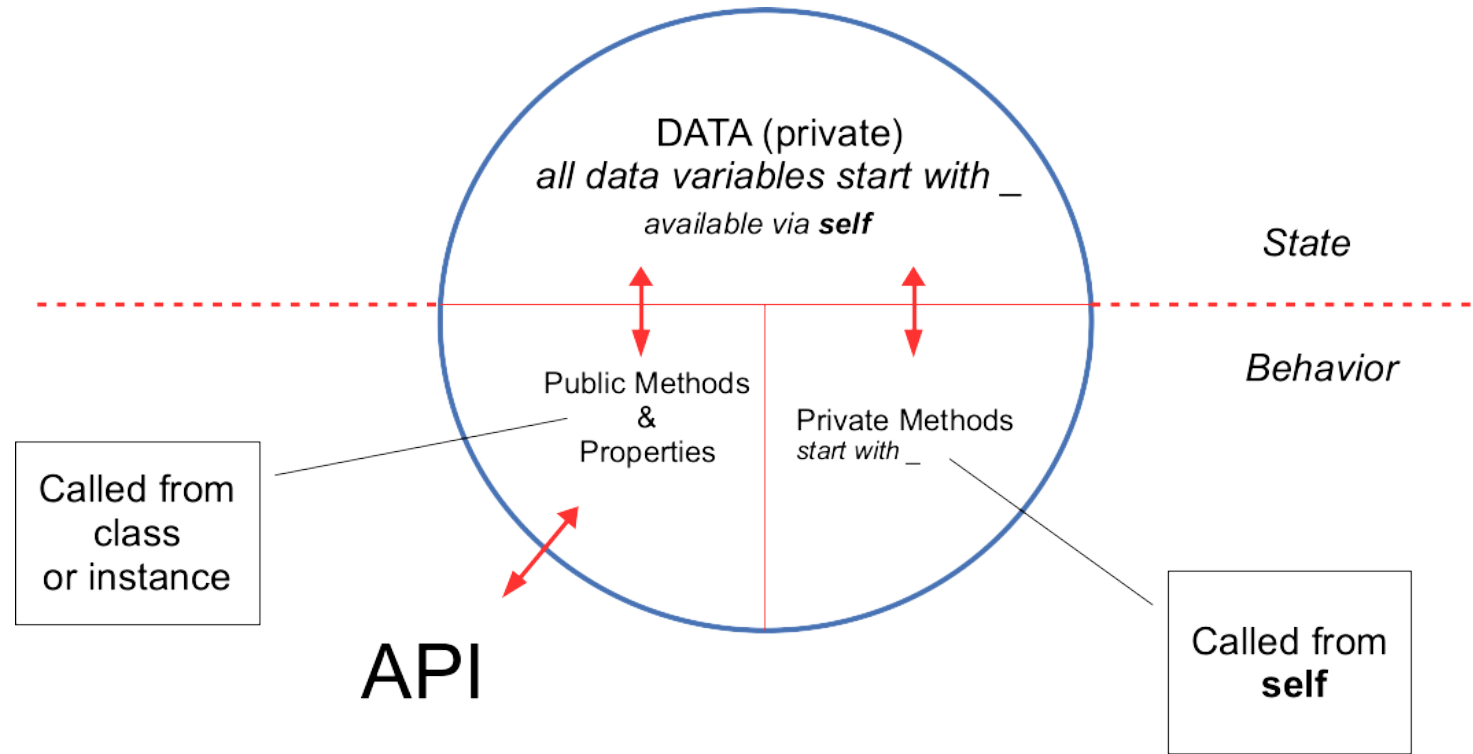


Advanced Python for JPMC

TTPS4805-GKJ

A Python Class



str() vs repr()

str()	repr()
For humans	How to reproduce object
"Informal" form	"Official" form
Info about object	Code to create object
If undefined, uses repr()	If undefined, uses object.__repr__()

ElementTree

XML

```
<presidents>
  <president term="1">
    <first>George</first>
    <last>Washington</last>
  </president>
  <president term="2">
    <first>John</first>
    <last>Adams</last>
  </president>
</presidents>
```

