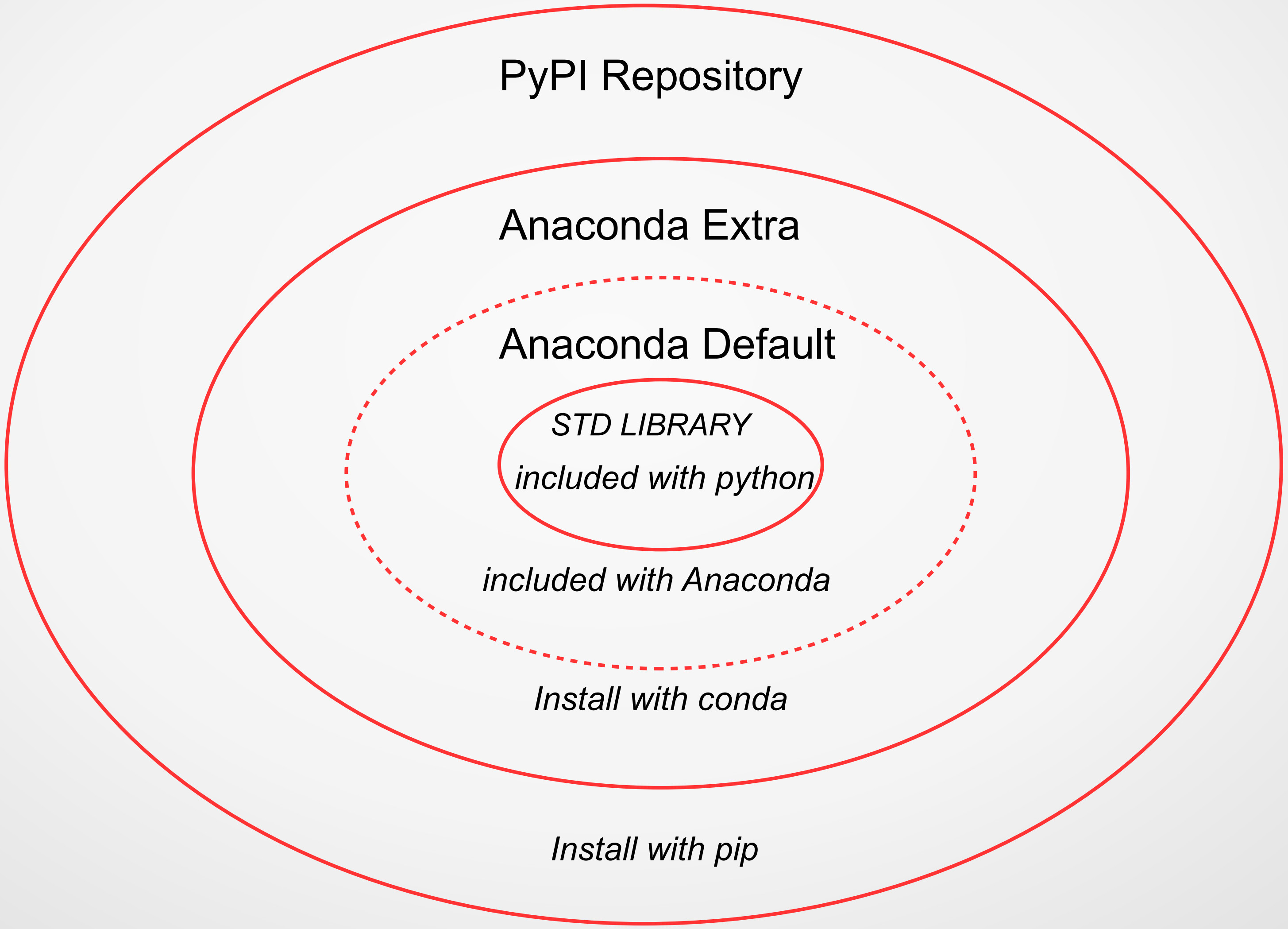
Configuring Visual Studio Code

- Auto-save
  - Search for "auto save"
  - Set to **afterDelay**
- Launch folder
  - Search for "execute in"
  - Check box for Python > **Terminal: Execute in File Dir**
- Minimap
  - Search for "minimap enabled"
  - Uncheck **Editor > Minimap: Enabled**

Configuring Visual Studio Code Fonts

- Editor font size
  - Search for "editor font size"
  - Set **Editor: Font Size** to desired size
- Terminal font size
  - Search for "terminal font size"
  - Set **Terminal > Integrated: Font Size** to desired size
- Themes
  - Go to **File > Preferences > Theme > Color Theme**
  - Choose new theme if desired

Python Modules (using Anaconda)



What Can Python Do?

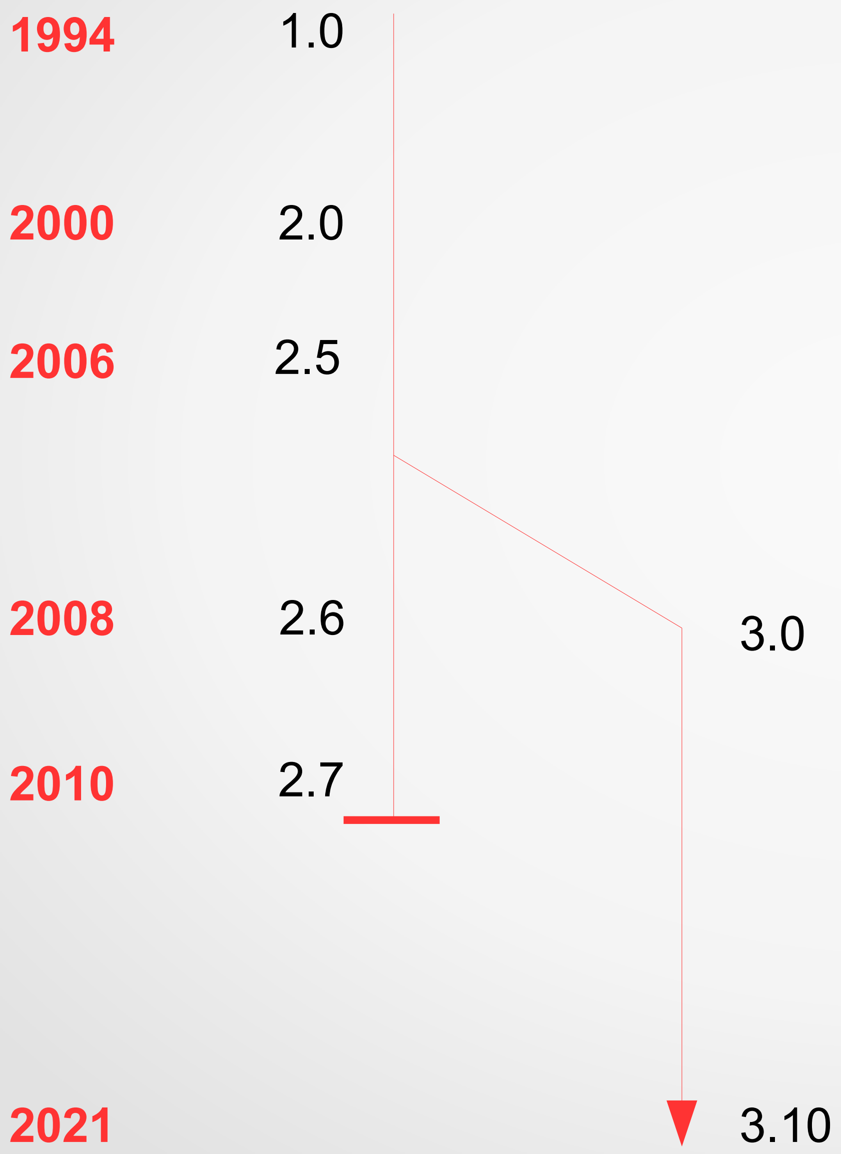
- Data science
  - Data visualization
- Web apps and APIs
- Cloud apps
- Data mining/web scraping
- Desktop GUI apps
- Sys Adm (Windows, Mac, Linux)
- DevOps/NetOps
- Scientific/Engineering apps

Advantages of Python

- Easy to learn
- Readable
- Multi-paradigm
  - Procedural
  - Functional
  - Object-oriented
- Modular
- Exceptions
- Large Standard library
- Many third-party modules (science, web, admin, ...)
- Fun!

