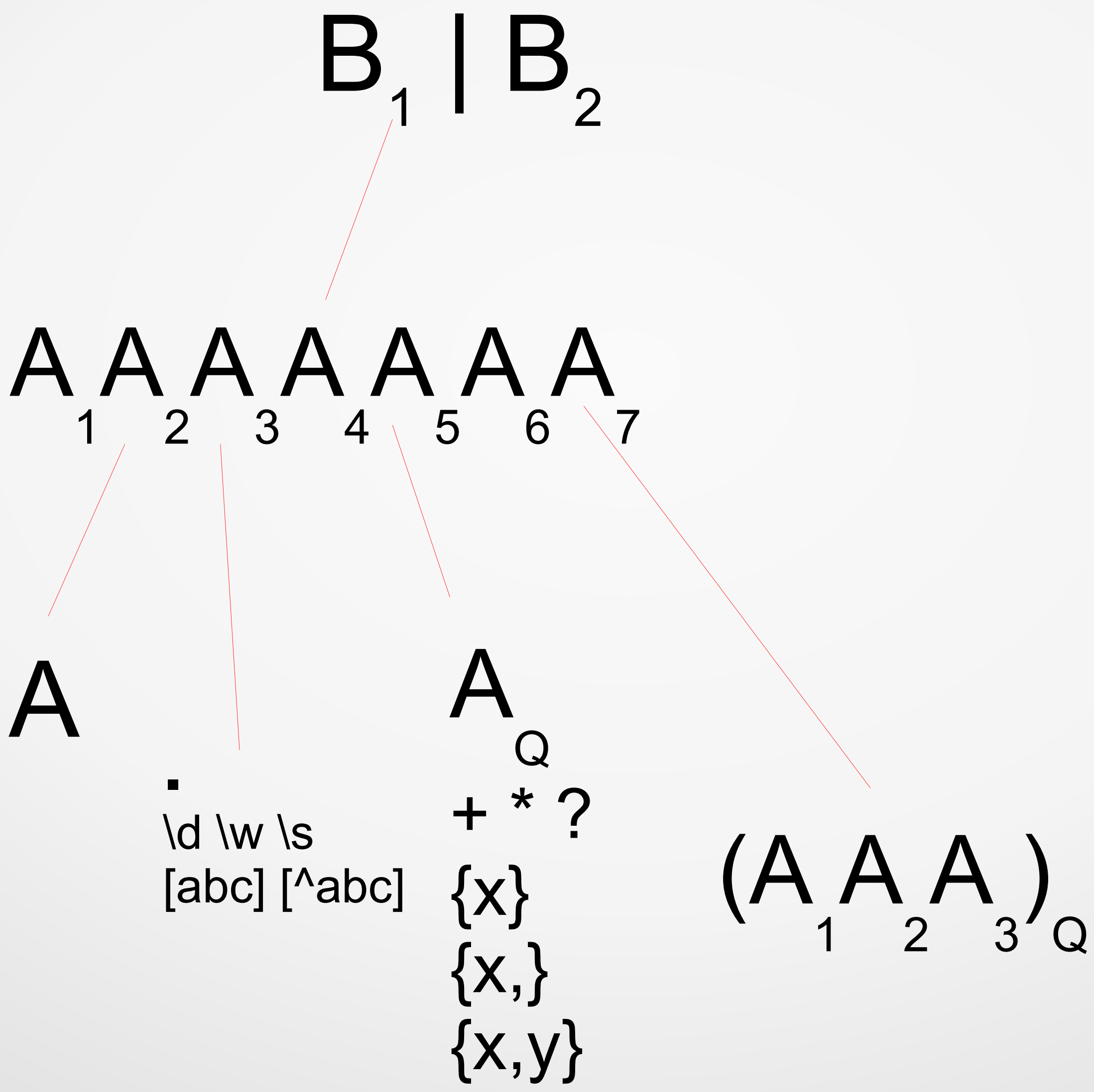
```
import mod

mod.spam()
```

c.py

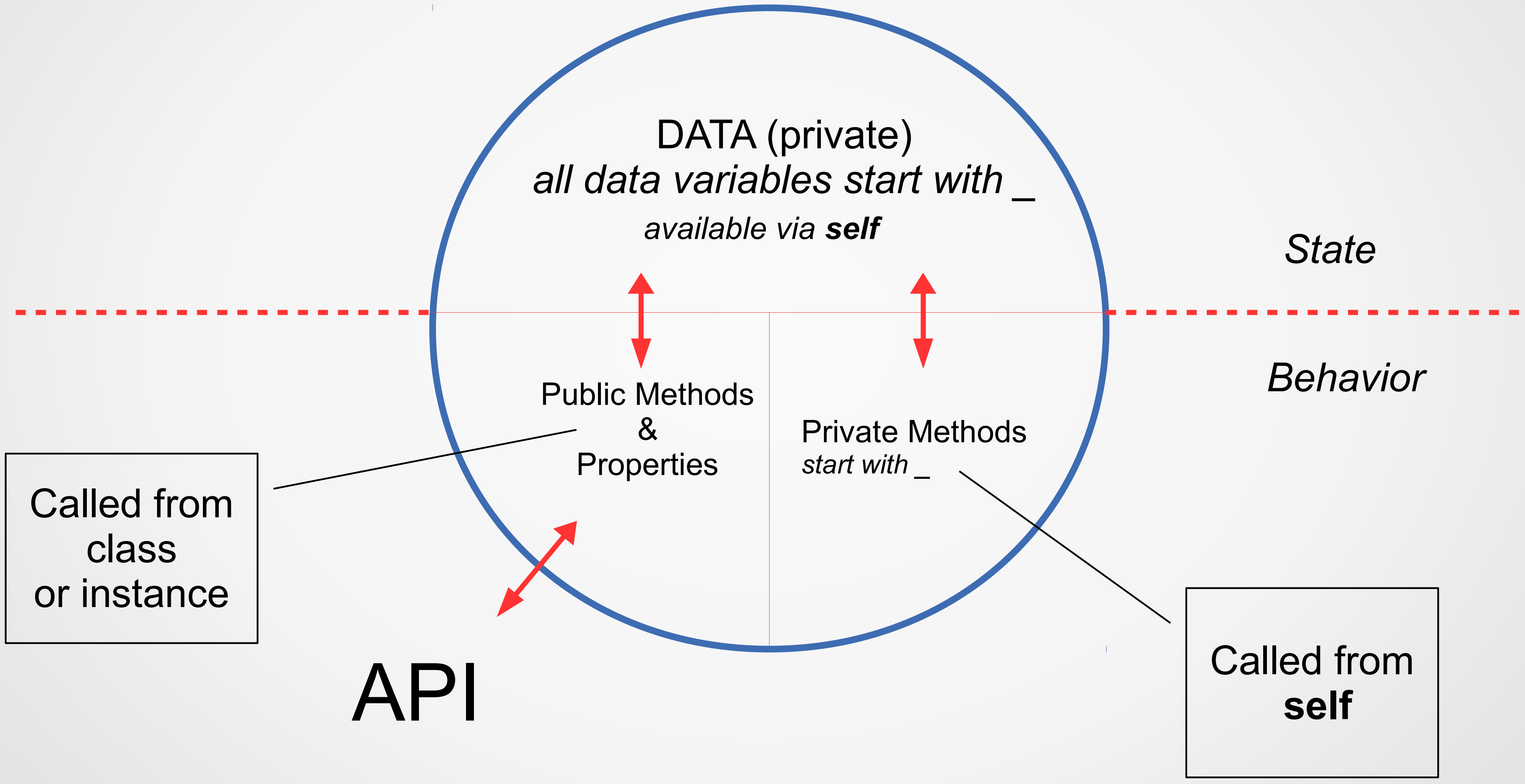
Regular expression tasks

- Search (is the match in the text?)
- Retrieve (get the matching text)
- Replace (substitute new text for match)
- Split (get what *didn't* match)

