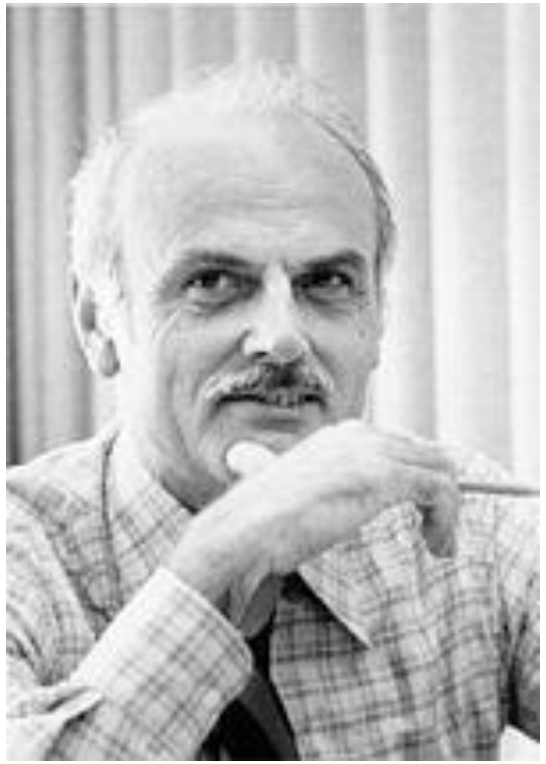
DS3500

Advanced Programming with Data

John Rachlin
Northeastern University

Database Programming

Codd, 1970.



Edgar Frank Codd
1923-2003

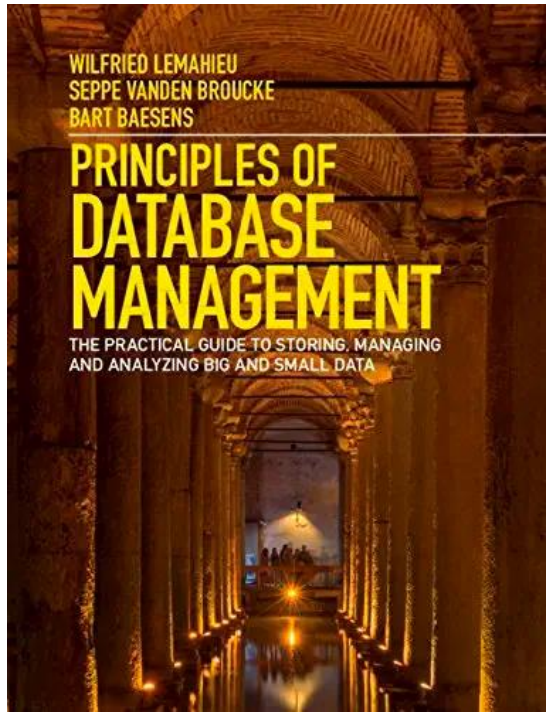
Information Retrieval

A Relational Model of Data for Large Shared Data Banks

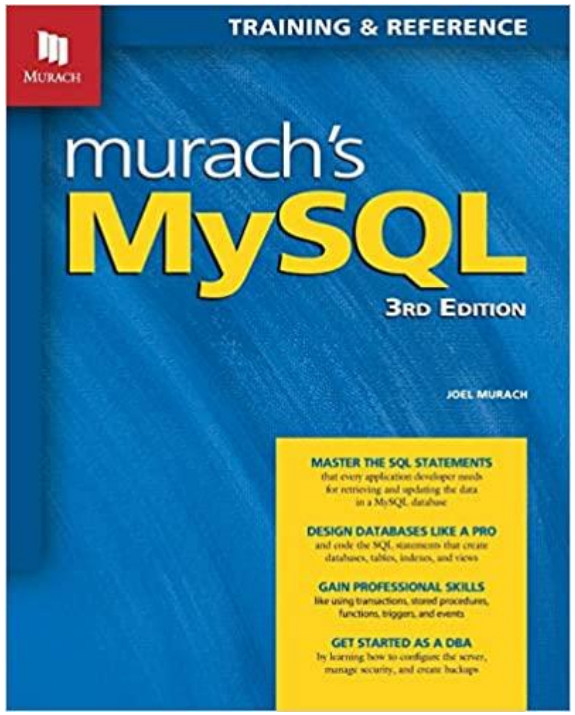
E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Textbooks



Principles of Database Management (1ed)
by LeMahieu, vanden Broucke and Baesens
Recommended



Murach's My SQL by Joel Murach (3rd edition)
(2nd ed OK)
Highly recommended

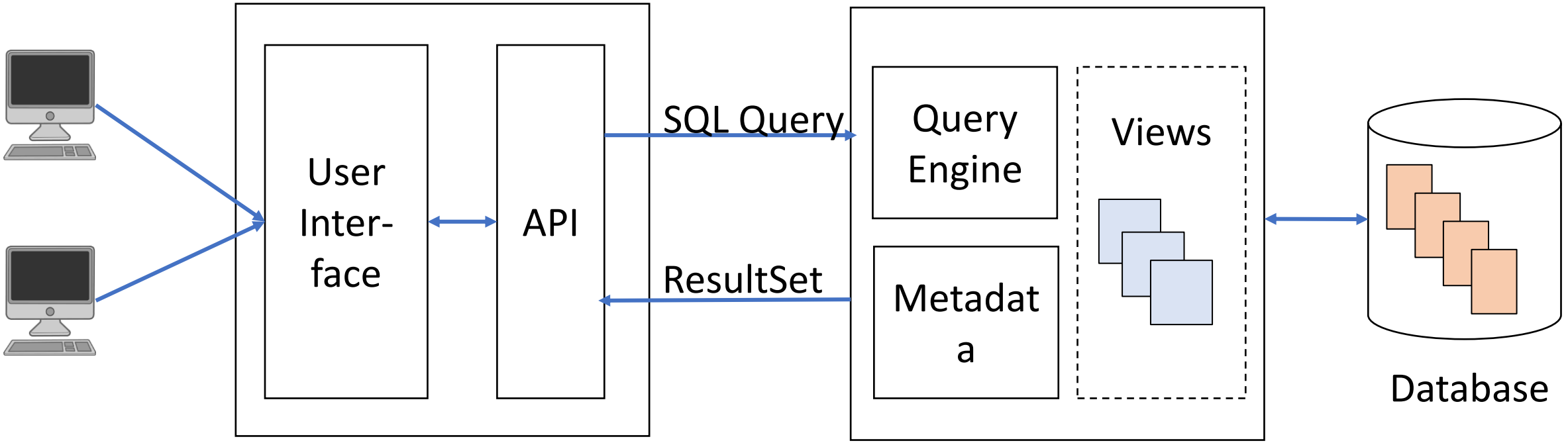
What is a Database?

- Shared collection of **logically related** data (and a description of this data), designed to meet the information needs of an organization.
- System **catalog** (metadata) provides description of data to enable program–data independence.
- Logically related data comprises entities, attributes, and relationships of an organization's information.
- In a relational database, the data is represented as a **collection of tables**.

Database Management System (DBMS)

- A software system that enables users to define, create, maintain, and control access to the database.
- (Database) application program: a computer program that interacts with database by issuing an appropriate request (SQL statement) to the DBMS.
- A *Relational* Database Management System (RDBMS or RDB) uses the relational model: i.e., it uses tables to represent entities. This remains the dominant model in industry today, but other approaches have emerged.

Database Application



(Usually running on a separate server)

A Simple Relational Data Model

Name	Salary	Branch
John White	30000	22 Deer Rd, London
Ann Beech	12000	163 Main St, Glasgow
David Ford	18000	163 Main St, Glasgow
Mary Howe	9000	16 Argyll St, Aberdeen
Susan Brand	24000	163 Main St. Glasgow
Julie Lee	9000	22 Deer Rd, London

Discussion: What are the disadvantages of this design?

A Better Relational Data Model

Branch

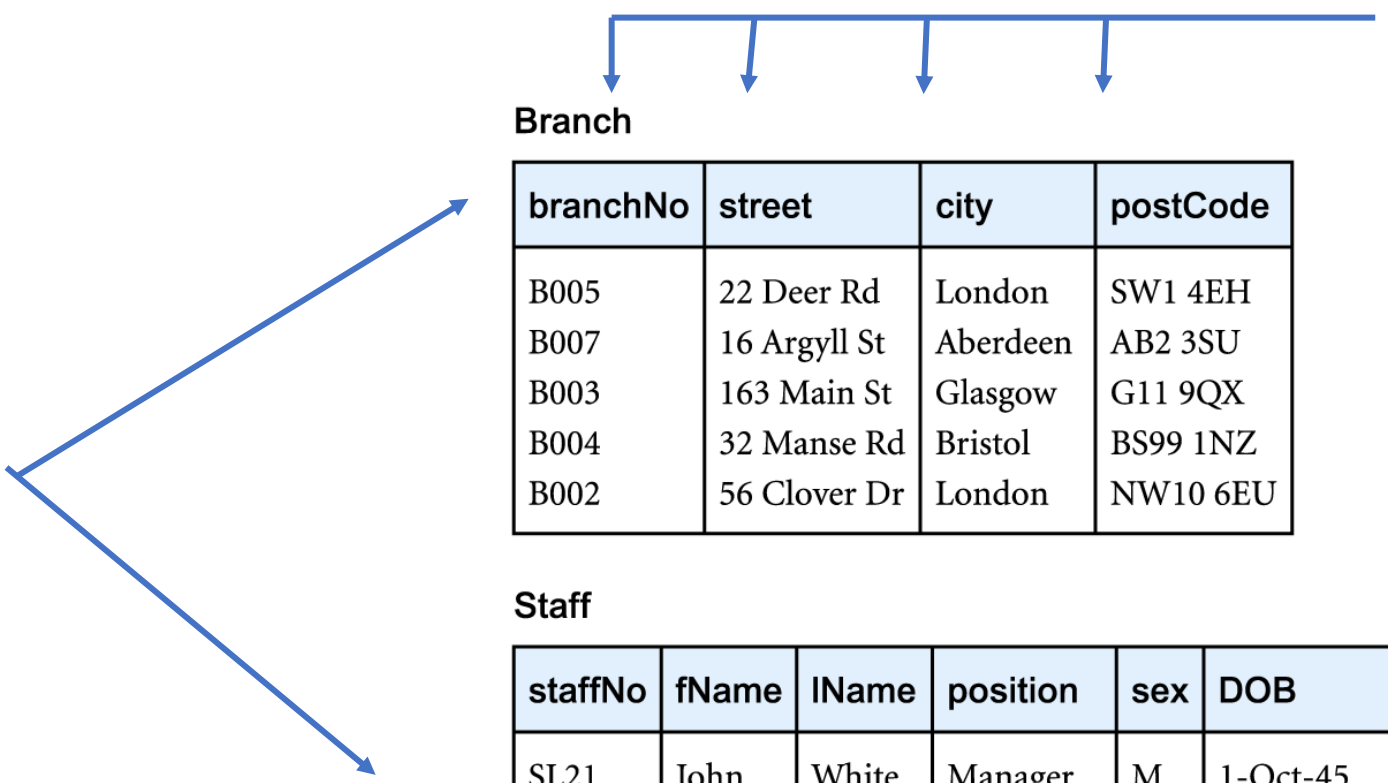
branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relational Data Model

Entities



Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

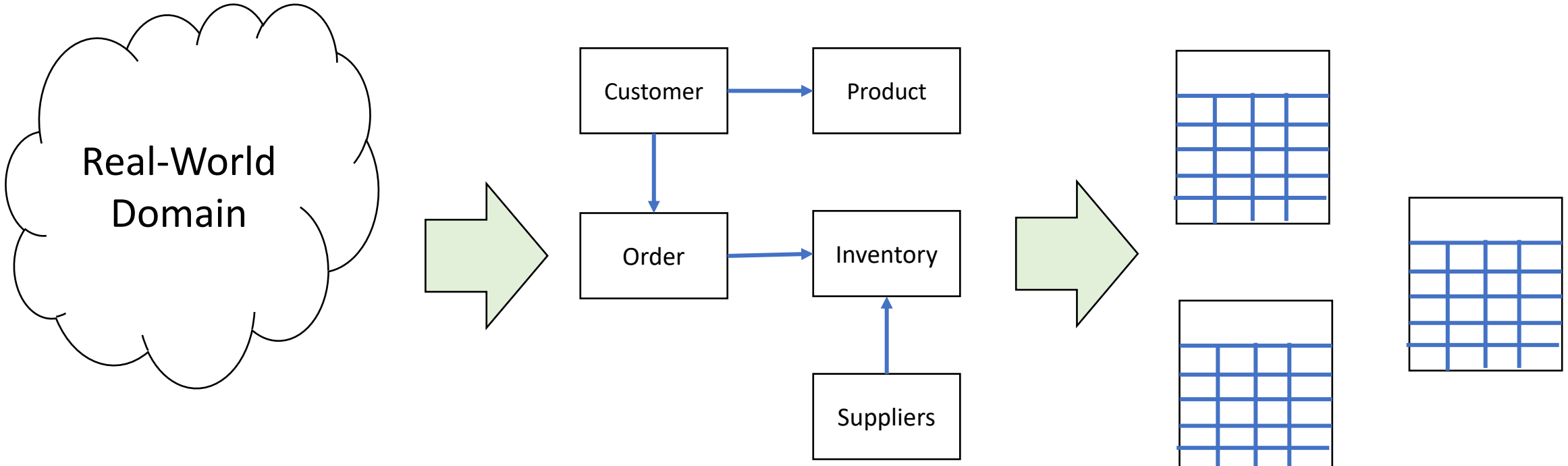
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

A column or ***attribute*** or **field** providing more details about the particular entity.