Disadvantages of Python

Python Evolution

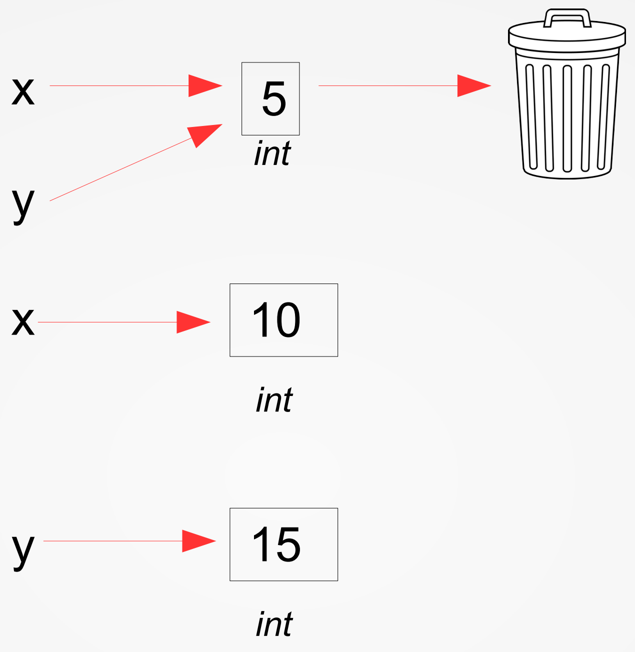


Desirable IDE Features

- Autocomplete
- Autoindent
- Syntax checking/highlighting
- Debugging
- Integration with source code control (e.g. git)
- Navigation
- Smart search-and-replace
- Project management
- Code snippets (AKA macros)
- File templates
- Variable explorer
- Python console
- Interpreter configuration (including installing modules)
- Unit testing tools

Creating variables

```
x = 5
y = x
x = 10
y = 15
```



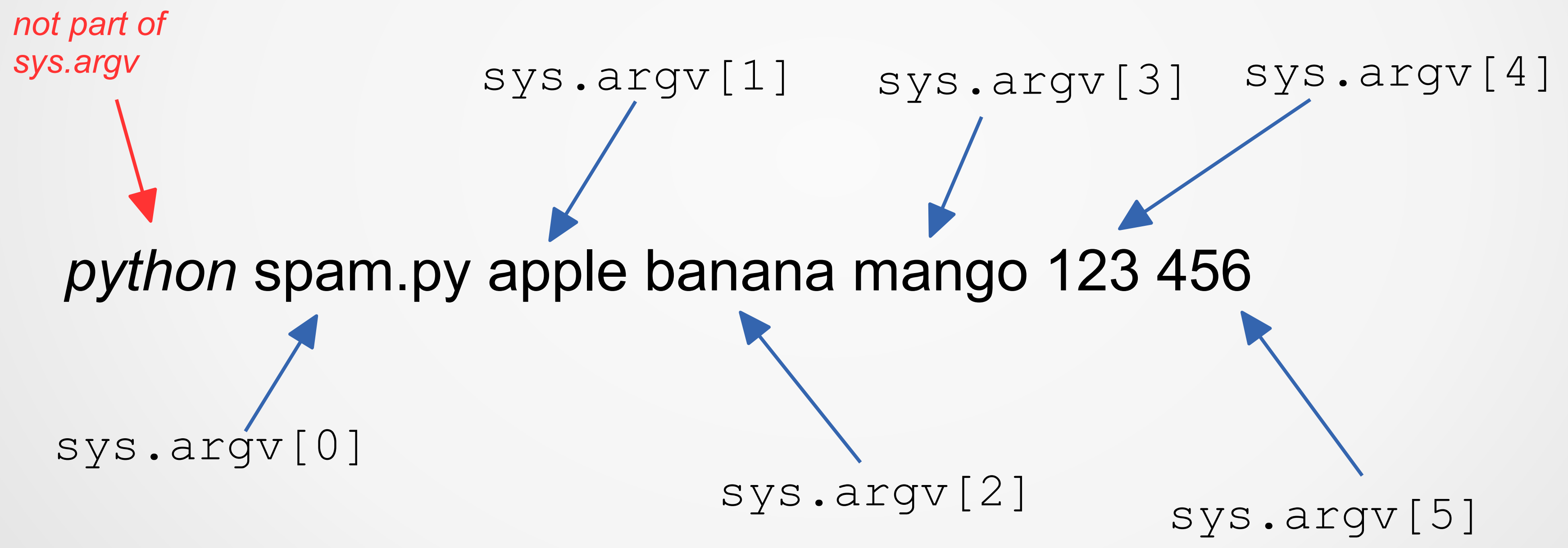
String literals

- Single-delimited (AKA single-quoted)
  - `'spam\n'        "spam\n"`
- Triple-delimited (AKA triple-quoted)
  - `'''spam\n'''    """spam\n"""`
- Raw
  - `r'spam\n'`

```
"Guido's the BDFL"

 """Guido's the "BDFL" of Python"""
```

Command Line Parameters

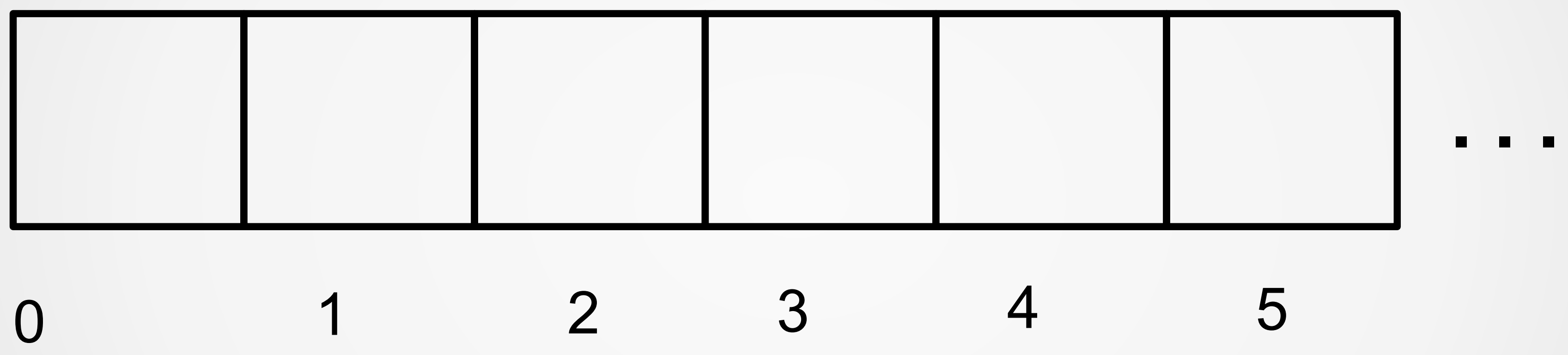


Indenting blocks

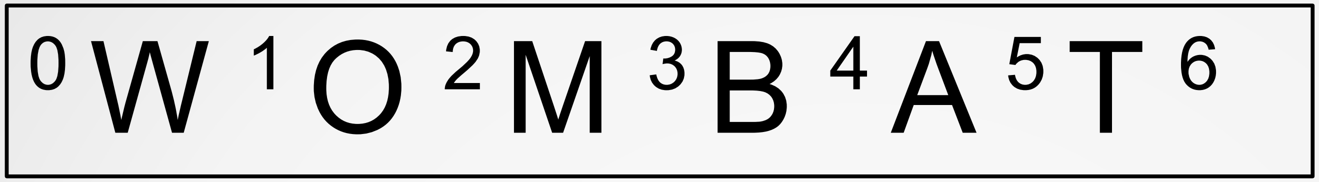
```
Block statement:
●●●●Statement
●●●●Statement
●●●●Nested Block Statement:
●●●●●●●●Statement
●●●●●●●●Statement
●●●●Statement
●●●●Statement

Statement
```