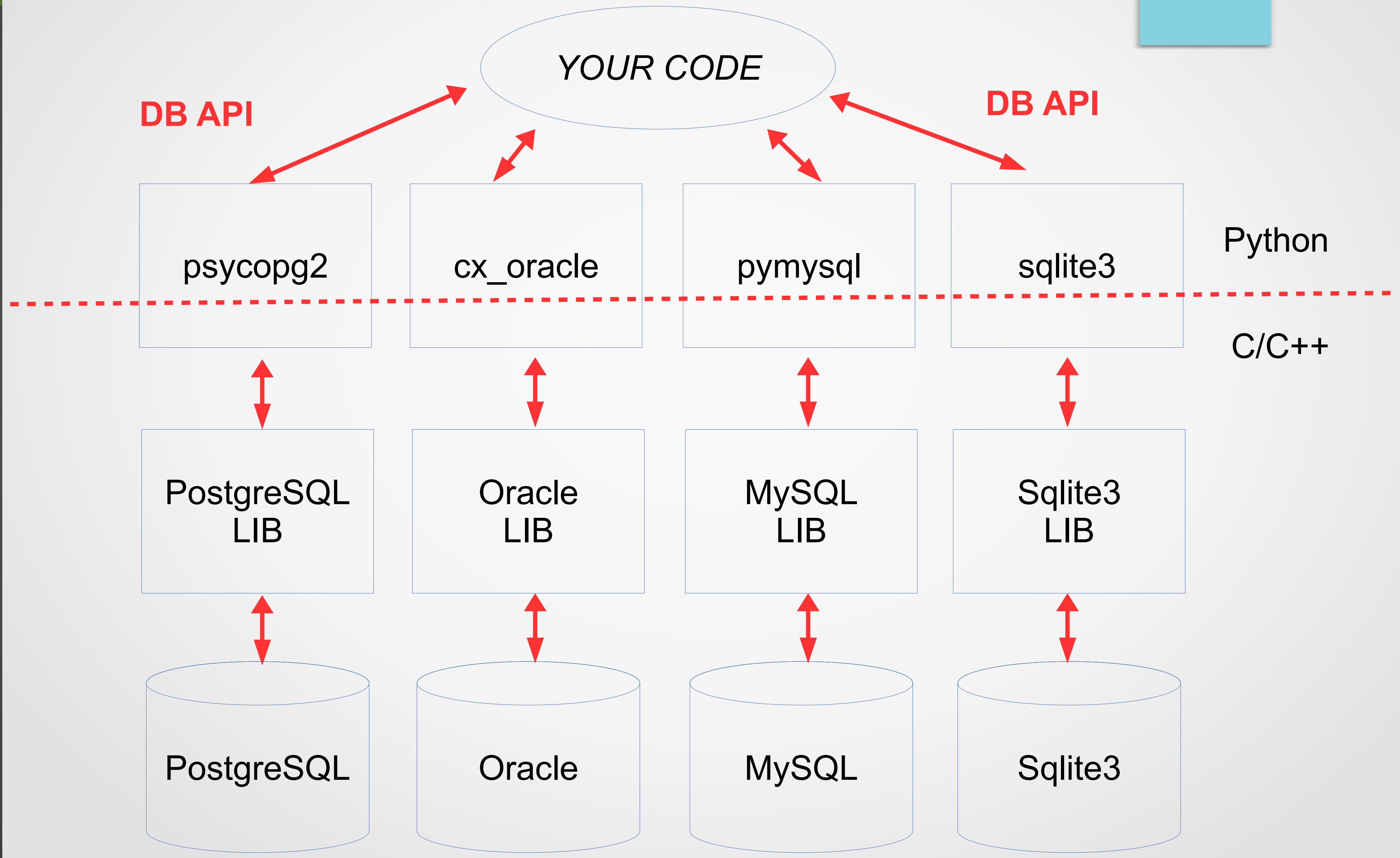
Regular Expressions



A Python Class


DATA (private)
all data variables start with _
available via ***self***
State
Behavior
Public Methods & Properties
Private Methods
start with _
Called from class or instance
Called from **self**
API

Python DB architecture



DB API

- conn = *package*.connect(*server, db, user, password, etc.*)
- cursor = conn.cursor()
- num_lines = cursor.execute(*query*)
- num_lines = cursor.execute(*query-with-placeholders, param-iterable*))
- all_rows = cursor.fetchall()
- some_rows = cursor.fetchmany(n)
- one_row = cursor.fetchone()
- conn.commit()
- conn.rollback()

SqlAlchemy ORM

DBMS Table

```
create table person (
    id int autoincrement,
    firstname varchar(30),
    lastname varchar(30),
    age int,
)
```