ElementTree

```
Element
  tag="presidents"
  Element {"term": "1" }
    tag="president"
    Element
      tag="first"
      text="George"
    Element
      tag="last"
      text="Washington"
  Element {"term": "2" }
    tag="president"
    Element
      tag="first"
      text="John"
    Element
      tag="last"
      text="Adams"
```

Decorators Save Typing

Instead of

```
def spam():  
    pass  
  
spam = deco(spam)
```

use

```
@deco  
def spam():  
    pass
```

spam is only typed once, instead of 3 times

Decorator Syntax

```
@DECORATOR  
def some_function():  
    pass
```

same as

```
some_function = DECORATOR(some_function)
```

Implementation

```
def DECORATOR(original_function):  
    @wraps(original_function)  
    def WRAPPER(*args, **kwargs):  
        # add code here  
        result = original_function(*args, **kwargs)  
        return result  
    return WRAPPER
```

Decorator with parameters

```
@DECORATOR(param, ...)  
def some_function():  
    pass
```

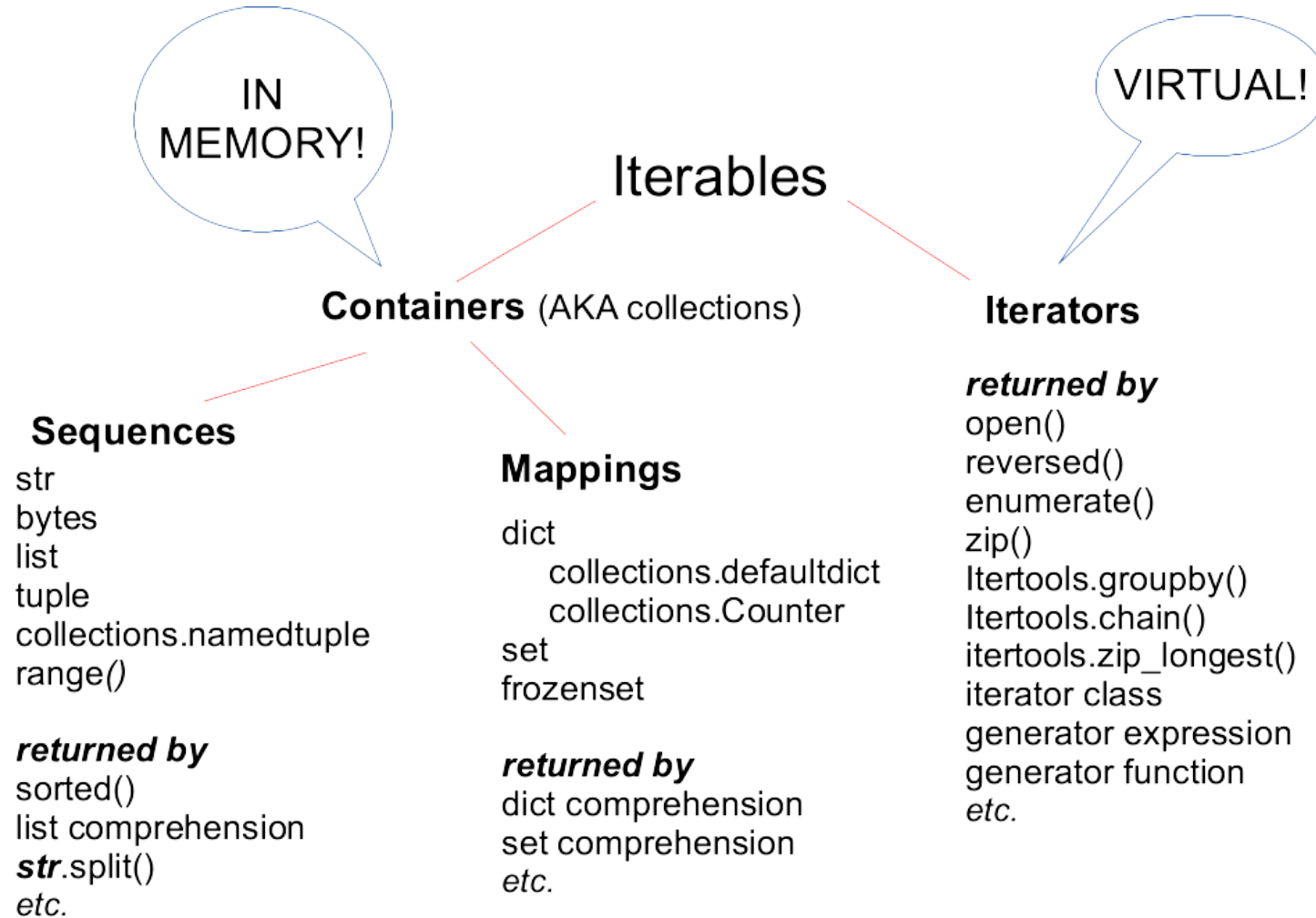
same as

```
some_function = DECORATOR(param, ...)(some_function)
```