Sequences



Slices



```
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

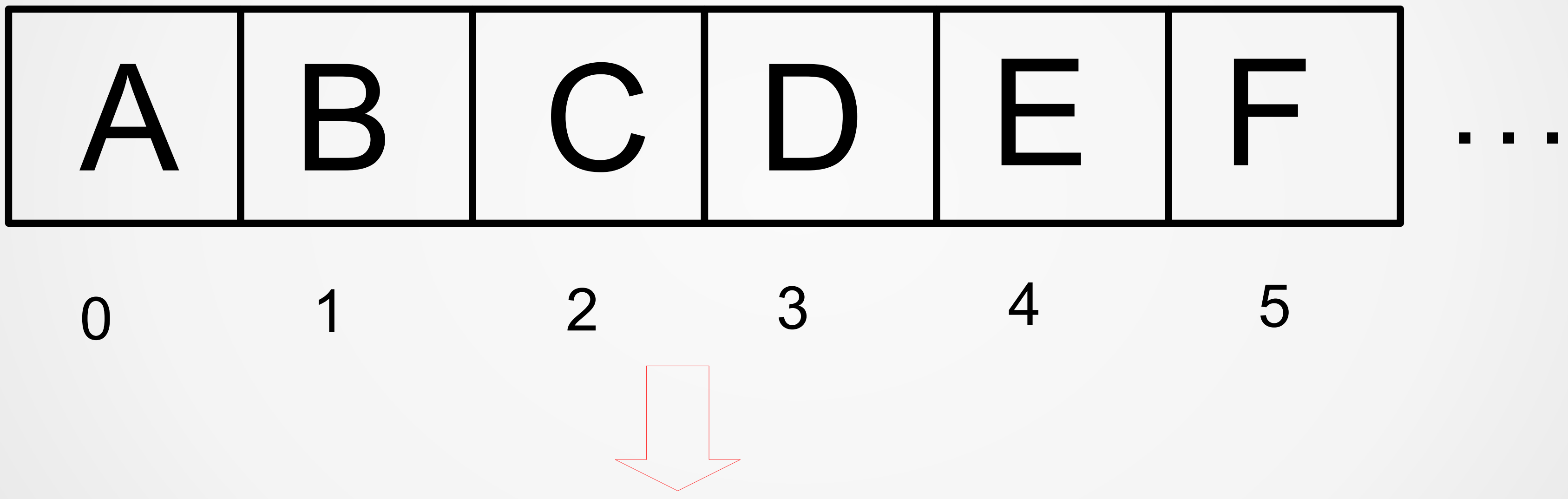
Lists

- Dynamic Array
- Mutable/unhashable
- Items usually same/similar type
- Position doesn't matter
- Typical use: looping
- Think "ARRAY"

Tuples

- Collection of related fields
- Immutable/hashable
- Items mixed and matched
- Position matters
- Typical use: unpacking
- Think "STRUCT" or "RECORD"

enumerate()



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

Iterables

IN MEMORY!

VIRTUAL!

Iterables

Containers (AKA collections)

Generators

Sequences

str
bytes
list
tuple
    collections.namedtuple

returned by
sorted()
list comprehension
str.split()
etc.

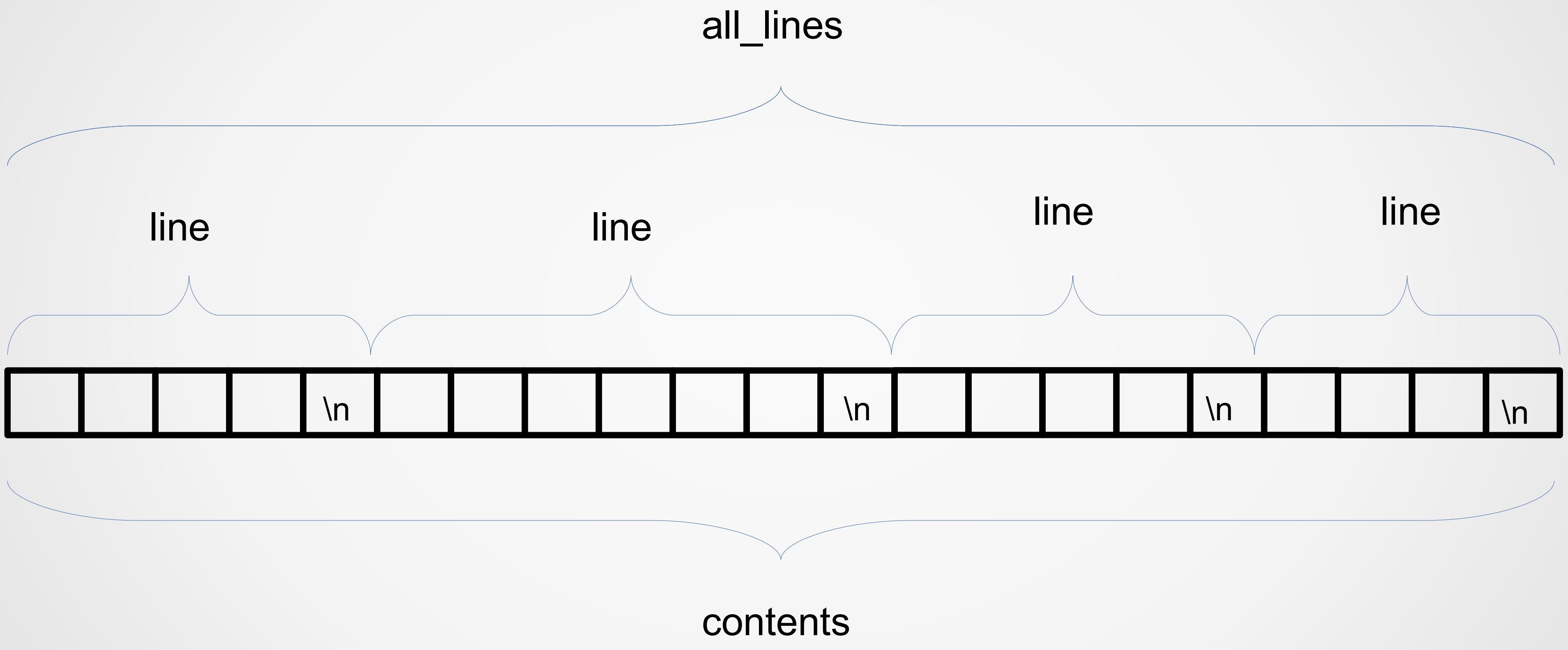
Mappings

dict
    collections.defaultdict
    collections.Counter
set
frozenset

returned by