- Name
- Datatype
- Other attributes

A row or ***record*** describing one instance of the entity

The Modeling Process



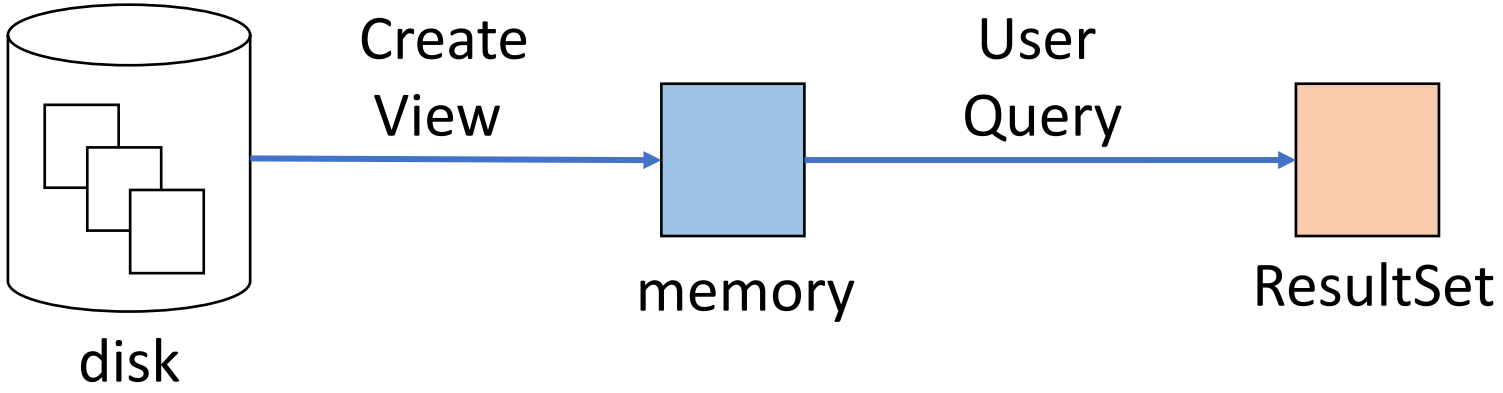
- **Entities** (things, events, concepts)
- **Attributes** (annotations on entities)
- **Relationships** (connections or associations between entities)

Entity-Relationship Diagram
(A *conceptual* data model)

Relational Data Model
(A *logical* data model)

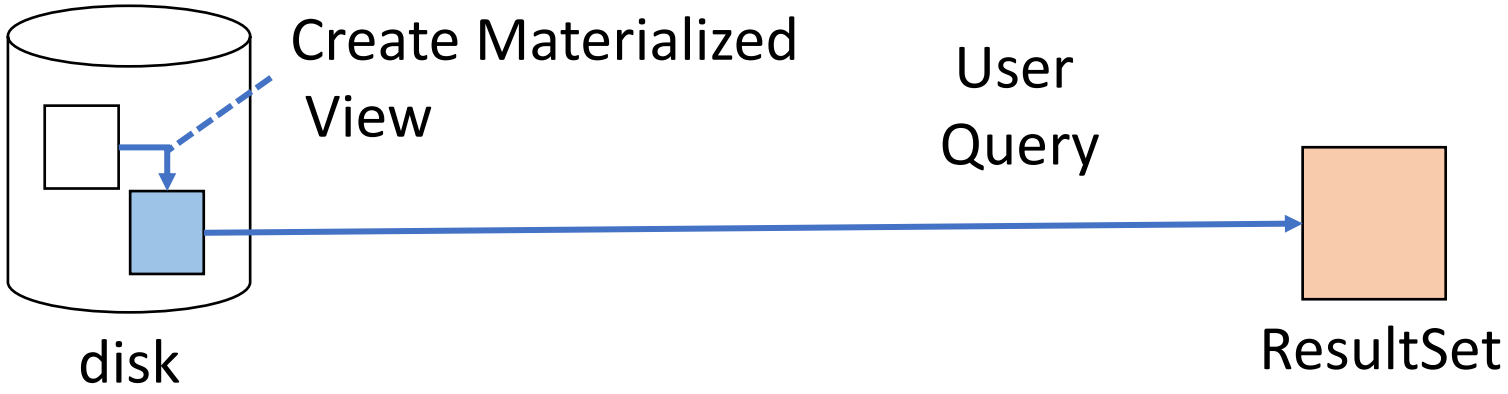
How Views Work

Standard View (MySQL)



- View is a *virtual* table
- Always up-to-date
- Every access regenerates the create-view operation in the background

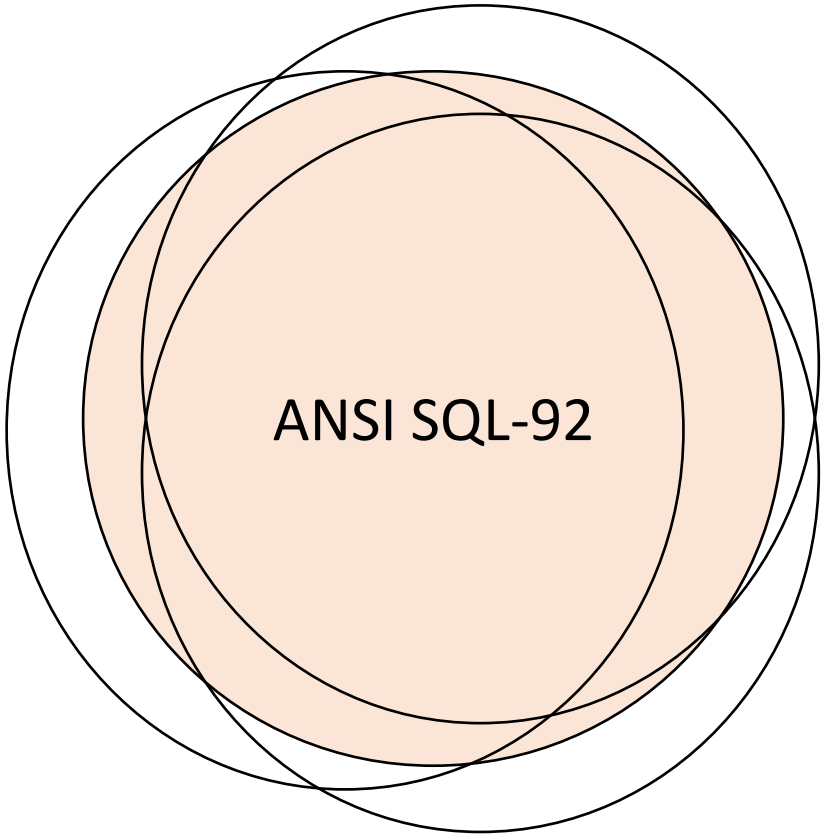
Materialized View (Some DBMSs)



- View is a *physical* table on disk
- View is not always up-to-date: Needs to be periodically synced.
- Subsequent user queries are faster, however because creation is done only once.

SQL: The language of databases

- SQL = Structured Query Language
- Pronounced *either* S.Q.L. or *See*-quel (both are perfectly acceptable.)
- The language was *standardized* over time (SQL-86, SQL-89, SQL-92) but most vendors do not implement the standard completely or they introduce vendor-specific features / dialects.



One standard, many dialects

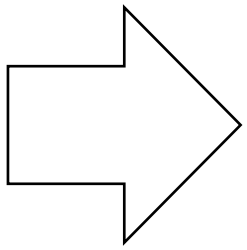
What is *structured* about SQL?

Simplified syntax for selecting data:

```
SELECT  <Columns to retrieve>
FROM    <Source tables>
WHERE   <Conditions on individual rows>
ORDER BY  <Sorting criteria>
```

An example of how SQL is a declarative language

```
SELECT  invoice_id, invoice_total
FROM    invoices
WHERE   invoice_total > 100.0
ORDER BY  invoice_total
```



invoice_id	invoice_total
1	125.00
4	2199.99

Procedural Languages for DB are numerous

Source	Common name	Full name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase / Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
IBM Netezza	NZPLSQL[18]	(based on Postgres PL/pgSQL)
Microsoft / Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MonetDB	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
NuoDB	SSP	Starkey Stored Procedures
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language
SAP	SAP HANA	SQL Script

Source:
https://en.wikipedia.org/wiki/SQL

Most common datatypes

- CHAR(n), VARCHAR(n)
- INT (SIGNED or UNSIGNED)
- FLOAT, DOUBLE, DECIMAL(m,d)
- DATE, TIME, DATETIME
- BLOB, TEXT (CLOB)
- ENUM

Creating Table Examples

A very basic table with just four columns.

```
USE demo;

DROP TABLE IF EXISTS users;

CREATE TABLE users (
    user_id INT,
    username VARCHAR(20),
    dob DATE,
    class ENUM ('Basic', 'Premium')
);
```

What is a PRIMARY KEY?

- A **PRIMARY KEY** is a value or set of values that uniquely identifies a single row in your database table.
- Primary keys cannot be NULL (empty / unassigned)
- Primary keys must be unique
- Primary keys are usually INTs but they could be other datatypes

A Relational Data Model

Branch

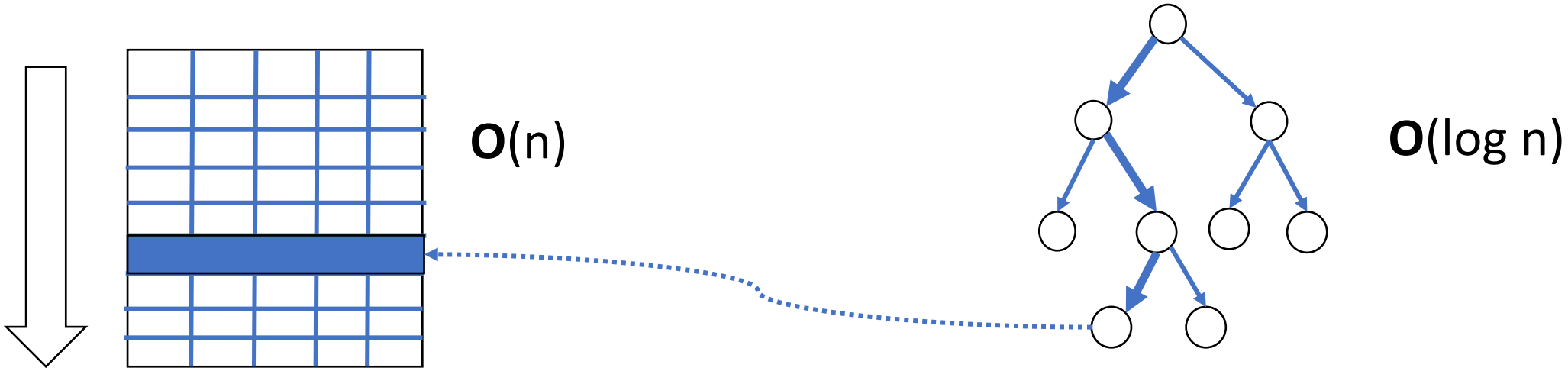
branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PRIMARY KEY: What's it good for?

The database management system builds an **index** for the table around the primary key so that, given the value of the primary key, it can look up the row containing the rest of the values very quickly.