Python class

```
class person(base):
  id = Column(
        Integer,
        primary_key=True
   )
   last_name = Column(String(50))
   first_name = Column(String(50))
   age = Column(Integer)
```

ElementTree

presidents.xml

```
<presidents>
     <president term=1>
          <lastname>Washington</lastname>
          <firstname>George</firstname>
    </president>
    <president term=2>
          <lastname>John</lastname>
          <firstname>Adams</firstname>
    </president>
</presidents>
```

ElementTree

```
Element
       tag='presidents'
   Element {'term':1 }
      tag='president'
        Element
            tag='lastname'
            text='Washington'
        Element
            tag='firstname'
            text='George'
    Element {'term':2 }
      tag='president'
         Element
              tag='lastname'
              text='Adams'
         Element
             tag='firstname'
             text='John'
```

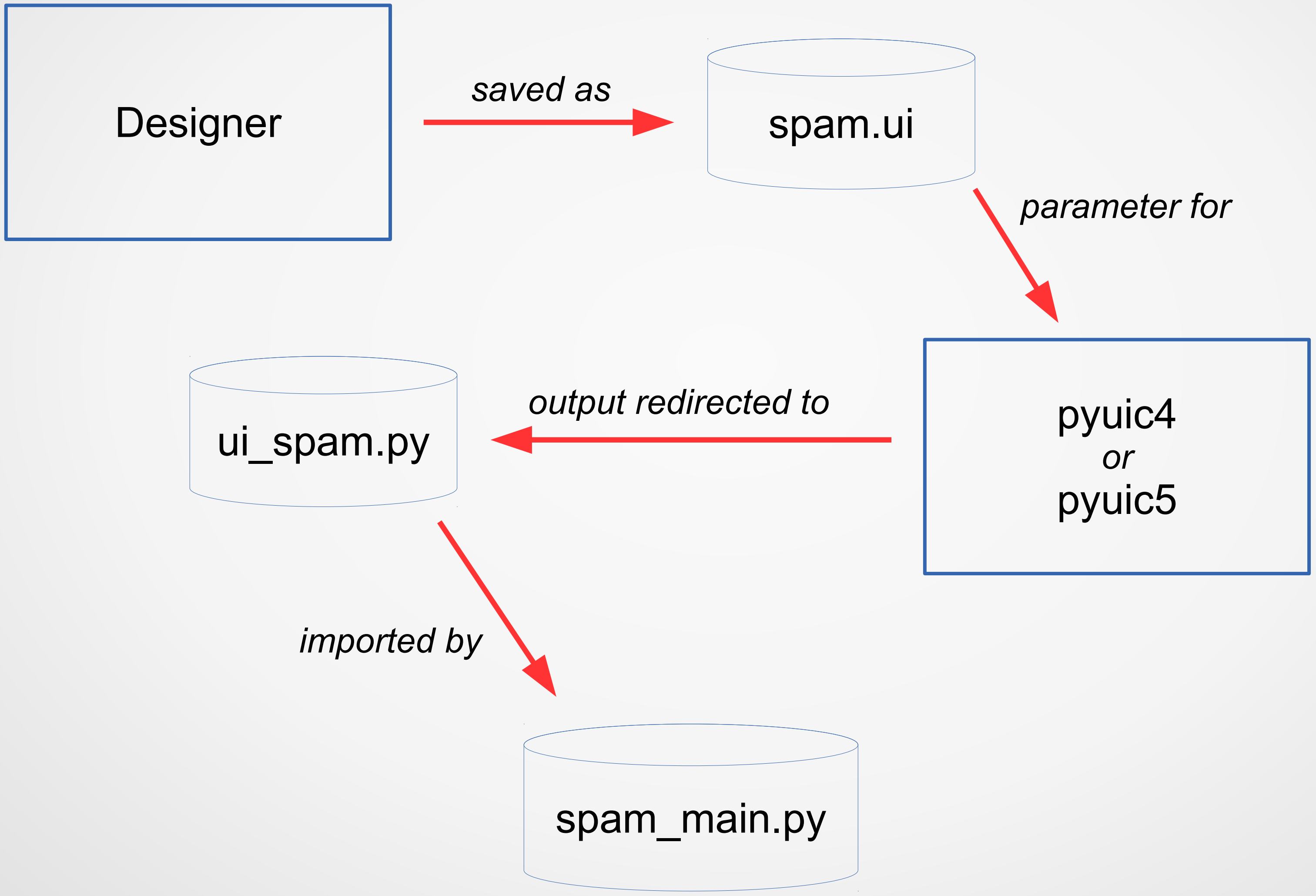
Good sources of Python books

- http://www.packtpub.com
- http://www.oreilly.com

Accessing Excel from Python

- pandas.read_excel()
- openpyxl
- win32com (requires Excel to be running)
- use CSV/TSV
- xlrd, xlwt, xlutil

PyQt Designer Workflow



Jupyter Notebook or IDE?

- Jupyter Notebook
  - Exploratory
  - Temporary
  - Experimental
  - Self-contained
  - Share results
  - Easy visualization
  - One file

- IDE (PyCharm, Spyder, ...)
  - Structured
  - Permanent
  - Modular
  - Share code
  - Development tools
  - GUI takes more effort
  - Many files

Pandas Dataframe Indexing

- DF.*indextype*[row_indexer, column_indexer]
  - Default indexer is **:** (all values)
  - Indexer can be
    - Label (examples: 'a', 5, 'result')
    - List of labels (examples: ['a', 'b', 'e'], [5, 4, 1])
    - Slice (example: 'a':'f', 2:3, 3:, 20150123: :
- Index types
  - .loc (label or Boolean array, NOT positional)
  - .iloc (integer or Boolean array, positional)
  - .ix (hybrid – primarily label, falls back to integer)

Decorator Syntax

```
@mydecorator
def myfunction():
    pass
```

same as

```
myfunction = mydecorator(myfunction)
```