dict comprehension
set comprehension
etc.

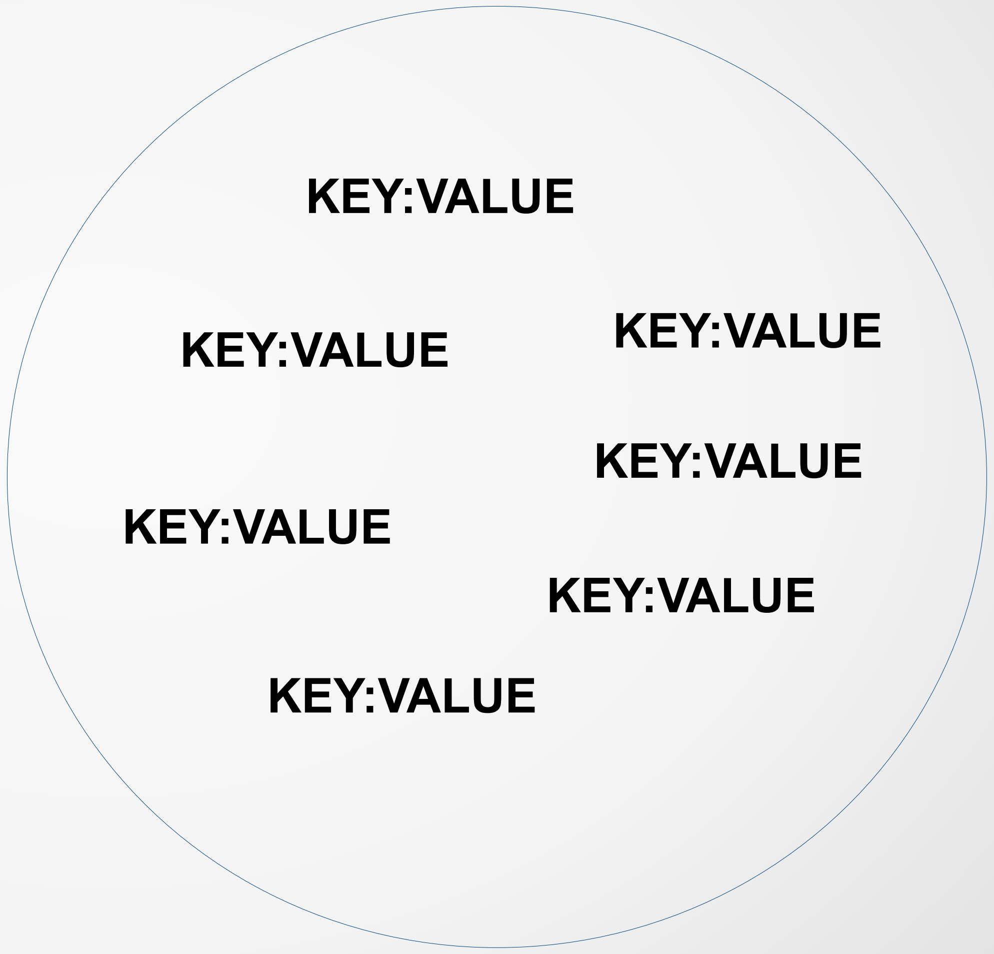
returned by
open()
range()
enumerate()
dict.items()
zip()
Itertools.groupby()
itertools.izip()
reversed*()*
generator expression
generator function
generator class
etc.

Reading text files



Dictionary

- Key/value pairs
- Keys are unique and immutable
- Keys stored in insertion order
- Use .items() to loop through k/v pairs

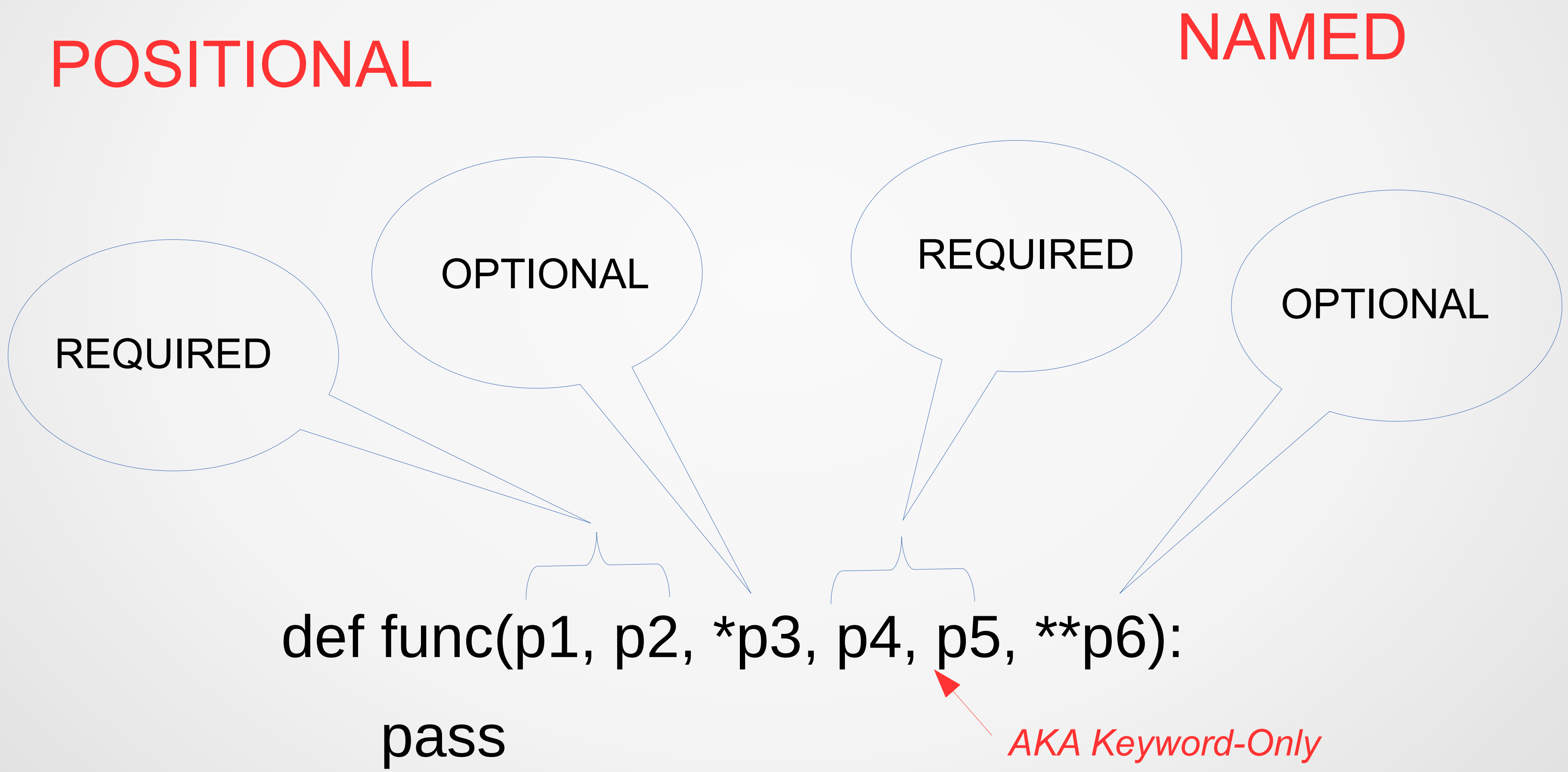


dict.items()

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

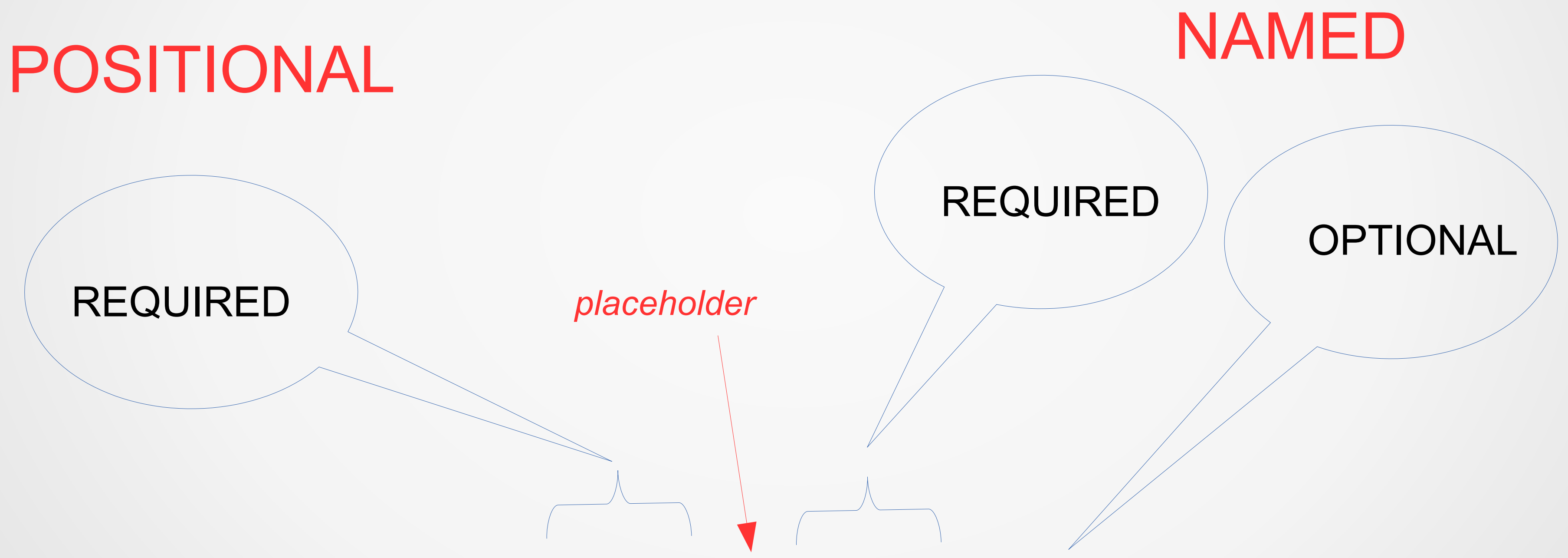
(A, 100), (B, 200), (C, 300), (D, 400), (E, 500), (F, 600) ...

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

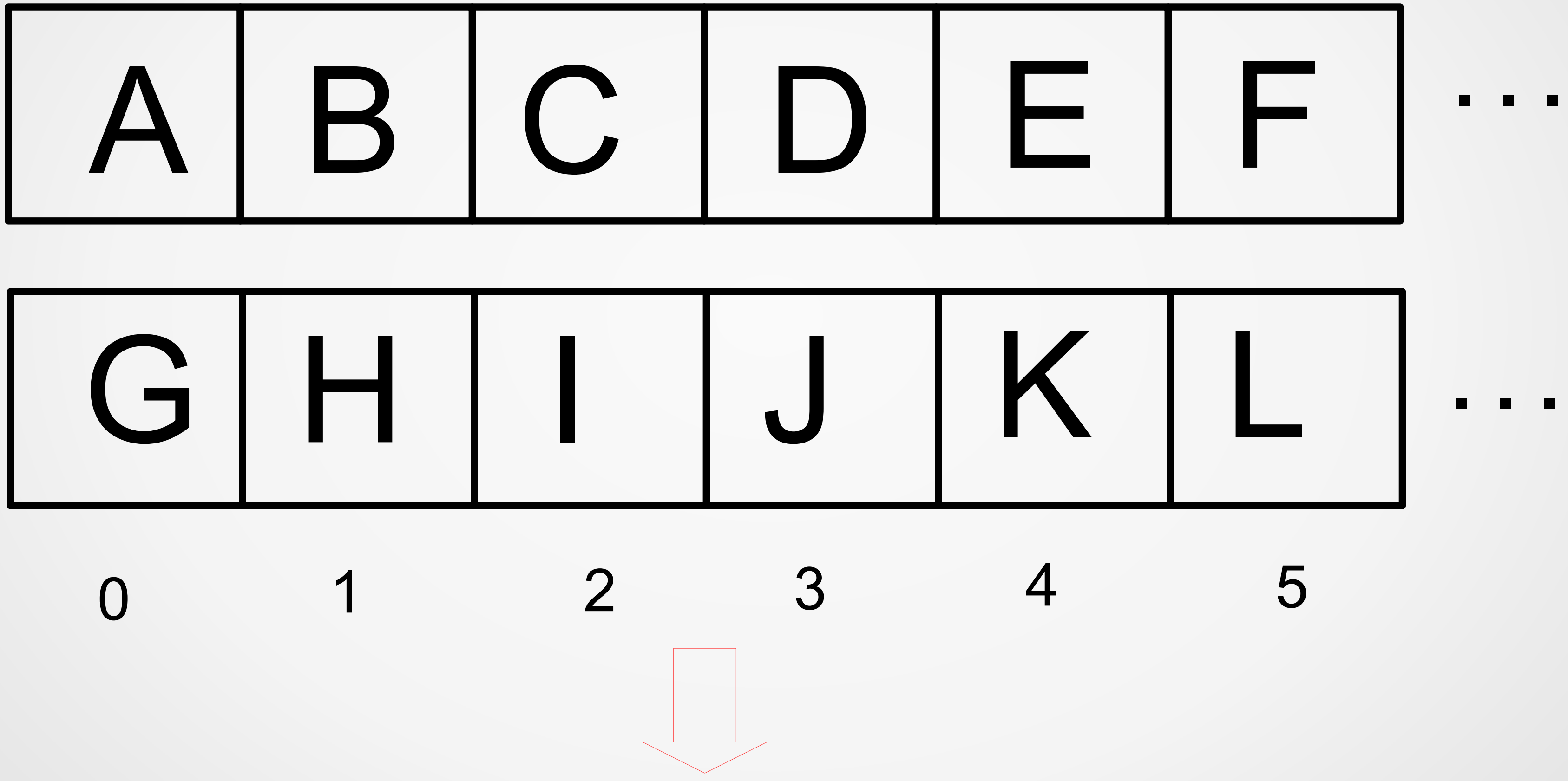
- Read-only reference is passed
- Mutables may be changed via reference
- Immutables may not be changed