Implementation

```
def DECORATOR(param, ...):  
    def WRAPPER_FACTORY(original_function):  
        @wraps(original_function)  
        def WRAPPER(*args, **kwargs):  
            # add code here using decorator params  
            result = original_function(*args, **kwargs)  
            return result  
        return WRAPPER  
    return WRAPPER_FACTORY
```

Iterables



Containers

- All elements in memory
- Can be indexed with []
- Have a length

Builtin containers

Sequences

`list`

`tuple`

`string`

`bytes`

`range`

Mapping types

`dict`

`set`

`frozenset`

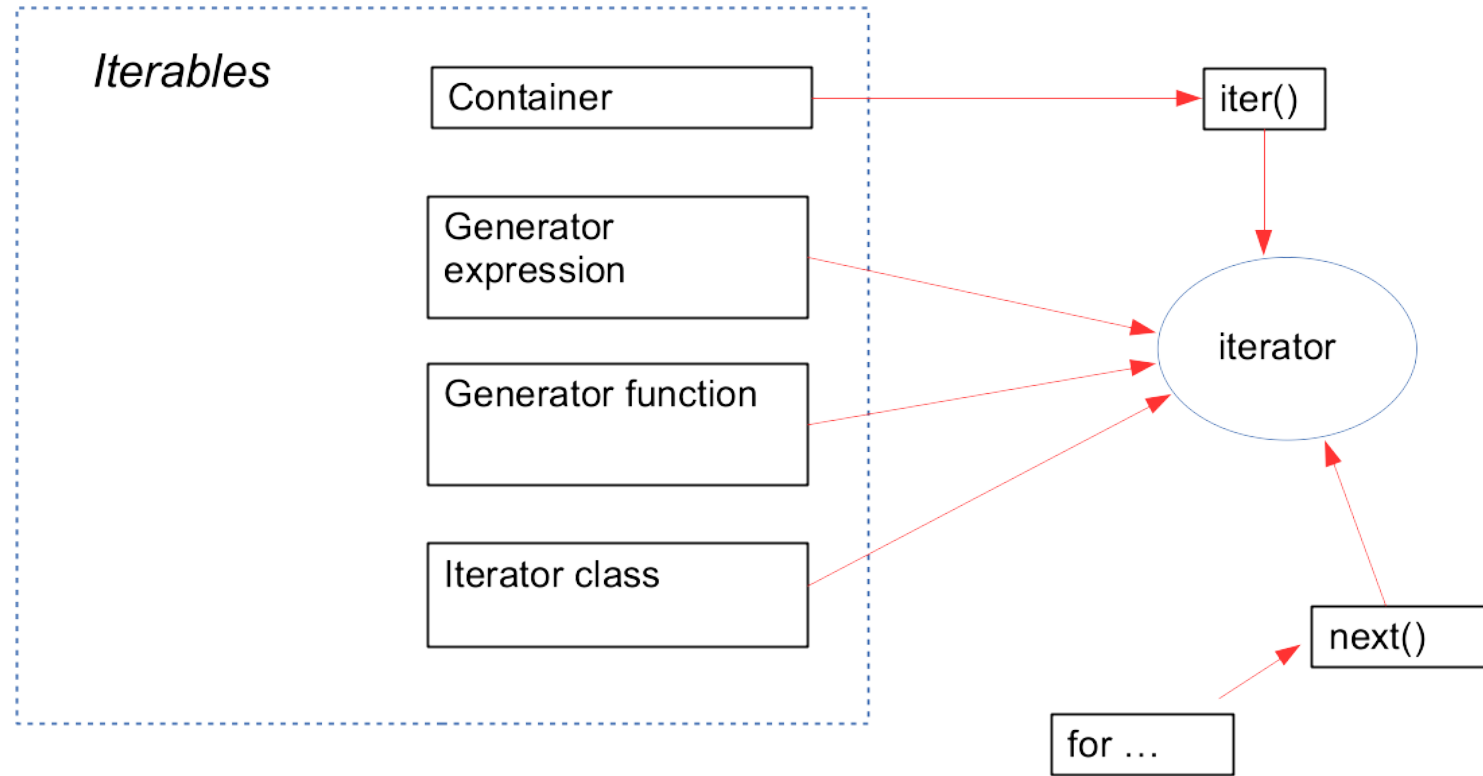
Iterators

- Virtual (no memory used for data)
- Lazy evaluation (JIT)
- Cannot be indexed with []
- Do not have a length
- One-time-use