O(n)

O(log n)

Without a primary Key
we must scan row-by-row
looking for the right value

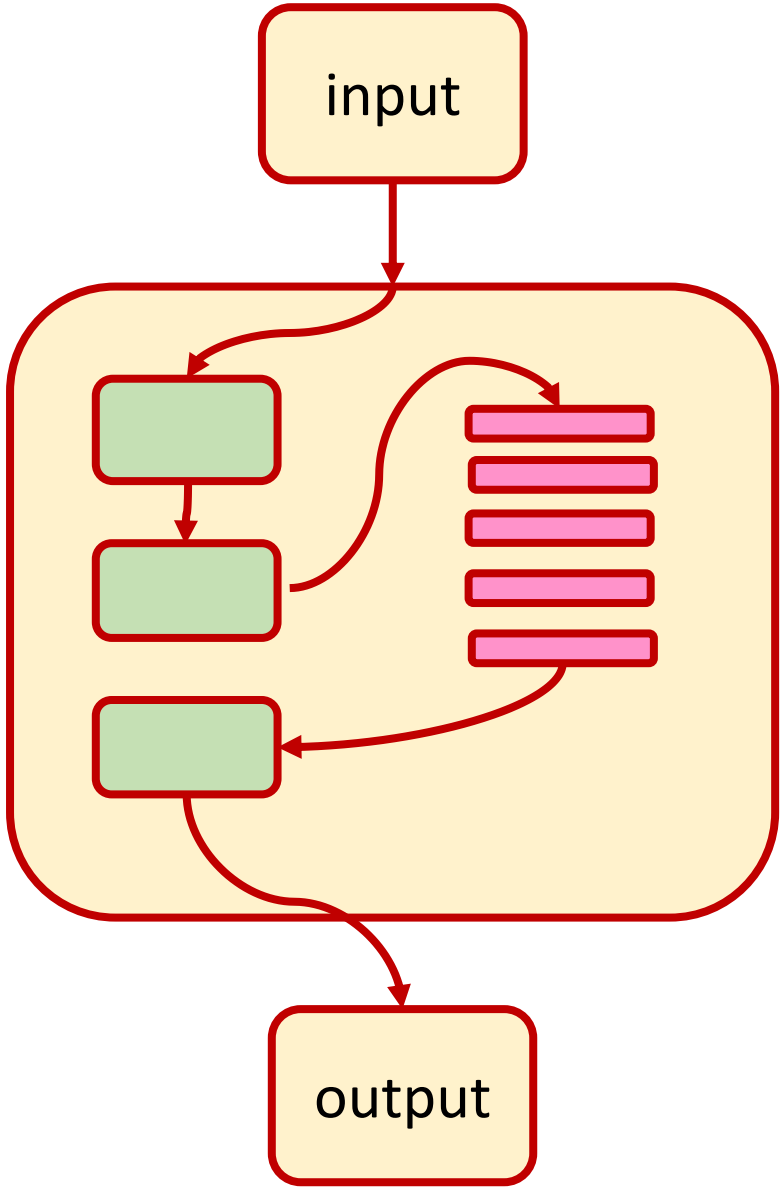
With Primary Key
we look up the the row location
by searching through a tree.

Object-Oriented Programming (OOP)

The procedural paradigm

Programs are *recipes*: a series of statements that transform our data into visualizations and insight.

In procedural programming, we manage complexity by being **modular**: adhering to top-down programming practices that break down difficult tasks into a sequence of more manageable sub-tasks.



Recipes are a powerful metaphor for data science!

start with raw data

load data

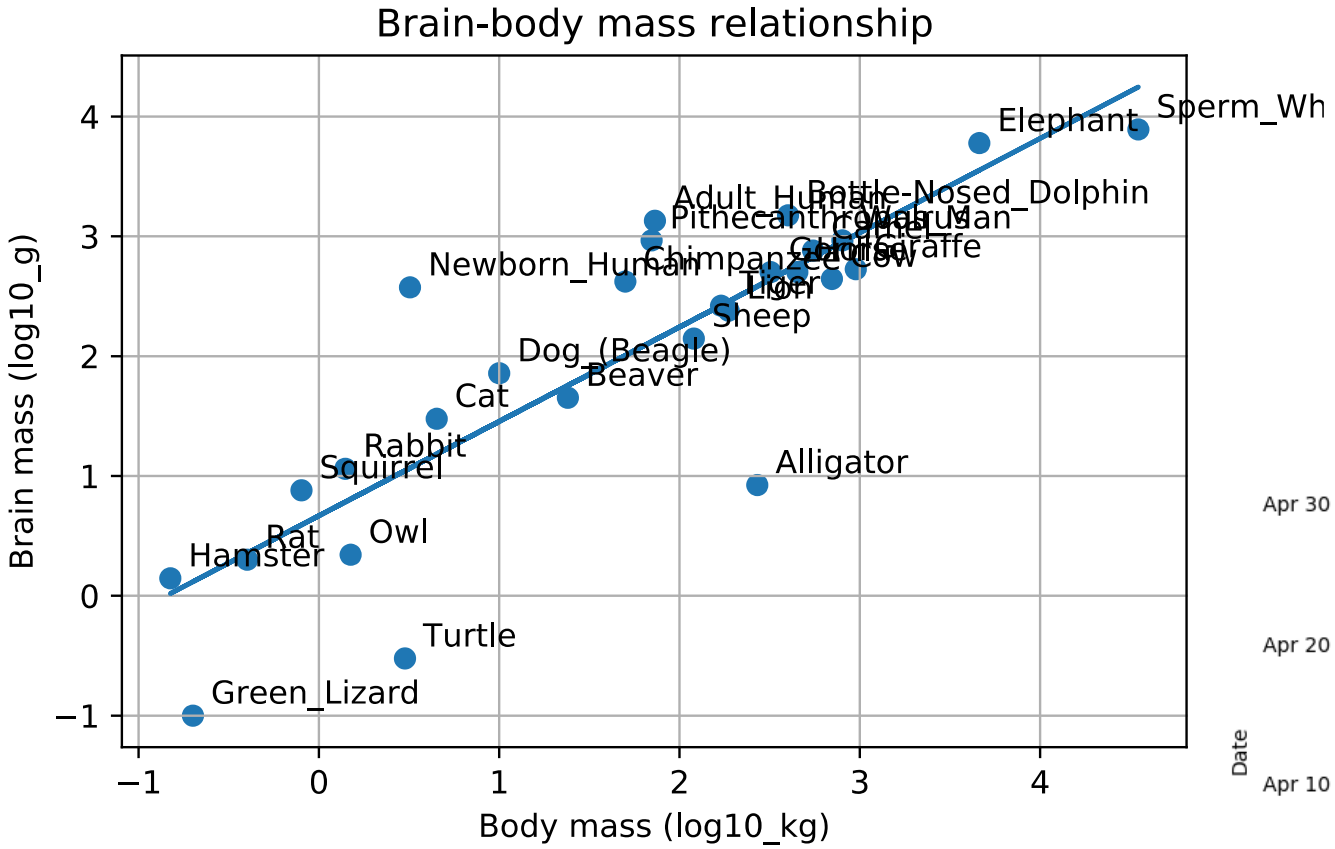
data refinement (munging)

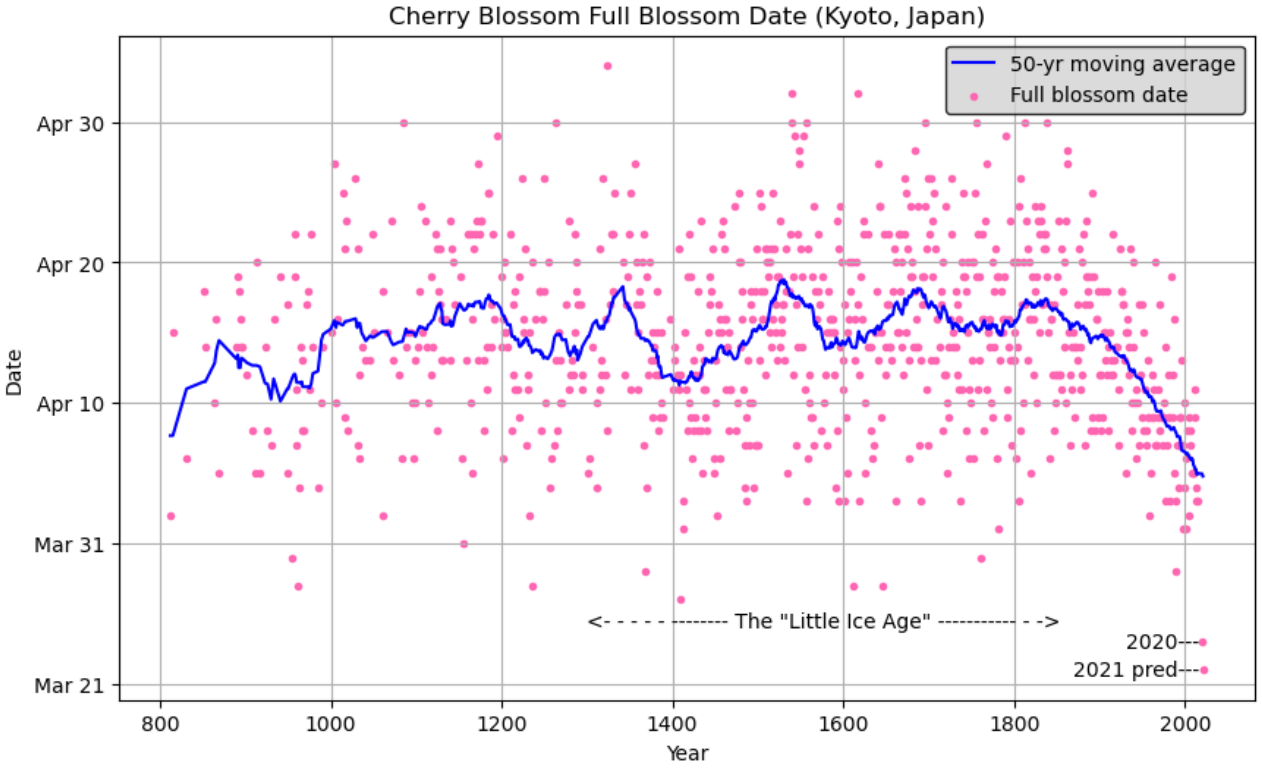
build data structures

processing and visualization

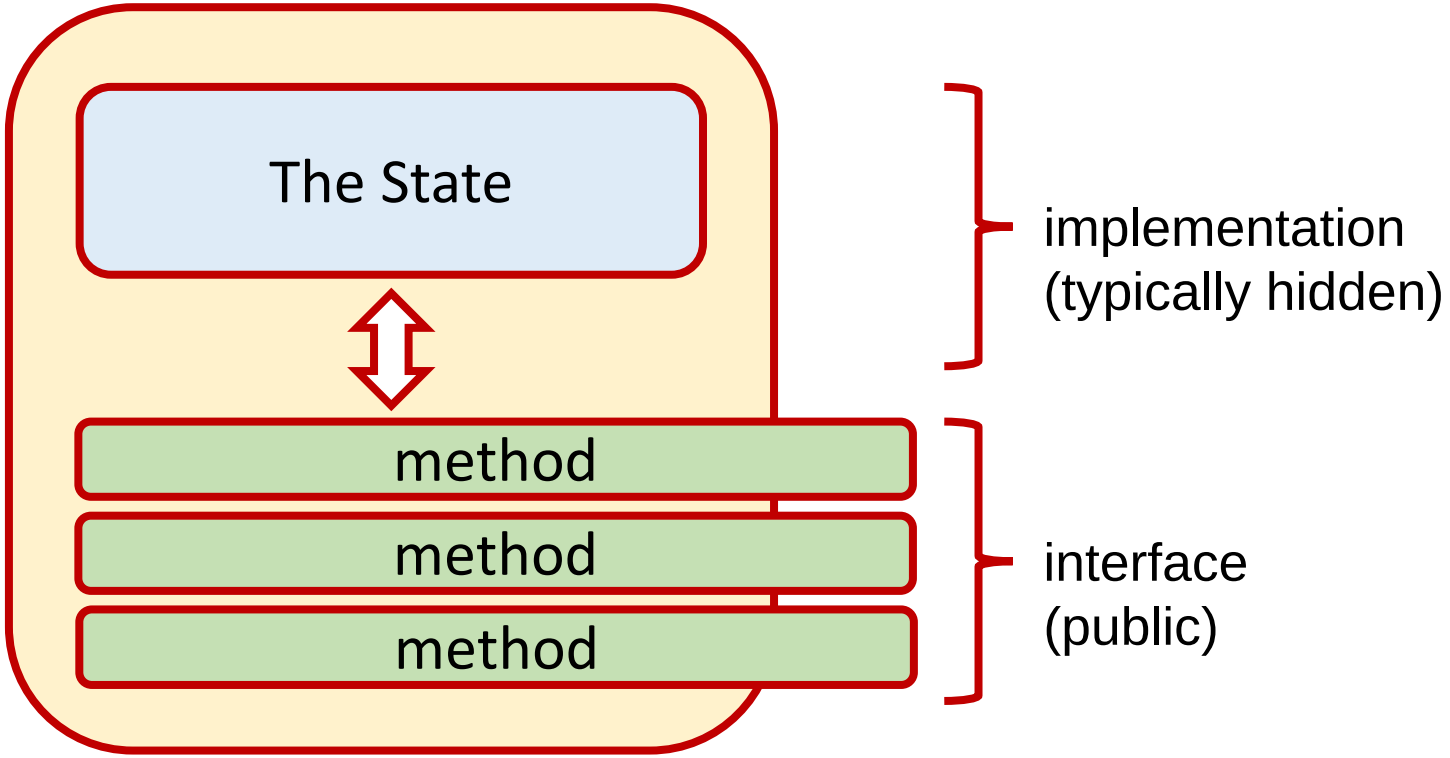
don’t forget to label your axes!