```
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

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

- Nested iterables

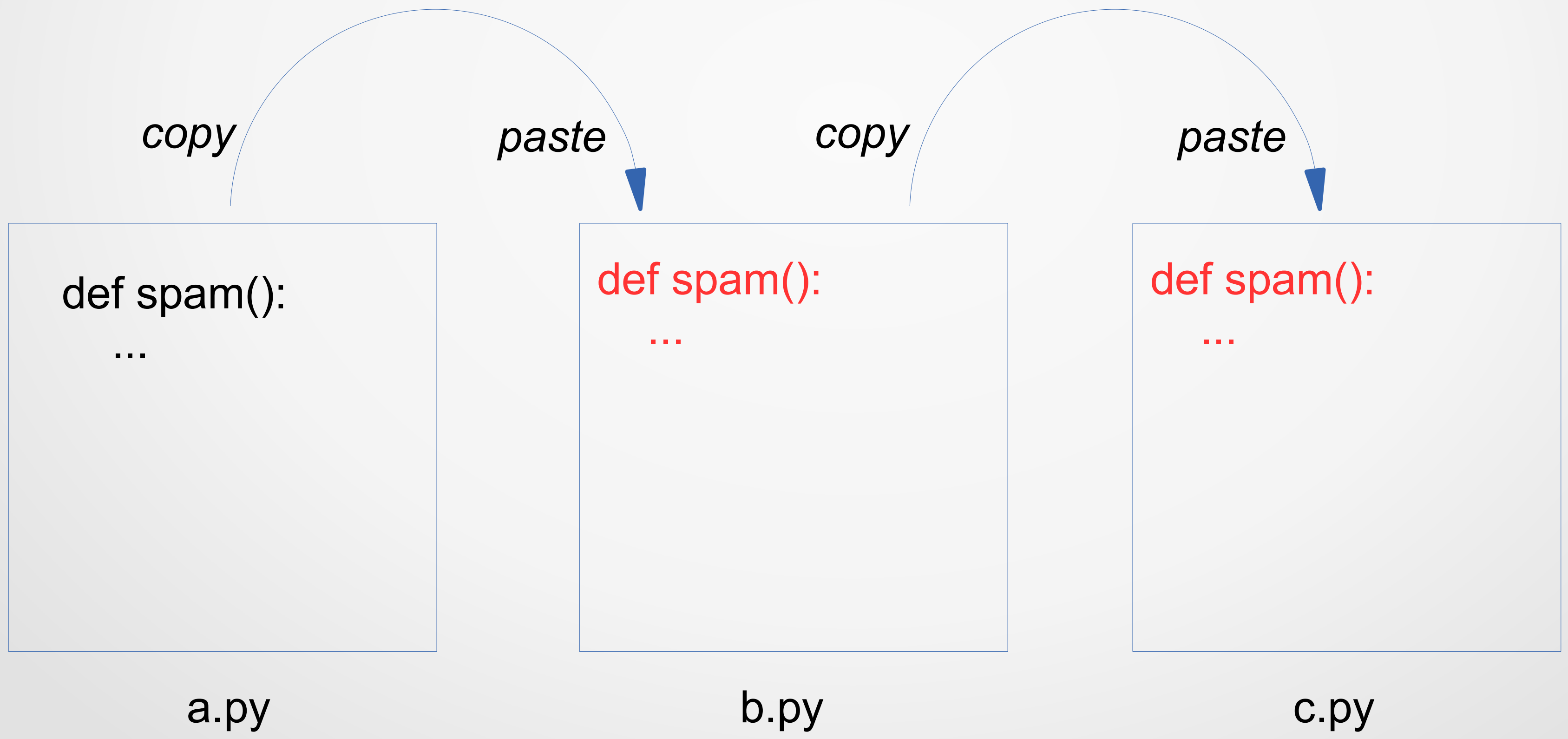
  [O_1, O_2, O_3], [O_1, O_2, O_3], [O_1, O_2, O_3], …

  – ***dict***.items() *special case of nested iterables*

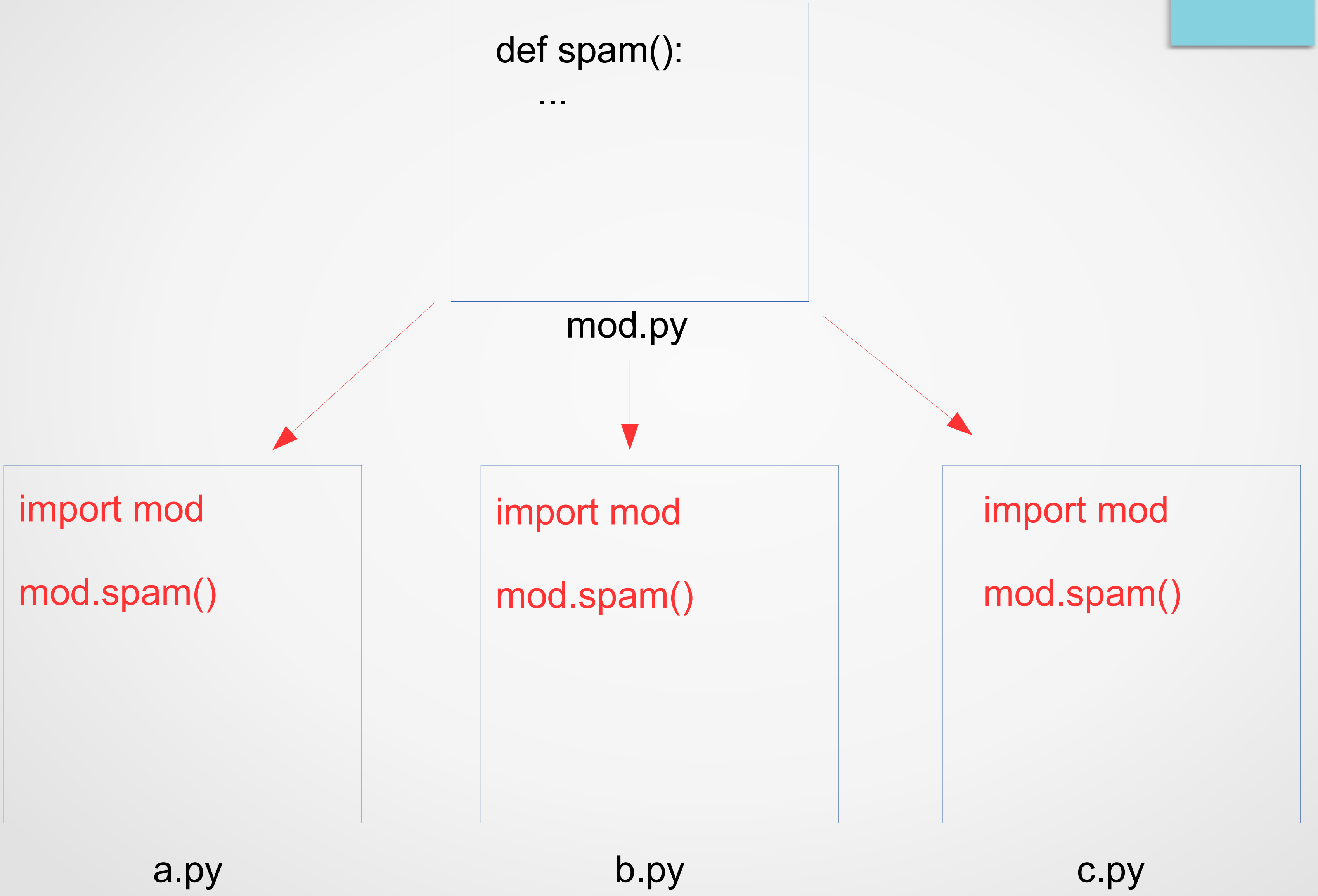
    (key, value), (key, value), (key, value), ...

Copying and pasting functions

DON'T DO THIS!!



Using a module



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

$\text{Branch}_1 \mid \text{Branch}_2$

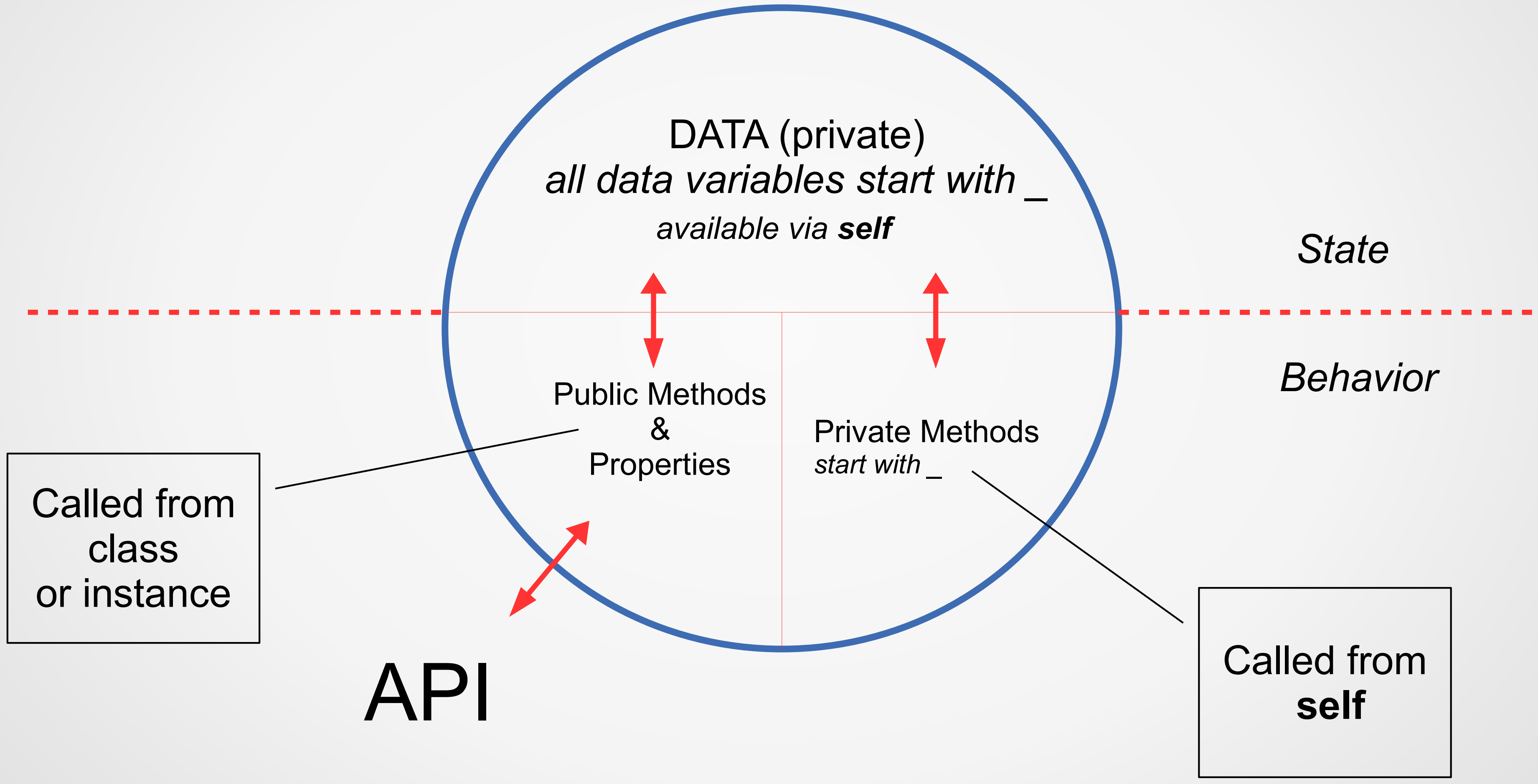
$\text{Atom}_1\text{Atom}_2\text{Atom}_3(\text{Atom}_4\text{Atom}_5\text{Atom}_6)\text{Atom}_7$