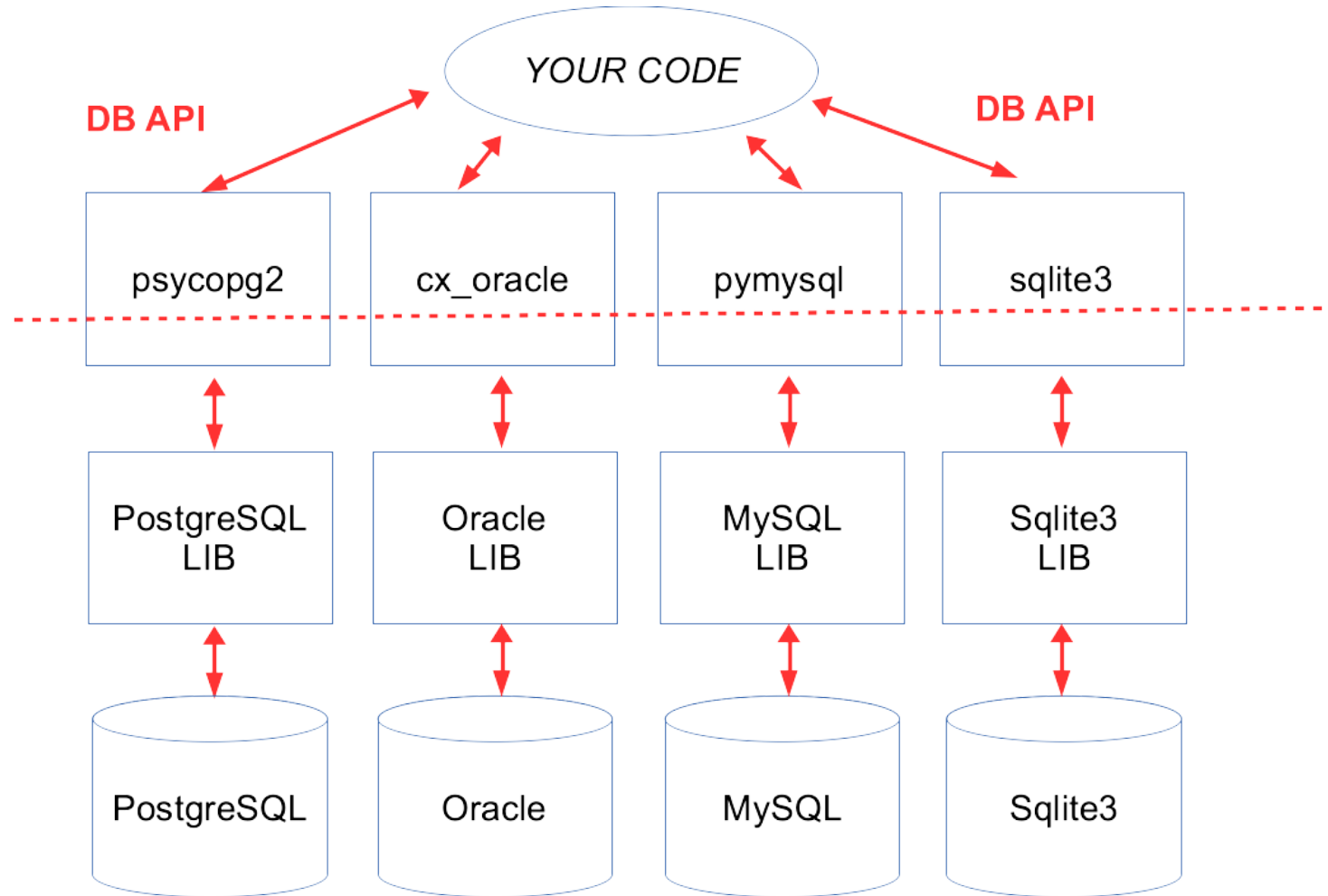
Iterators returned by

- `open()`
- `enumerate()`
- `DICT.items()`
- `zip()`
- `reversed()`
- *generator expression or function*
- *iterator class*

Iterators



Python DB Interface



Python 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()`

Creating Variables

```
x = 5
```

Creating Variables

`x = 5`



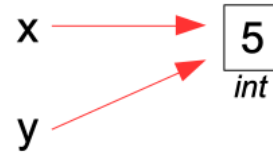
Creating Variables

```
x = 5  
y = x
```



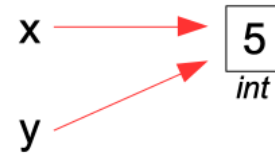
Creating Variables

```
x = 5  
y = x
```