…and it's adequate for a wide range of use-cases!





The object paradigm (more later!!)



Objects have fields and attributes that constitute the **state** of the object.

The state is accessed and modified through various **methods** that constitute the object's **interface**.

The layout and organization of the state (i.e., the **implementation**) is usually shielded from the user.

Beautiful Code (and also the DS2500 HW grading rubric!)

- **Modular**: The program is broken down into small reusable functions that are more easily implemented, tested, and verified.

- **Reusable**: Functions are parameterized for maximum flexibility. Functions that go together are packaged together in a file. You aren't re-inventing the wheel.

- **Readable**: Your code is well-documented so that another programmer can understand your thought process. (Your future self will thank you!) Every function or method is documented including a specification of expected inputs and outputs. Don't go overboard!

```
x = 5 # assign 5 to x - not a useful comment!!
```

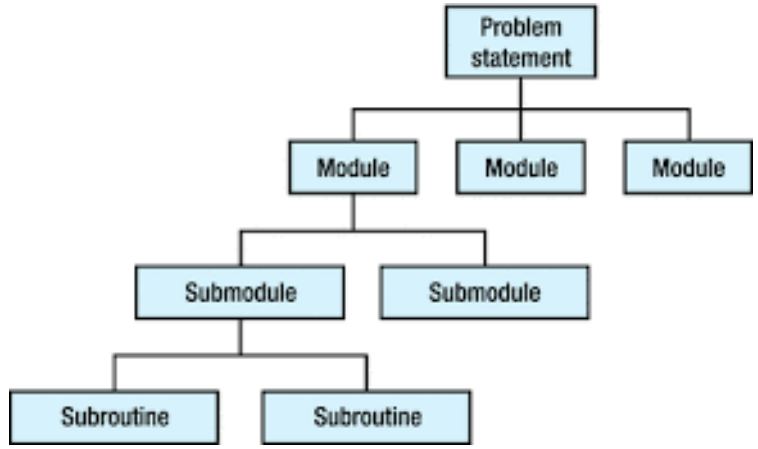
- **Efficient**: Uses data-structures appropriately. Implements algorithms that meet performance requirements without being unnecessarily complex.

- **Correct**: Satisfies the requirements of the project or assignment.

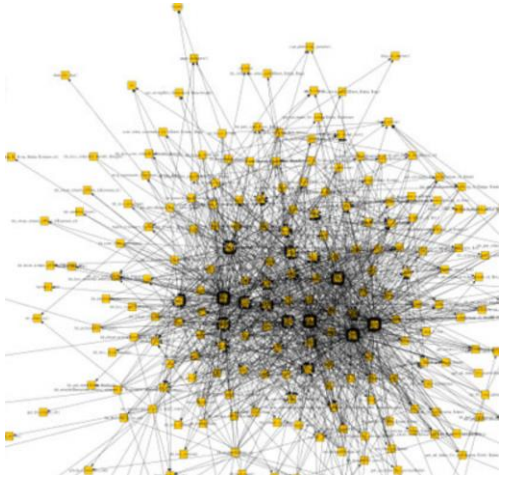
Starting with modular design

- **Modular programming** is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable **modules**, such that each contains everything necessary to execute only one aspect of the desired functionality. (Wikipedia)
- Modules are usually small (<100 lines of code)
- Modules are *re-useable* (if well-written)
- Modular programming limits cross-dependencies

modular code



Spaghetti code



Regular (Traditional) Packages in Python

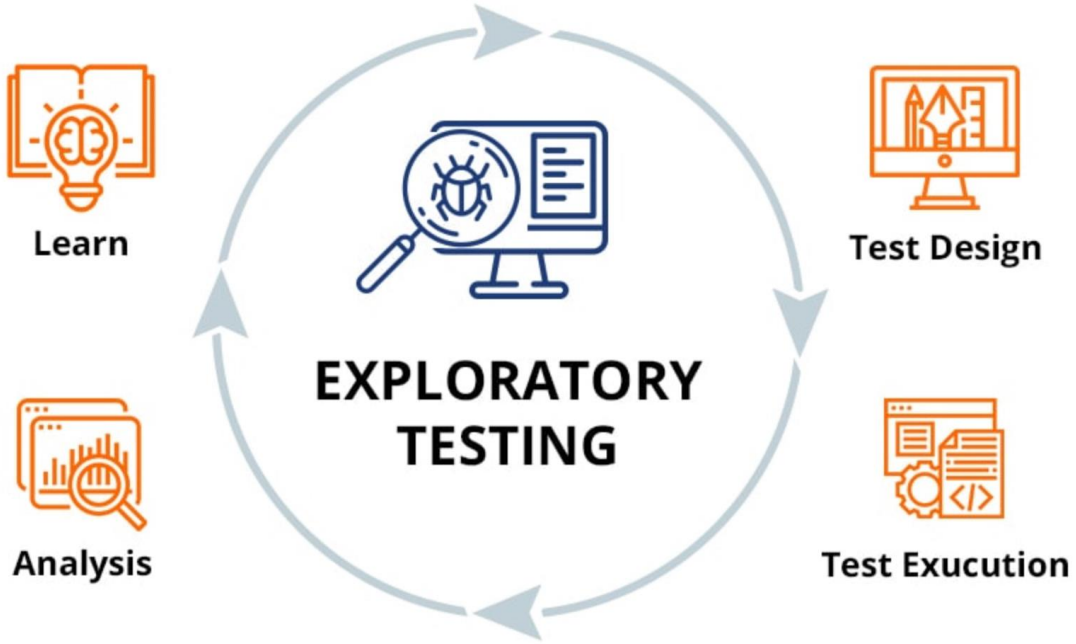
- Regular packages are traditional packages as they existed in Python 3.2 and earlier.
- A regular package is typically implemented as a directory containing an __init__.py file. When a regular package is imported, this __init__.py file is implicitly executed, and the objects it defines are bound to names in the package's namespace.
- The __init__.py file can contain the same Python code that any other module can contain, and Python will add some additional attributes to the module when it is imported.

```
parent/
    __init__.py
    one/
        __init__.py
    two/
        __init__.py
    three/
        __init__.py
```

Importing `parent.one` will implicitly execute `parent/__init__.py` and `parent/one/__init__.py`. Subsequent imports of `parent.two` or `parent.three` will execute `parent/two/__init__.py` and `parent/three/__init__.py` respectively.

Exploratory (Manual) Testing

- Running a program against an unstructured (possibly incomplete) set of cases in order to validate a list of features is known as *exploratory* testing. This is *testing without a plan.*
- Every change requires manual re-testing to make sure nothing has broken.
- Dependent on the skill of the tester to ensure complete coverage.



Source: https://blog.qatestlab.com/2011/02/23/exploratory-testing-advantages-and-disadvantages/

Automated Testing

- Automated Testing involves the execution of a *test plan.*
- Python has built-in tools to support this.
- A *test runner* executes a series of pre-defined tests and compares results against expectations.
- **unit test:** Testing a single component, function, or module.
- **integration test**: Testing multiple components and their interactions.

Exploratory vs. Automated Testing

	EXPLORATORY TESTING	AUTOMATED TESTING
Documentation	not much need of documentation	documentation is essential
Lead-in time	little or no lead-in time	lead-in time is required
Test scripts	not much efforts for test script creation	test scripts are important
Test coverage	no measurement	can be measured
Based on	tester's understanding of a product	specifications
Reproducibility	low	high

Source: https://blog.qatestlab.com/2011/02/23/exploratory-testing-advantages-and-disadvantages/

pytest: A framework for python testing (pytest.org)

Search Go

Table Of Contents

Home
Install

Installation and Getting Started

Pythons: Python 3.6, 3.7, 3.8, 3.9, PyPy3

Platforms: Linux and Windows

PyPI package name: pytest

Documentation as PDF: download latest

`pytest` is a framework that makes building simple and scalable tests easy. Tests are expressive and readable—no boilerplate code required. Get started in minutes with a small unit test or complex functional test for your application or library.

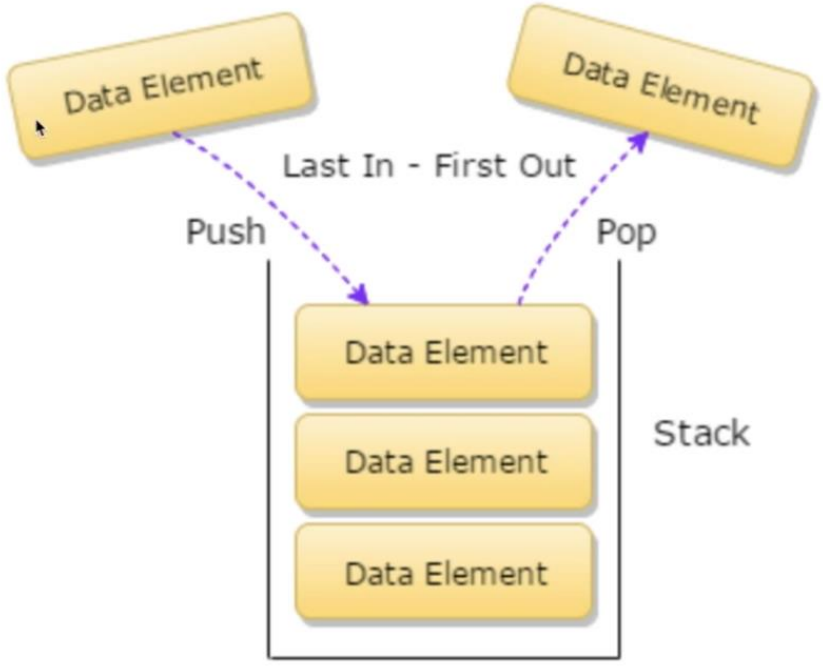
```
$ conda install pytest
$ conda install pytest-cov
$ conda list | grep pytest
```

Stacks and Queues

Stack (First In, Last out)

push: Add an element to the top of the stack

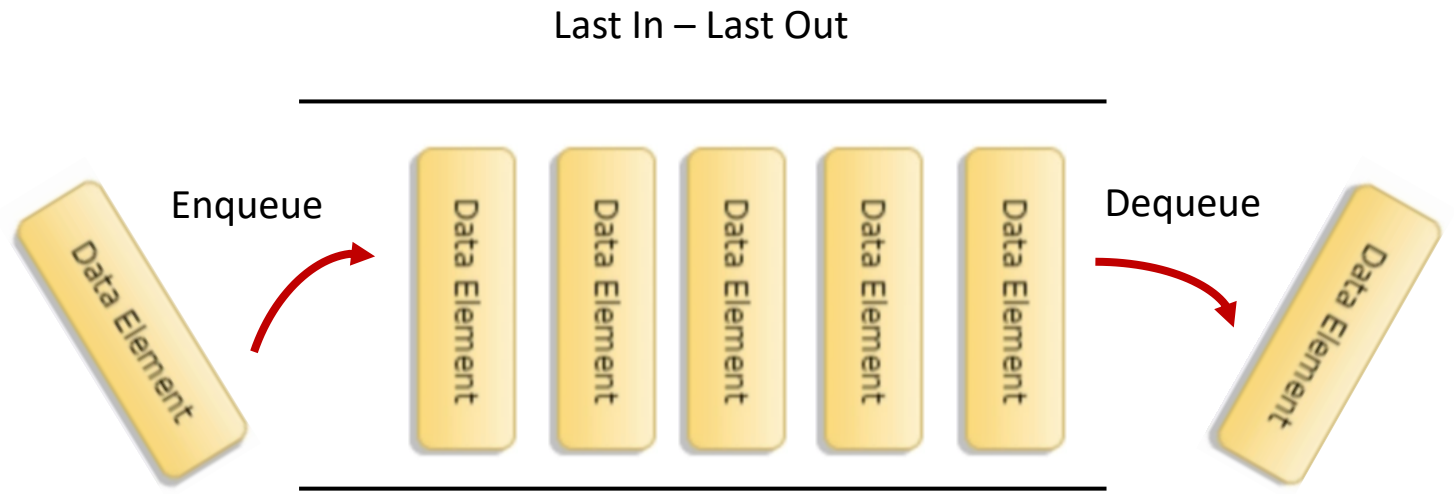
pop: Remove an element from the top of the stack