A a 1 ;

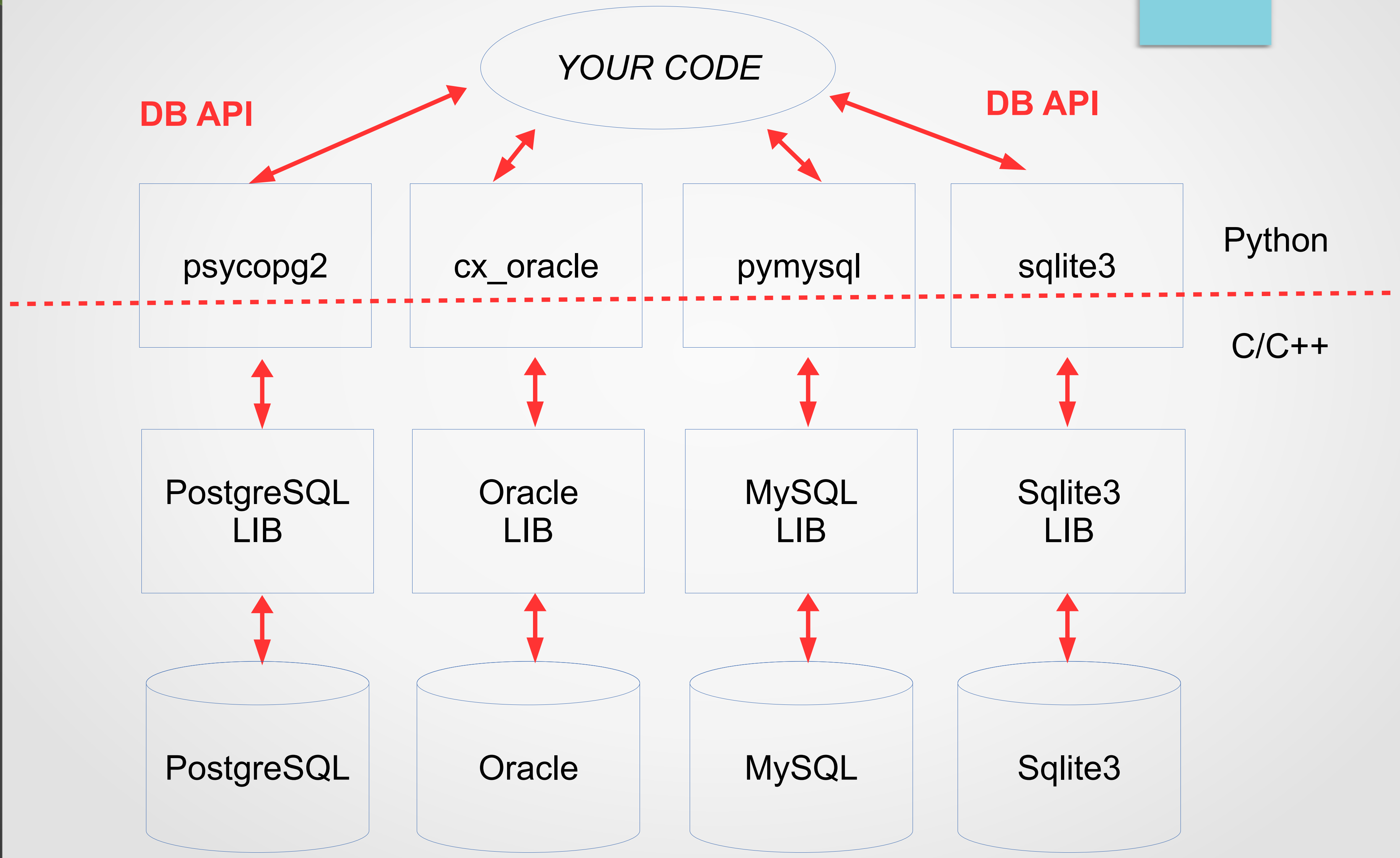
. \d \w \s
[abc]
[^abc]

$\text{Atom}_{\textit{repeat}}$

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
API
Called from **self**

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()

How a *for* loop really works

```
values = ["a", "b", "c"]
```

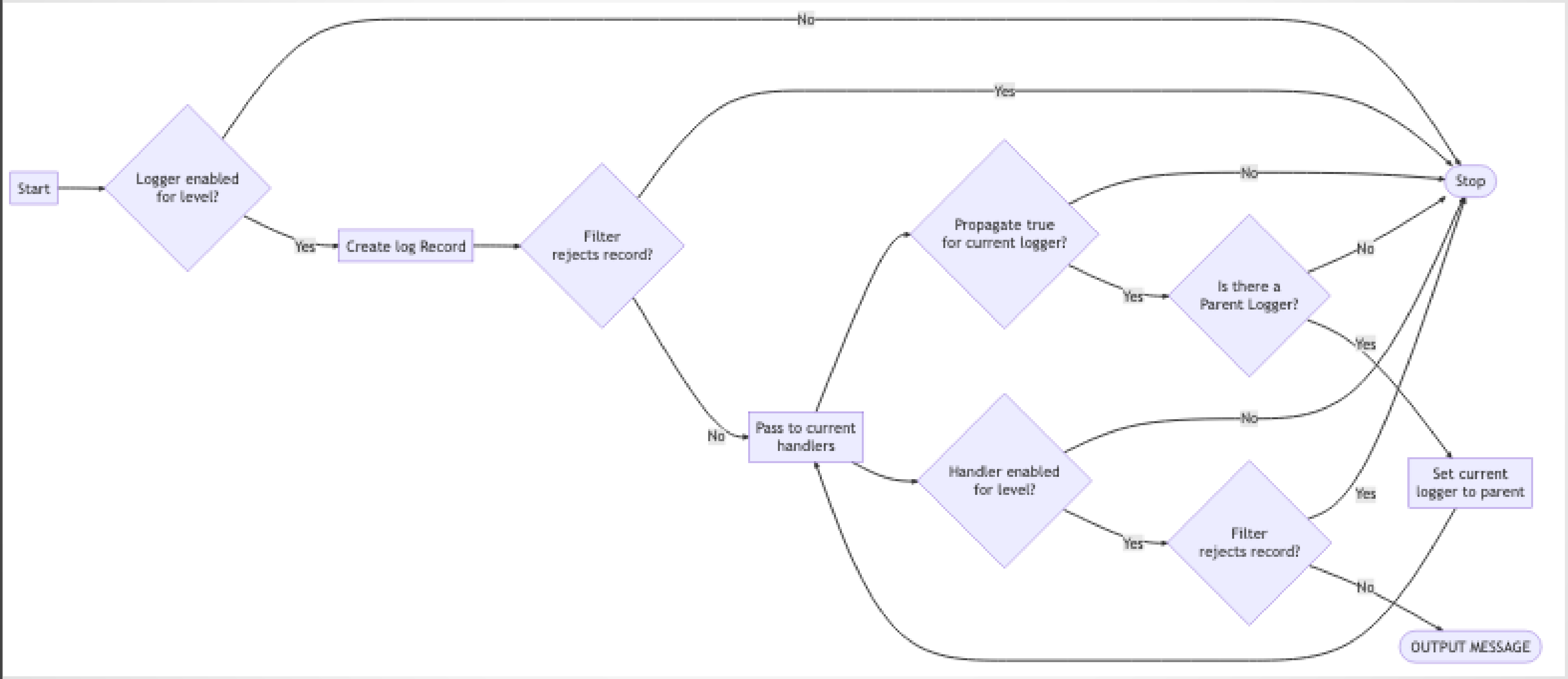
for loop:

```
for value in values:
    print(value)
```

while loop:

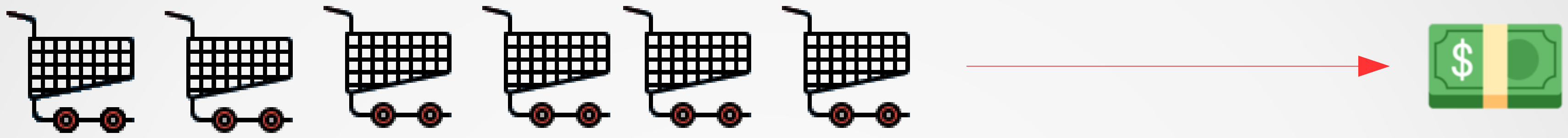
```
it = iter(values)
while True:
    try:
        value = next(it)
    except StopIterationError:
        break
```

Logging flowchart

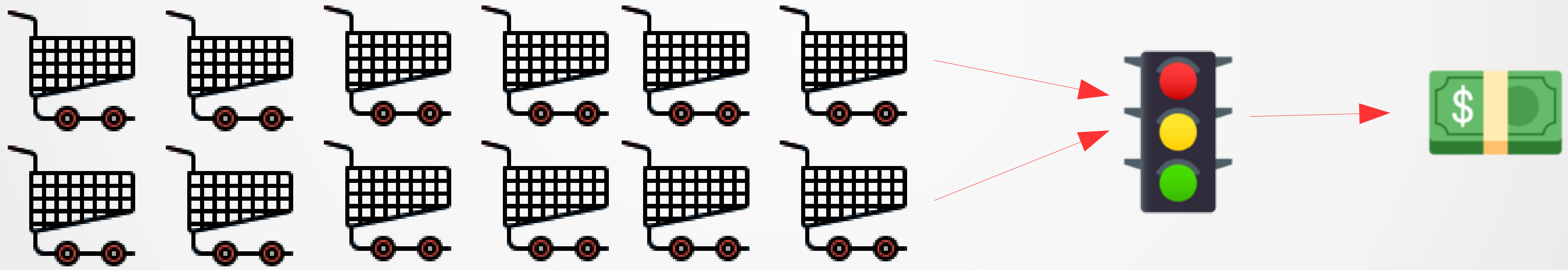

Start
Logger enabled for level?
No
Yes
Create log Record
Filter rejects record?
Yes
No
Pass to current handlers
Propagate true for current logger?
No
Yes
Is there a Parent Logger?
No
Yes
Handler enabled for level?
No
Yes
Filter rejects record?
Yes
No
Stop
Set current logger to parent
OUTPUT MESSAGE