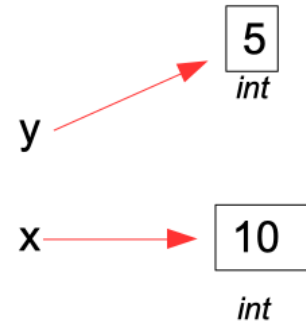
Creating Variables

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



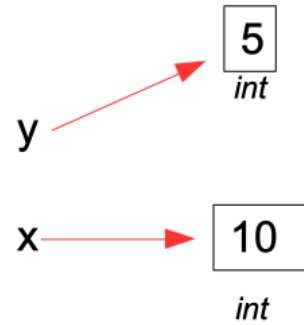
Creating Variables

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



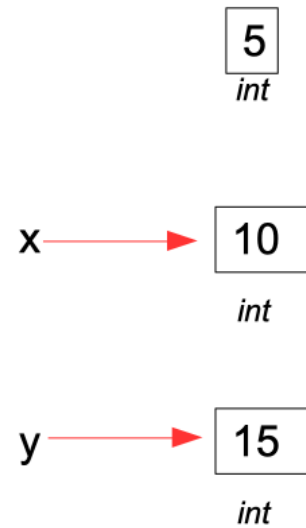
Creating Variables

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



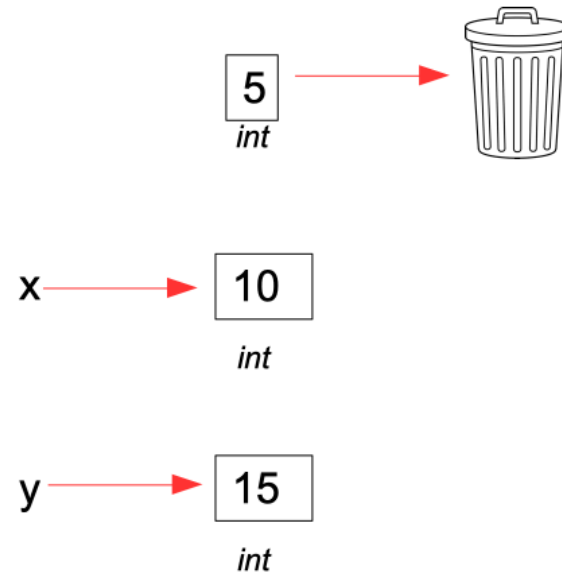
Creating Variables

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



Creating Variables

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



Variable Scope

builtin

`print()`
`len()`

global

`COUNT = 0`
`LIMIT = 1`

local

```
def spam(ham):  
    eggs = 5  
    print(eggs)  
    print(COUNT)
```

Variable scope

```
ALPHA = 10

def spam(beta):
    gamma = 20
    print(ALPHA)
    print(beta)
    print(gamma)

spam(1234)
```

BUILTIN

GLOBAL

LOCAL

Lists vs Tuples

Lists	Tuples
Dynamic array	Collection of related fields
Mutable/unhashable	Immutable/hashable
Position doesn't matter	Position matters
Use case: iterating	Use case: indexing or unpacking
"ARRAY"	"STRUCT" or "RECORD"