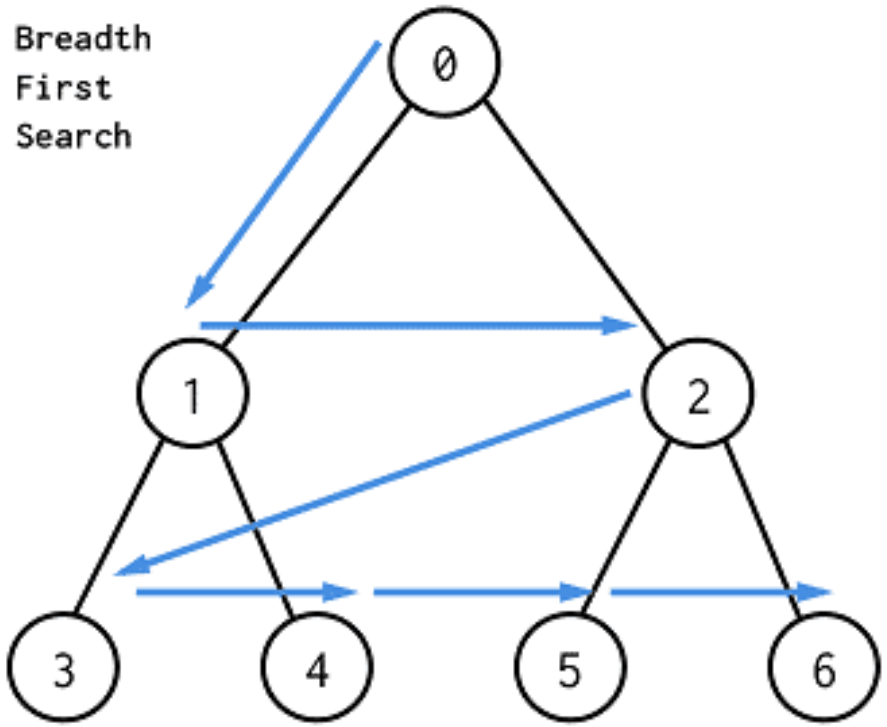
Queue (First In, First out)

enqueue: Add an element to the front of the queue

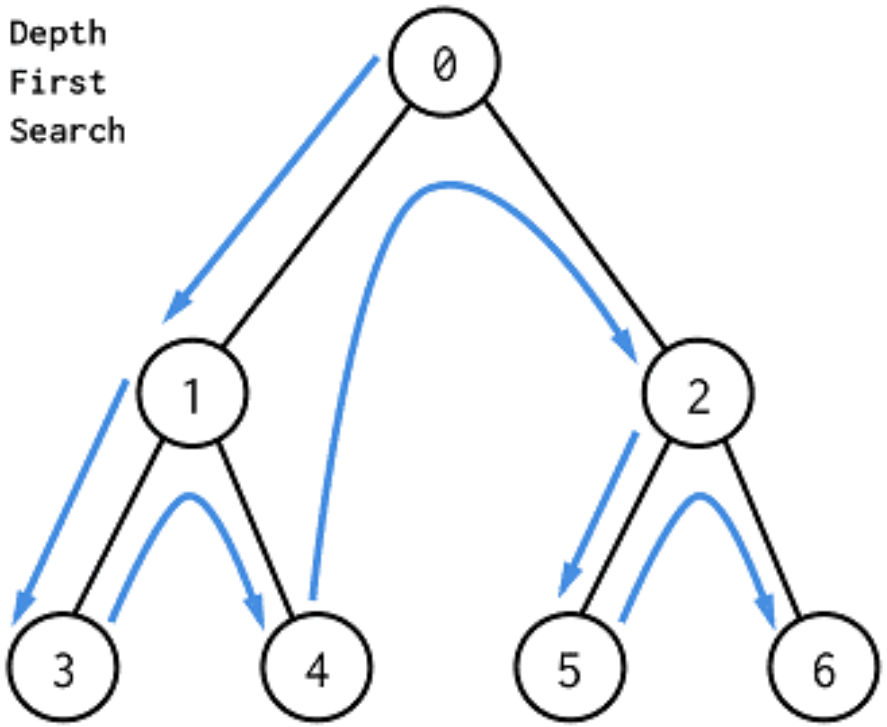
dequeue: Remove an element from the end of the queue



BFS vs. DFS

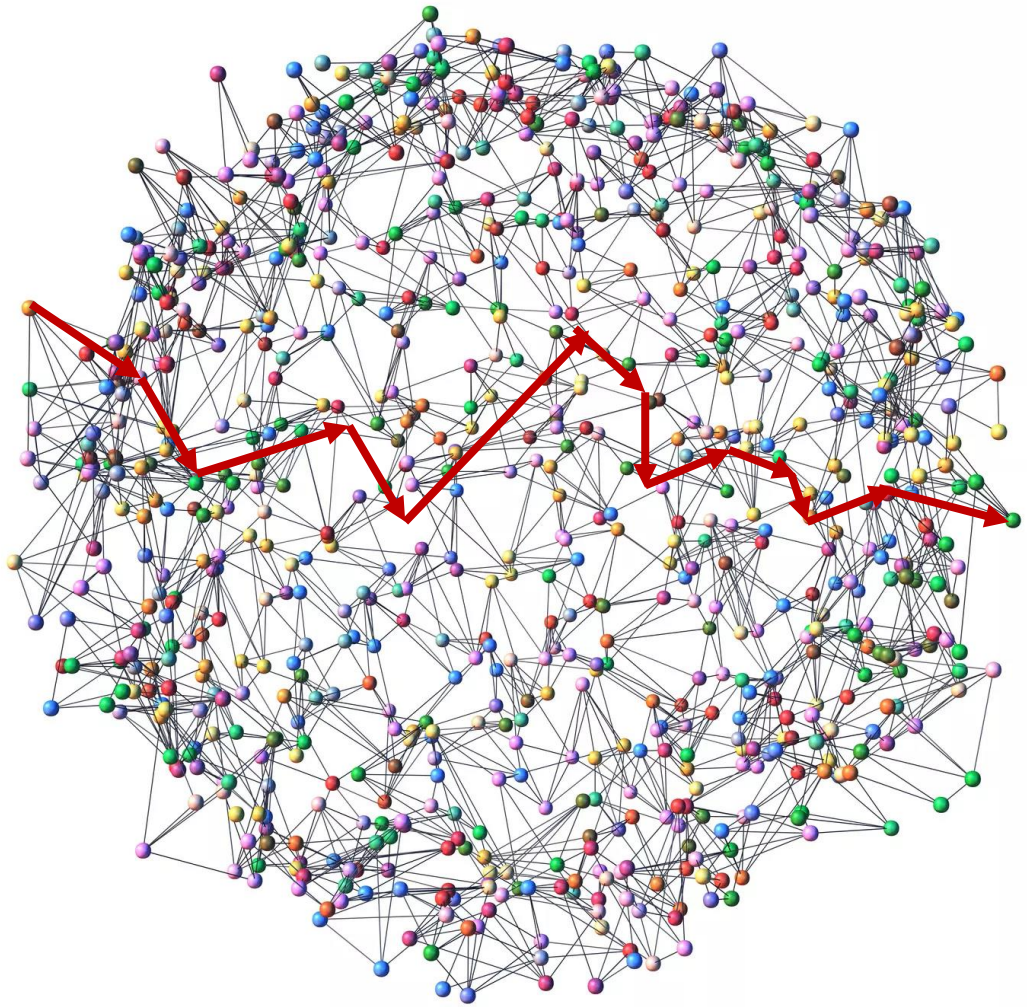


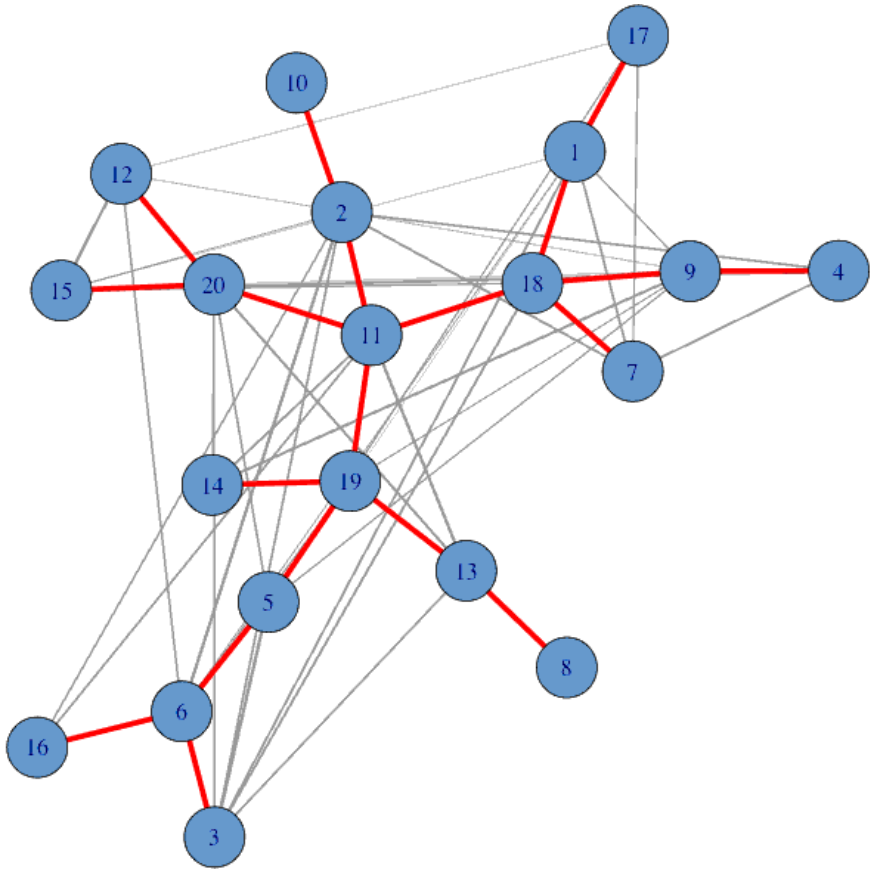
Track visited nodes
in a QUEUE



Track visited nodes
in a STACK

Graph Traversal, Finding Paths and Spanning Trees





Directory Structure

pytest
- dstruct
  - queue.py
  - stack.py
- tests
  - test_queue.py
  - test_stack.py

Project Name
- *package*
  - file1.py
  - file2.py
- *test-directory*
  - test_file1.py
  - test_file2.py

Running pytest

Report # of passes and failures only

$ python –m pytest

Report outcome of each test

$ python –m pytest –v

Report code coverage percentages (requires pytest-cov)

$ python –m pytest –v –cov

Report coverage and identify missing lines

$ python -m pytest -v --cov --cov-report term-missing

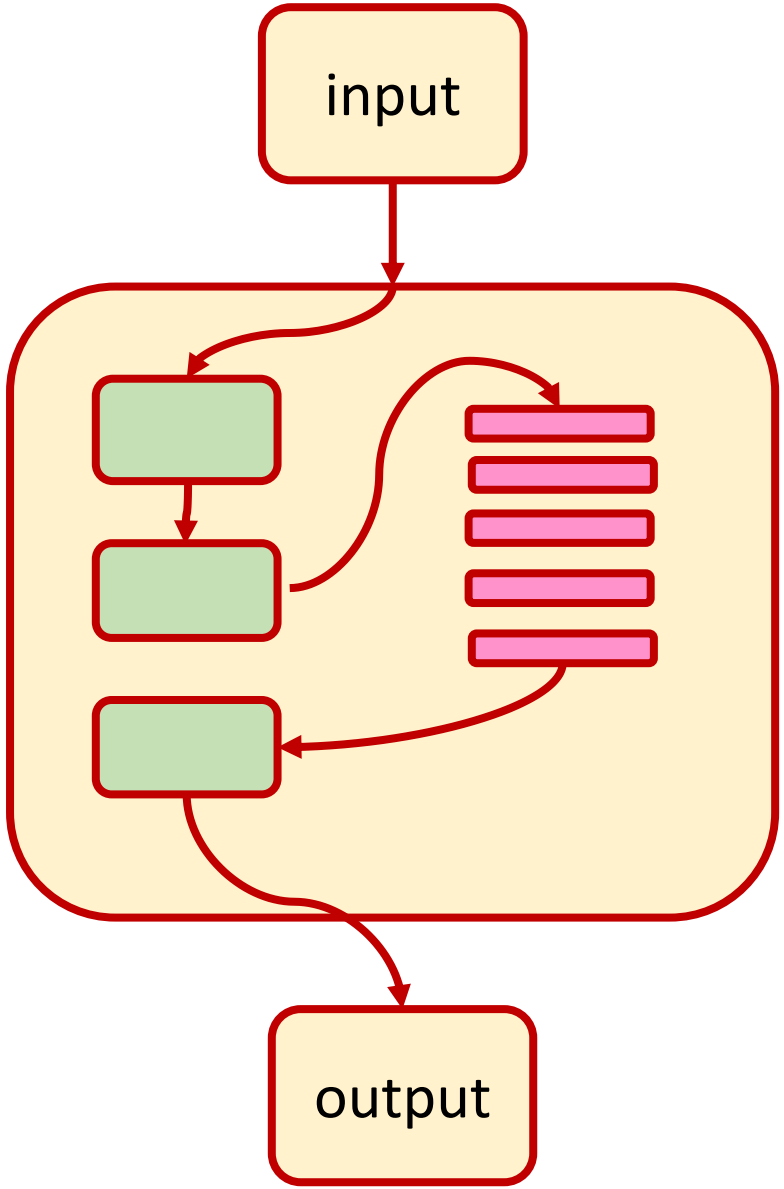
Running pytest

```
============================================================================================================================ 6 passed in 0.03s ===================
(base) [552 uranus:ds3500 ] python -m pytest -v --cov --cov-report term-missing
============================================================================================================================ test session starts =================
platform darwin -- Python 3.8.8, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- /Users/rachlin/apps/anaconda3/bin/python
cachedir: .pytest_cache
rootdir: /Users/rachlin/code/ds3500
plugins: cov-2.12.1, anyio-3.3.0
collected 6 items

ds3500/dstruct/tests/test_queue.py::test_constructor PASSED
ds3500/dstruct/tests/test_queue.py::test_enqueue PASSED
ds3500/dstruct/tests/test_queue.py::test_dequeue PASSED
ds3500/dstruct/tests/test_stack.py::test_constructor PASSED
ds3500/dstruct/tests/test_stack.py::test_push PASSED
ds3500/dstruct/tests/test_stack.py::test_pop PASSED

---------- coverage: platform darwin, python 3.8.8-final-0 -----------
Name                                 Stmts   Miss  Cover   Missing
------------------------------------------------------------------
ds3500/dstruct/queue.py                 17      0   100%
ds3500/dstruct/stack.py                 15      0   100%
ds3500/dstruct/tests/test_queue.py      27      0   100%
ds3500/dstruct/tests/test_stack.py      21      0   100%
------------------------------------------------------------------
TOTAL                                   80      0   100%


============================================================================================================================ 6 passed in 0.09s ===================
(base) [553 uranus:ds3500 ]
```