Multiprogramming

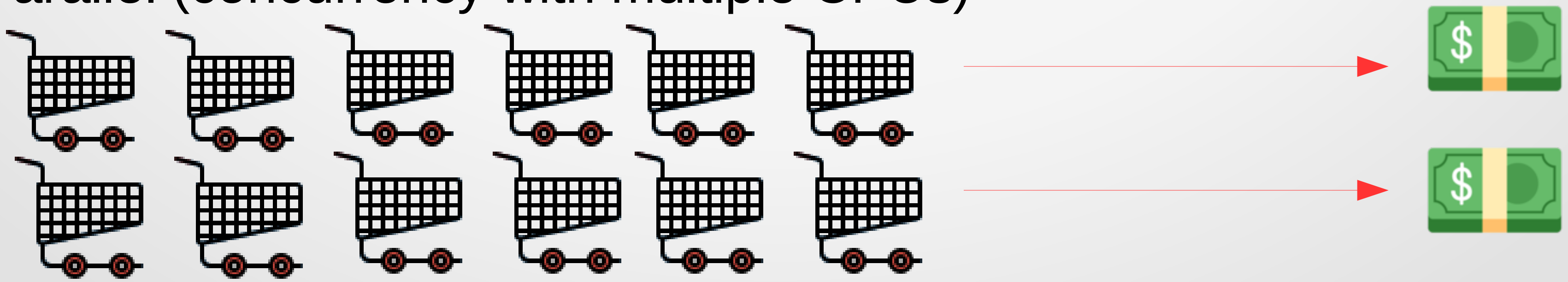
- Sequential



- Multitasking (concurrency with 1 CPU)



- Parallel (concurrency with multiple CPUs)



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"
```

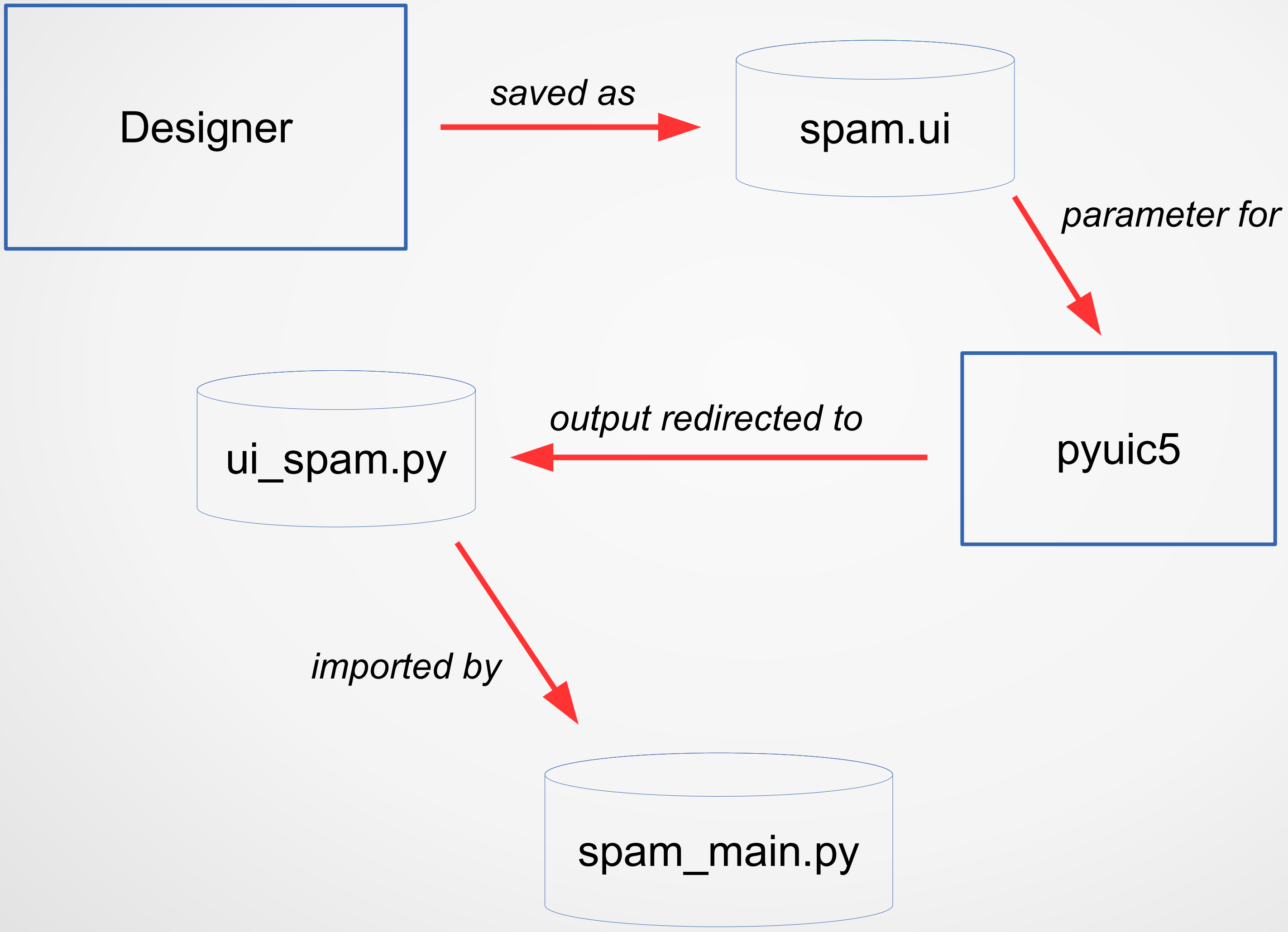
Good sources of Python books

- 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 vs. IDE

- Jupyter Notebook
  - Research
  - Exploratory
  - Experimental
  - Self-contained
  - Easy visualization
  - One file
  - Sharable

- IDE (PyCharm, Spyder, ...)
  - Production
  - Structured
  - Modular
  - Share code
  - Development tools
  - Harder visualization
  - Many files
  - Distibutable

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

URL Mapping

Show how the URL maps to the actual Django files, including the url conf and the views, and maybe the templates

•Two hard problems in computer science

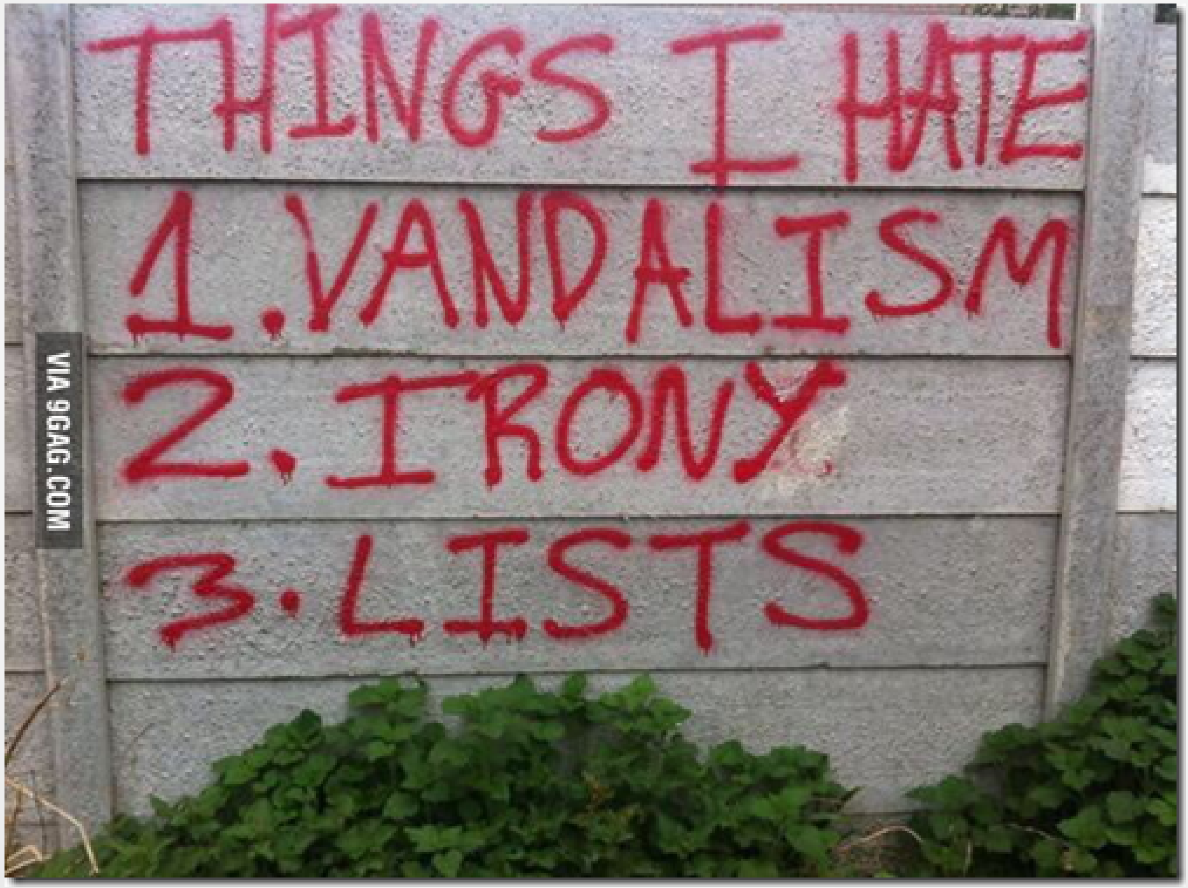
- cache invalidation
- naming things
- off-by-one errors

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)
```

Things I Hate



If programming languages were religions

- Perl would be Voodoo - An incomprehensible series of arcane incantations that involve the blood of goats and permanently corrupt your soul. Often used when your boss requires you to do an urgent task at 21:00 on friday night.