```
@mydecorator(myparam)
def myfunction():
    pass
```

same as

```
myfunction = mydecorator(myparam)(myfunction)
```

Wheels

- Universal Wheel (all platforms)
  - Written for both Python 2 and Python 3
  - No extensions
- Pure Python Wheel (all platforms)
  - Written for Python 2 or Python 3
  - No extensions
- Platform Wheel (platform-specific)
  - Written for Python 2 or Python 3
  - Has extensions
  - Automatically created if non-Python code present

Context managers

```
with EXPR as VAR:
        BLOCK

mgr = (EXPR)
exit = type(mgr).__exit__  # Not calling it yet
value = type(mgr).__enter__(mgr)
exc = True
try:
    try:
        VAR = value  # Only if "as VAR" is present
        BLOCK
    except:
        # The exceptional case is handled here
        exc = False
        if not exit(mgr, *sys.exc_info()):
            raise
        # The exception is swallowed if exit() returns true
finally:
    # The normal and non-local-goto cases are handled here
    if exc:
        exit(mgr, None, None, None)
```

Why ranges are inclusive/exclusive (Edsger W. Djikstra)

- 2, 3, 4, 5
  - 2:6 inc/exc
  - 1:5 exc/inc
  - 2:5 inc/inc
  - 1:6 exc/exc
- 0, 1, 2, 3
  - 0:4 inc/exc
  - -1:3 exc/inc
  - 0:3 inc/inc
  - -1:4 exc/exc

- No Negative numbers
- Stop – start is # values
- Upper bound is lower bound of adjacent range

- -2, -1, 0, 1
  - -2:2 inc/exc
  - -3:1 exc/inc
  - -2:1 inc/inc
  - -3:2 exc/exc

Python IDEs for science and engineering

- PyCharm
- Spyder
- Roadeo
- Atom (with Hydrogen plugin)
- Sublime Text 3
- Python for Visual Studio code
- Eclipse with PyDev

What LDAP is not

- LDAP is not a server
- LDAP is not a database
- LDAP is not a network service
- LDAP is not an authentication procedure
- LDAP is not a user/password repository
- LDAP is neither open source nor closed source
- LDAP is not a product

LDAP is a PROTOCOL

Packages to install for Django classes

- django (conda)
- Environ
- dotenv
- cookiecutter
- django-environ
- django-debug-toolbar
- django-crispy-forms
- django-cms-installer
- django-allauth
- django-extensions
- psycopg, cx_oracle, pymssql, pyodbc, etc.