The procedural paradigm

Programs are *recipes*: a series of statements that transform our data into visualizations and insight.

In procedural programming, we manage complexity by being **modular**: adhering to top-down programming practices that break down difficult tasks into a sequence of more manageable sub-tasks.



Recipes are a powerful metaphor for data science!

start with raw data

load data

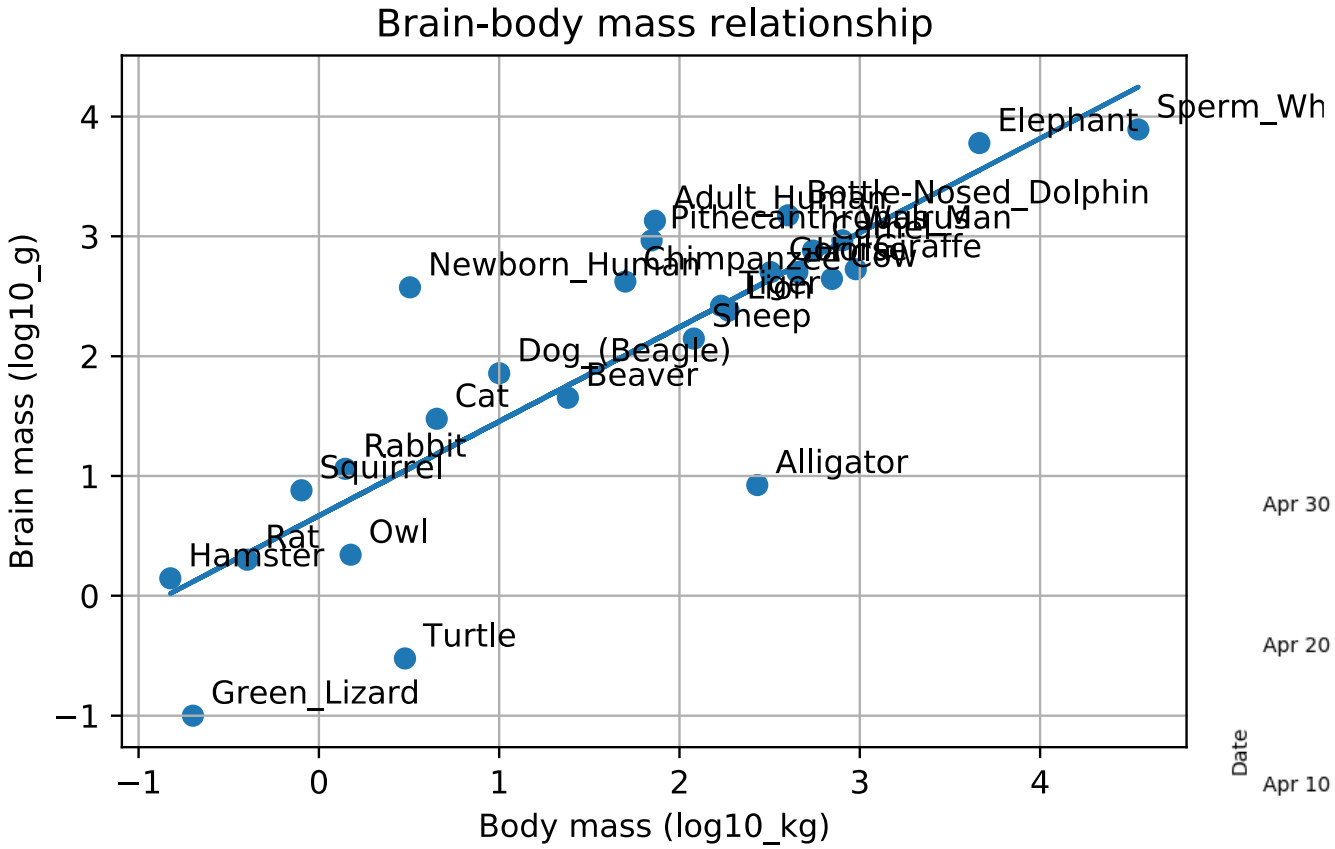
data refinement (munging)

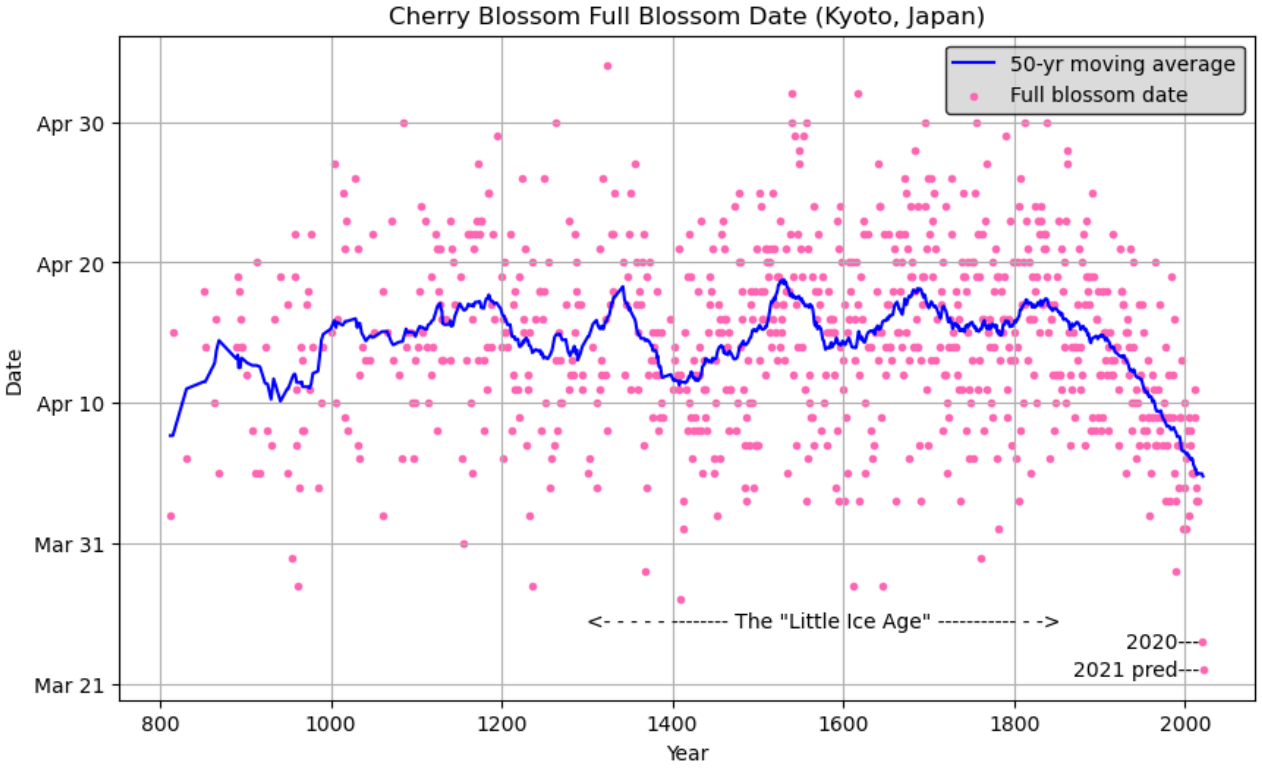
build data structures

processing and visualization

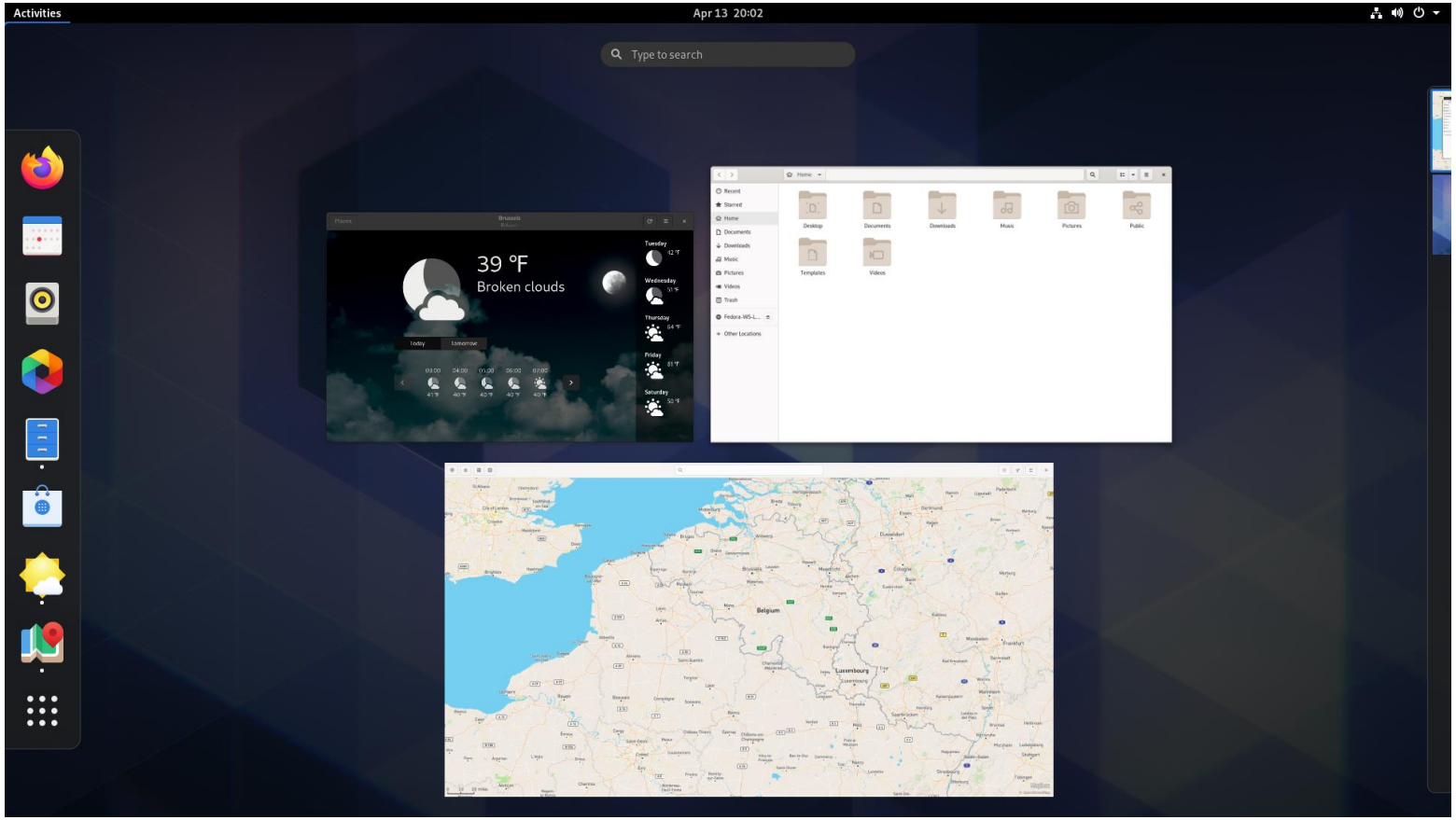
don’t forget to label your axes!