A Myth

Tuples are just read-only lists

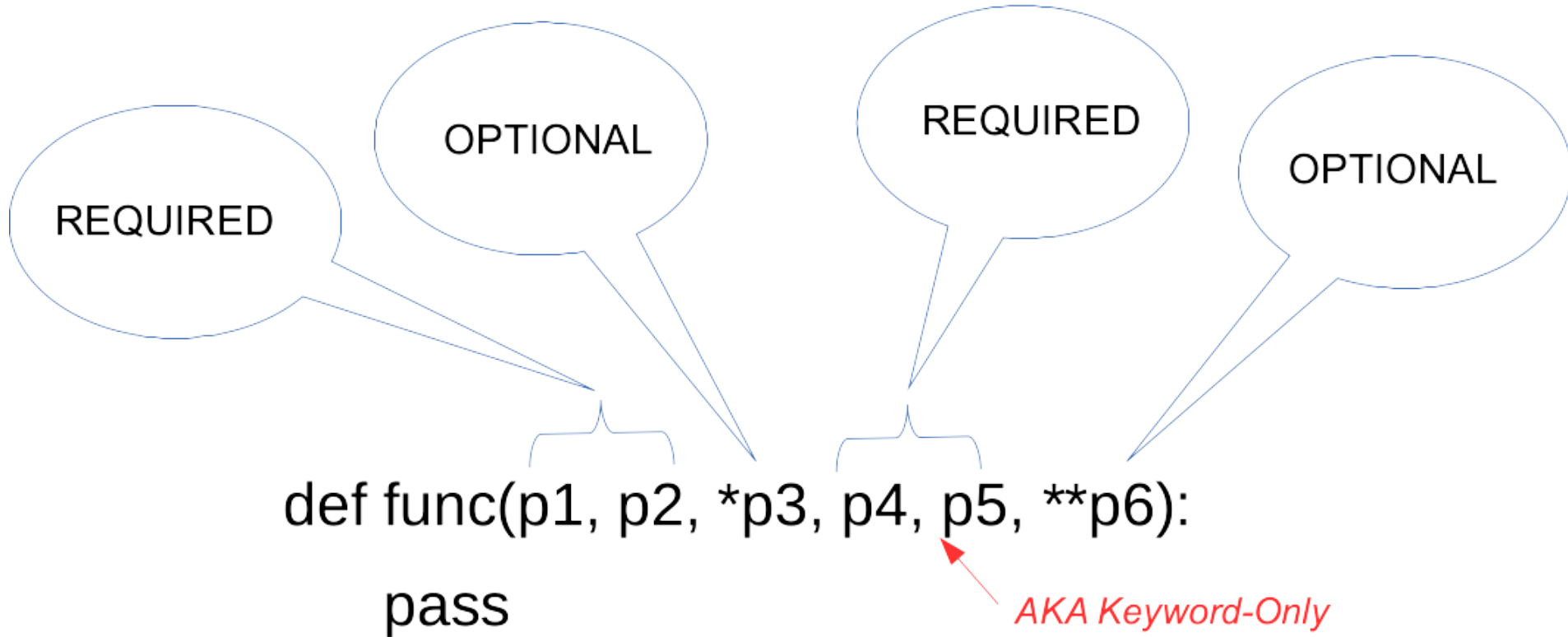
Tuple alternatives

- Standard library
 - namedtuple
 - dataclass
- Third-party