A Joke

- How do you tell the difference between a plumber and a chemist? Ask them to pronounce unionized.

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

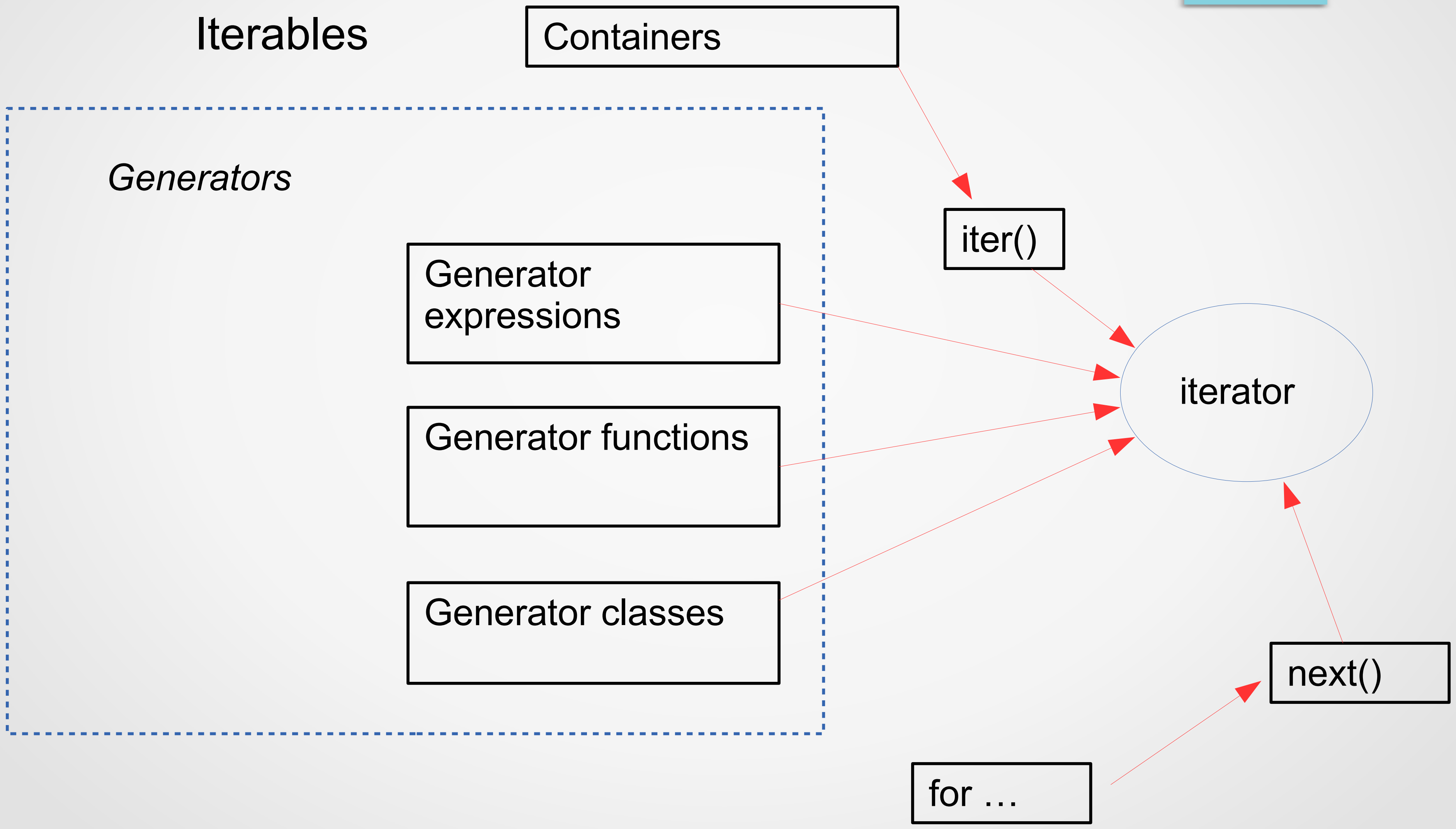
MongoDB Terminology

- _id – unique identifier in every record
- Collection – group of records ("table")
- Cursor – pointer to result set
- Database – Container of Collections ("database")
- Document – set of fields ("row" or "record")
- Field – name/value pair ("column")
- Embedded document – related data ("join")

Why use MongoDB

- Document-oriented
- Ad hoc queries
- Indexing
- Replication
- Load balancing

Iterables and iterators



Packages to install for Django classes

- django
- Environ
- dotenv
- cookiecutter
- django-environ
- django-debug-toolbar

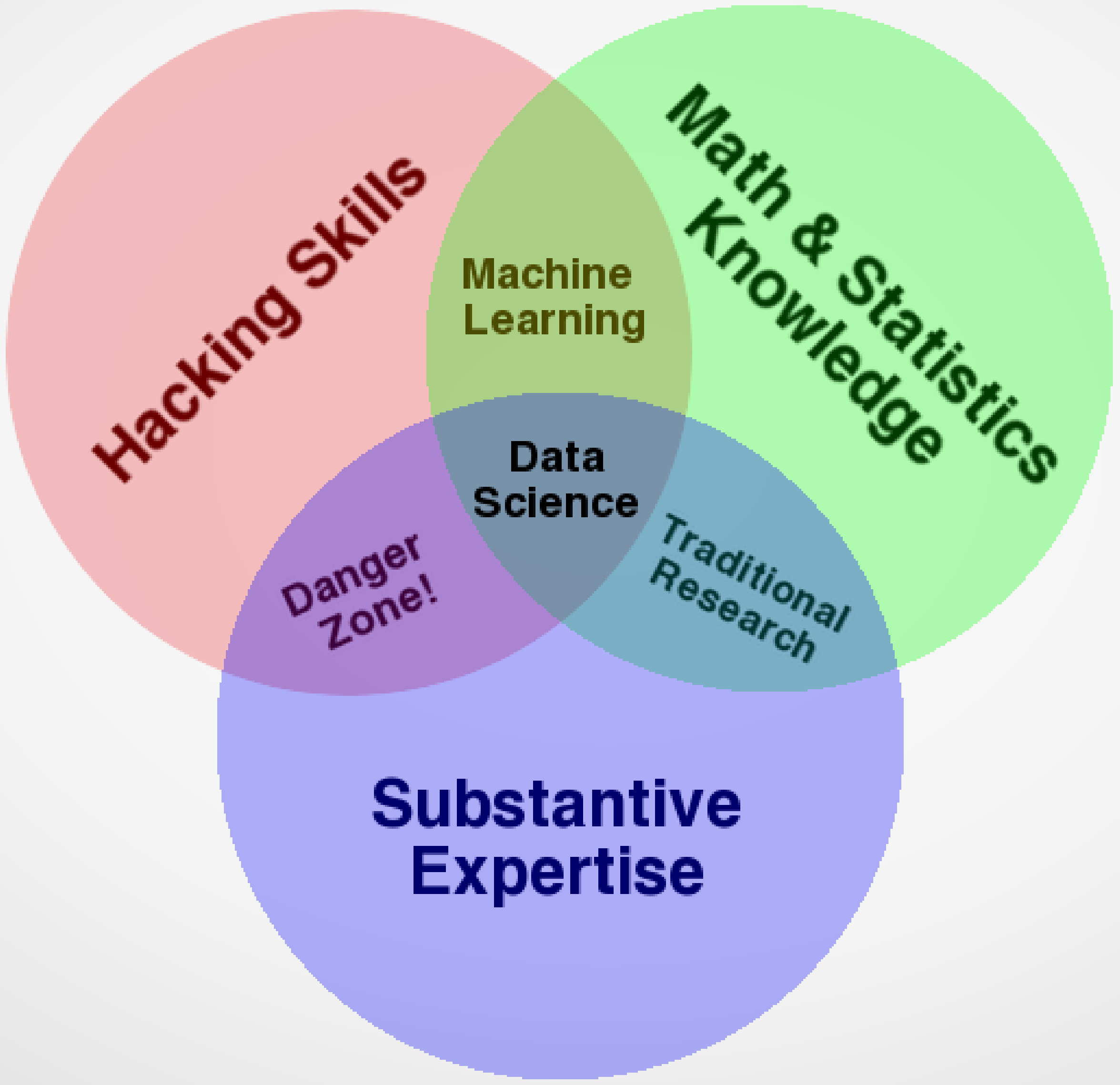
Ways to call C from Python

- Write Python-aware C code (tedious)
- Use SWIG to interface to existing C code
- Use Boost to interface to C code
- Use ctypes to access C dll/so/dylib
- Use cython with inline C code

Python Performance

1. Get your output correct
2. Write tests for the code that generates correct output
3. Optimize as much as you can
4. Benchmark
5. Run tests to make sure your code is correct

Drew Conway"s Venn Diagram of Data Science


Hacking Skills
Math & Statistics Knowledge
Machine Learning
Data Science
Danger Zone!
Traditional Research
Substantive Expertise