…and it's adequate for a wide range of use-cases!





But consider the Desktop



source: https://www.gnome.org

We **interact** with the desktop **interface** by performing actions on **objects** that each support well-defined **behaviors**.

Objects

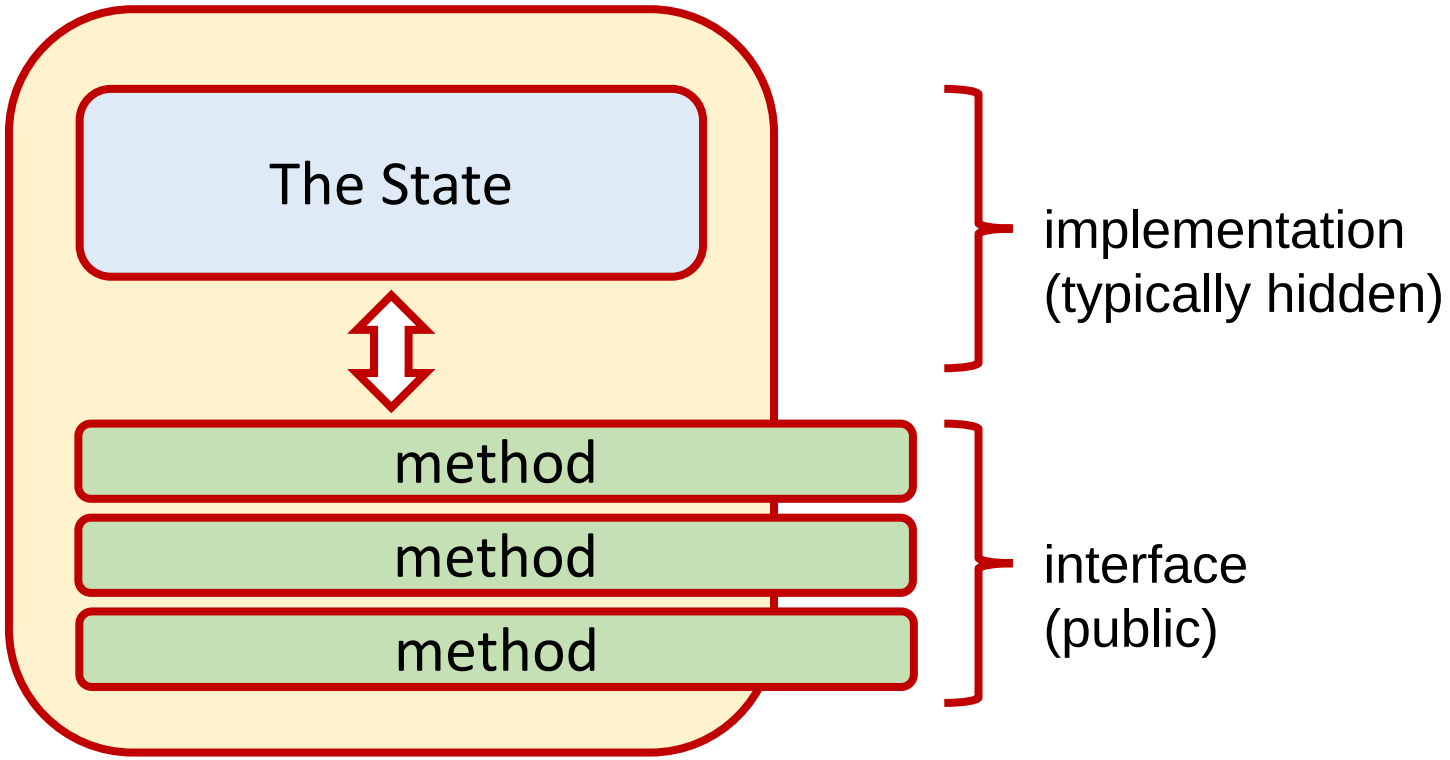
- files and folders
- windows
- apps
- menus
- status bar

Behaviors

- open/close
- move
- resize

The paradigm has clearly changed!

The object paradigm



Objects have fields and attributes that constitute the **state** of the object.

The state is accessed and modified through various **methods** that constitute the object's **interface**.

The layout and organization of the state (i.e., the **implementation**) is usually shielded from the user.

The Interface/Implementation Dichotomy

di·chot·o·my

/dīˈkädəmē/

noun

a division or contrast between two things that are or are represented as being opposed or entirely different.
"a rigid **dichotomy between** science and mysticism"

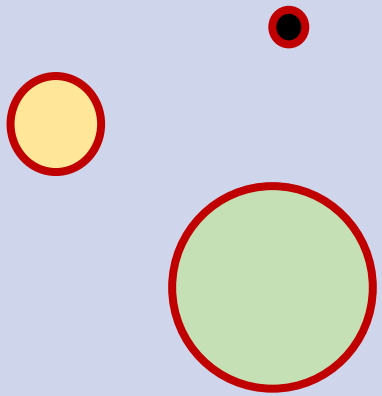
Similar: division | separation | divorce | split | gulf | chasm | difference

We can operate complex machines without knowing what's *under the hood*.

Similarly, to build more complex software, we need to express ideas at a higher level of abstraction with a focus of interface over implementation. Object-oriented thinking enables us to do this.

Classes and Objects

Classes define a **type**. It acts as a *template* or blueprint.
We then construct many objects that are **instances** of a particular class.

Class (Type)	List	Circle	Cat	Account
Object Instances	[1, 2, 3] ['Jack', 'Abby'] [] [(0,0), (5,7), (-2,2)]		 my_cat  not_my_cat	MyChecking MySavings . . *etc*.

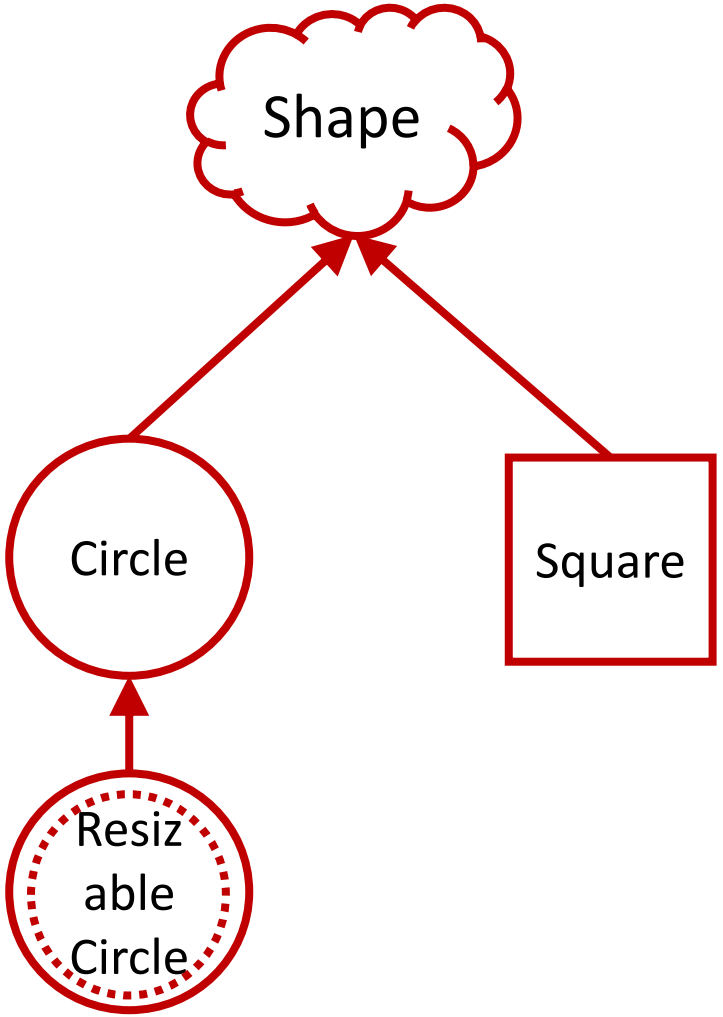
Lists, Tuples, and Dictionaries are objects!

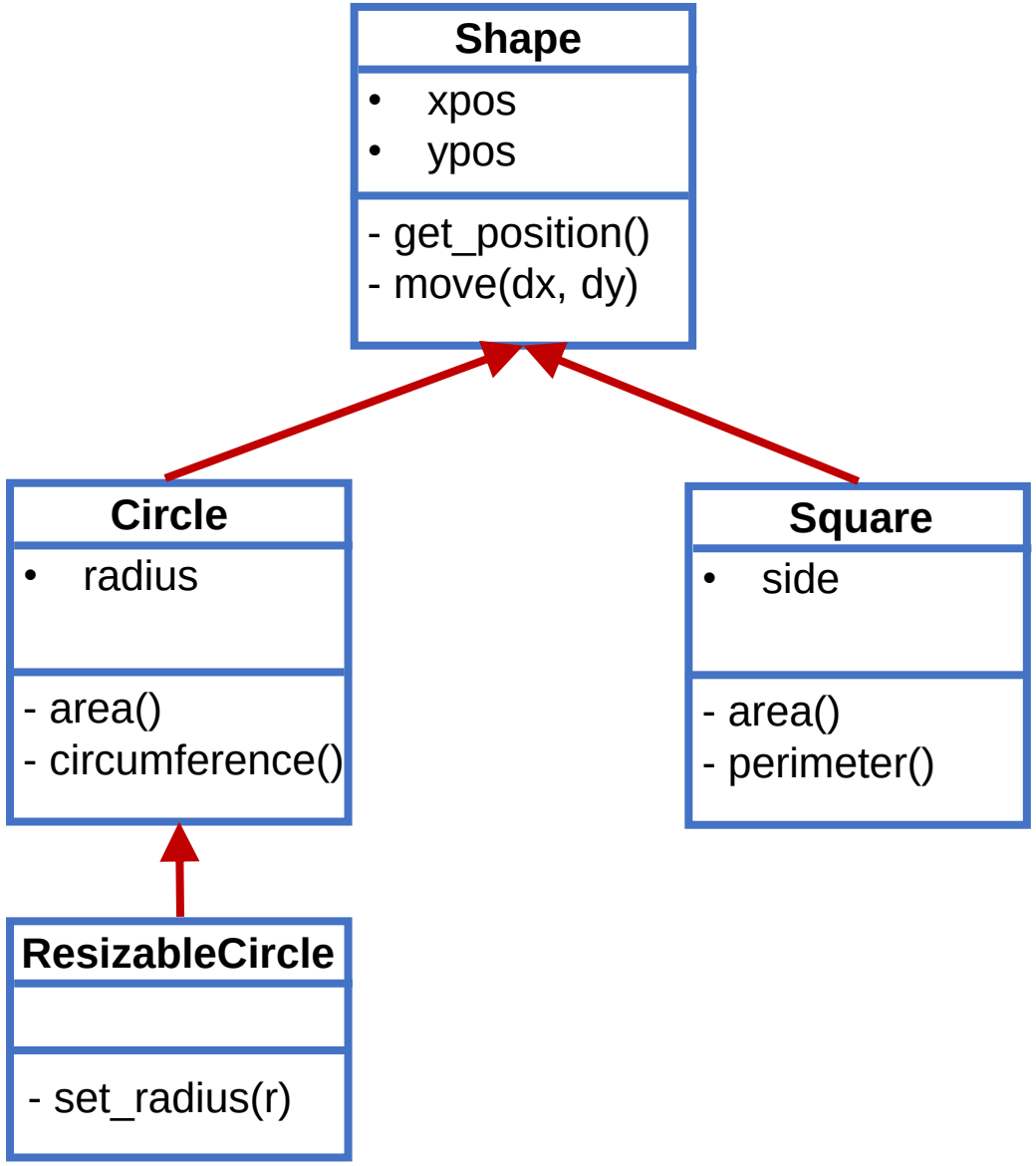
```
L = ['A', 'B', 'C', 'D']

# What is the class/type?

type(L)
list

# Call a method on the object

L.append('E')

L
['A', 'B', 'C', 'D', 'E']

# Pass L to a function - we can still do this!

len(L)
5

L[2]
'C'
```

```
T = ('A', 'B', 'C', 'D')

type(T)
tuple

len(T)
4

T[2]
'C'
```

```
D = {'ann':44, 'reuban':29, 'dachuan':37}

D['reuban']
29

len(D)
3

list(D.keys())
['ann', 'reuban', 'dachuan']
```