 - attrs
 - Pydantic

Function parameters

POSITIONAL

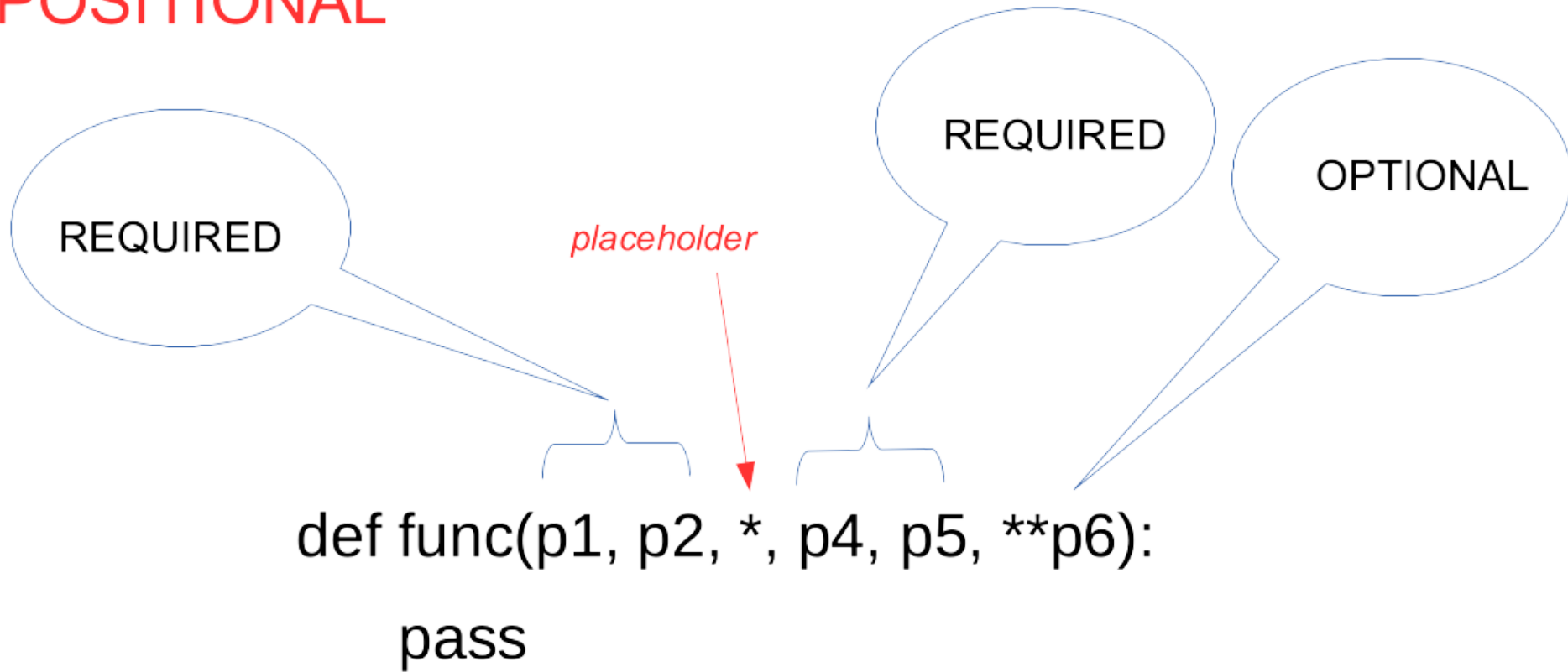
NAMED



Function parameters

POSITIONAL

NAMED



Concurrency