Class Inheritance

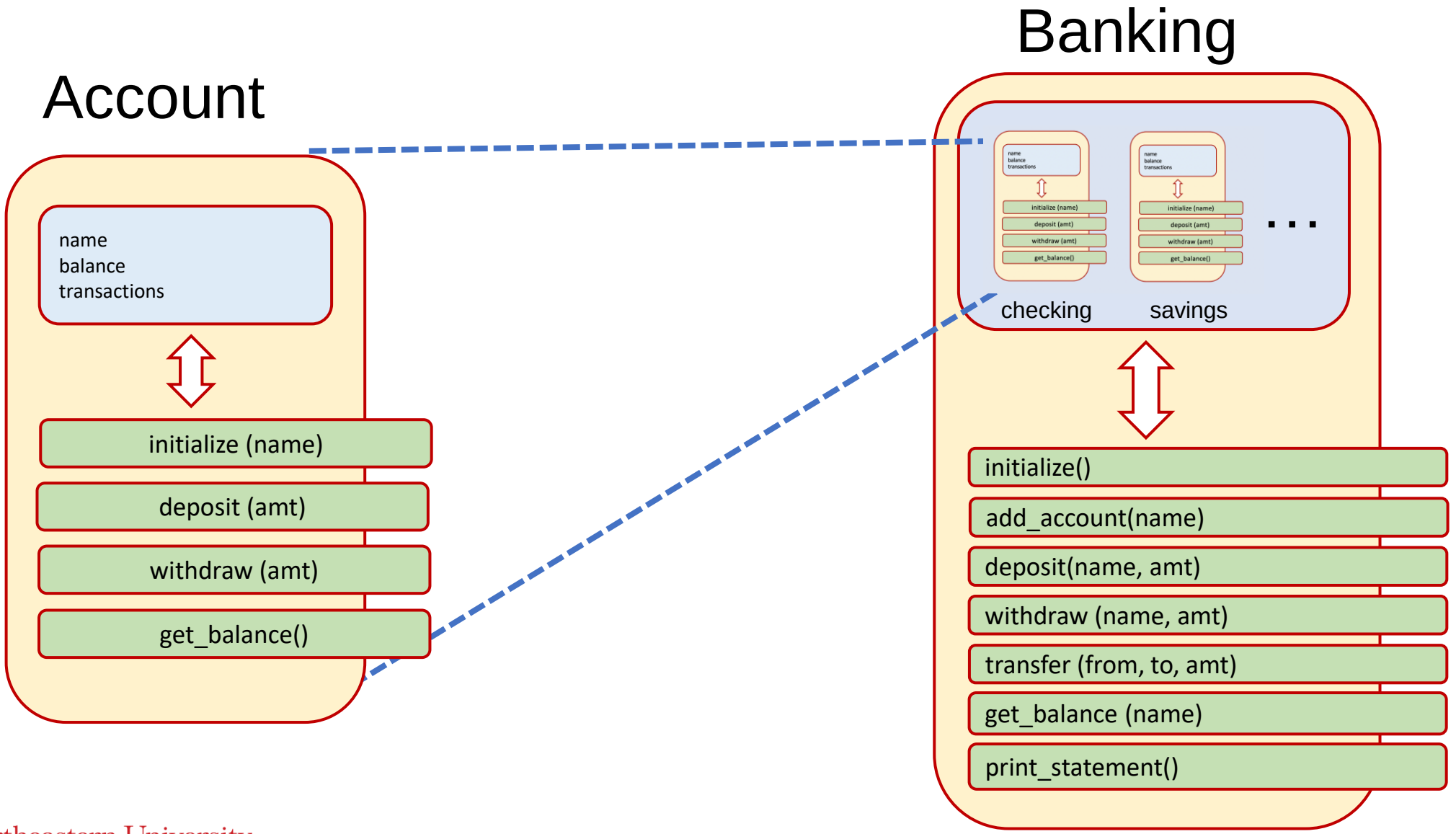




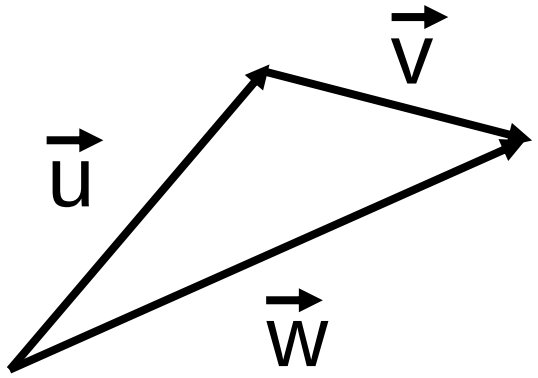
Abstract Base Classes (ABCs)

Abstract base classes define a set of methods and properties that a class must implement in order to be considered a duck-type instance of that class.

Composition



Operator overloading



This:

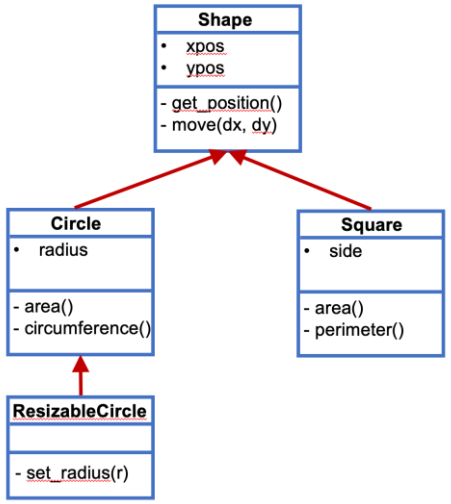
```
u = Vector(3, 4)

v = Vector(2, -1)

w = u + v

w
5i + 3j

u * v  # dot product
2
```

Not this:

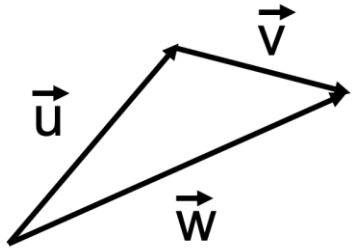
```
u = [3, 4]

v = [2, -1]

w = add(u, v)

w
[5,3]

dot_product(u, v)
2
```

The advantages of object-oriented design

Code re-use
(inheritance)

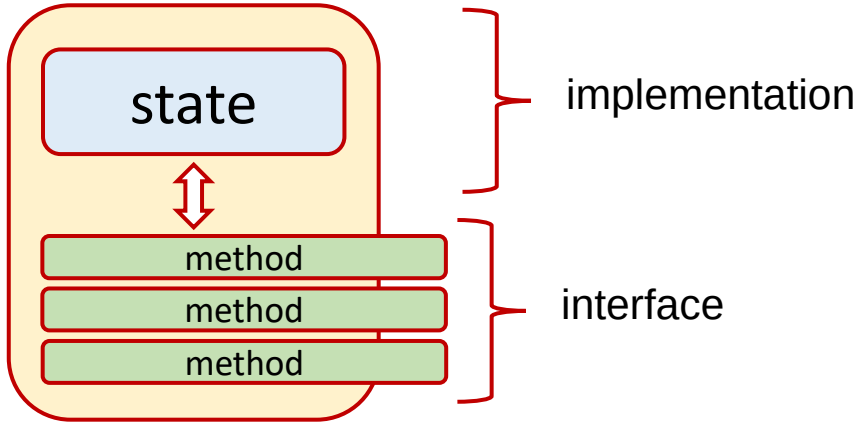


Expressive syntax
(operator overloading)



$$w = u + v$$

Managing complexity
(separation of implementation
and interface)



Static typing: Data types are pre-declared (Java)

- In most programming languages, the type of a variable, an object instance, a function parameter, and even the type of the function's return value (if any) must be declared and is FIXED **(static).**

- These constraints allow the compiler to check our program *before running the code.*

- (Usually this is done at *compile time* – such languages have a compiler.)

Static Typing: continued

In Java, we might say:

```
void drawShape(Shape sh) {
.
.
}
```

The function can accept any object instance of a Shape (*or any Shape sub-class*). We may not know the shape until runtime. This is called **runtime polymorphism**. (But it must be some kind of shape in accordance with our class hierarchy.)

Dynamic typing: Data types are determined at run-time

- In python, the interpreter figures out and checks the type at runtime. This is known as **dynamic typing**.

- Functions and methods don't specify return types or parameter types.

```
def draw_shape(sh):
    """ Draw a shape """
    sh.draw()
    .
    .
```

Dynamic Typing

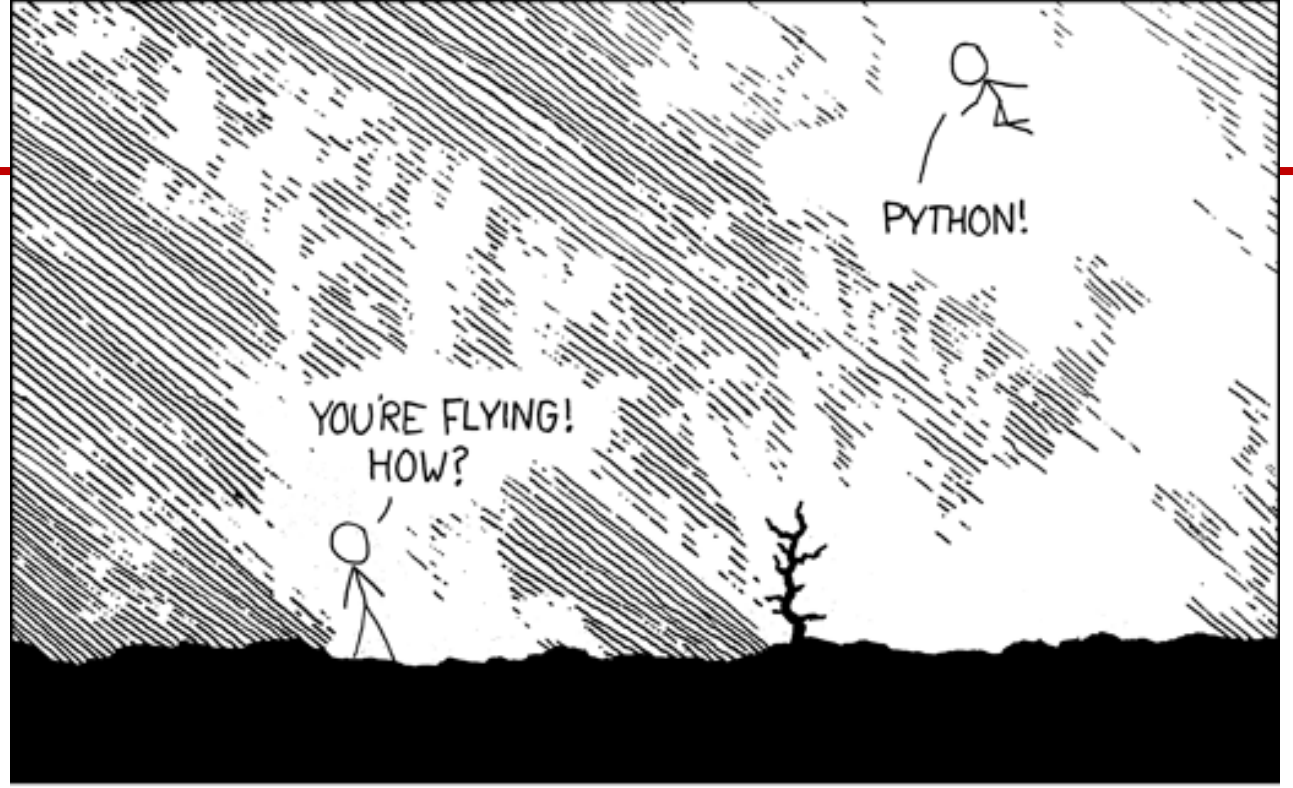
Advantage: Greater flexibility

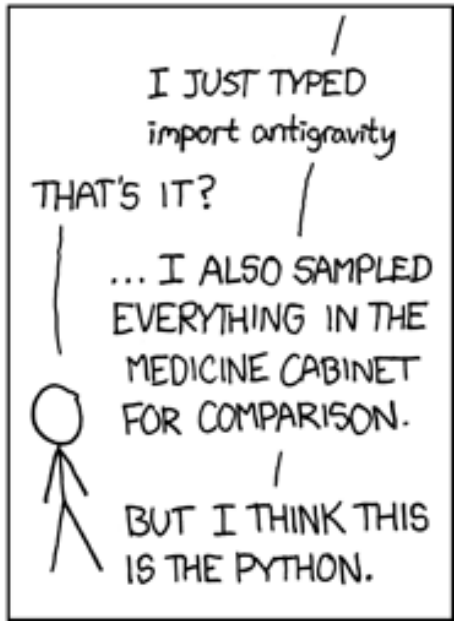
Disadvantage:

- We don't know there is a problem until the program crashes.
- The absence of a compiler means that certain code optimizations can not be performed.
- Interpreted languages like python will tend to run slower!

*How do we know that **sh** is part of the Shape class hierarchy that support a **draw** method? (We don't)*

XKCD





https://xkcd.com/353/

Type Hints

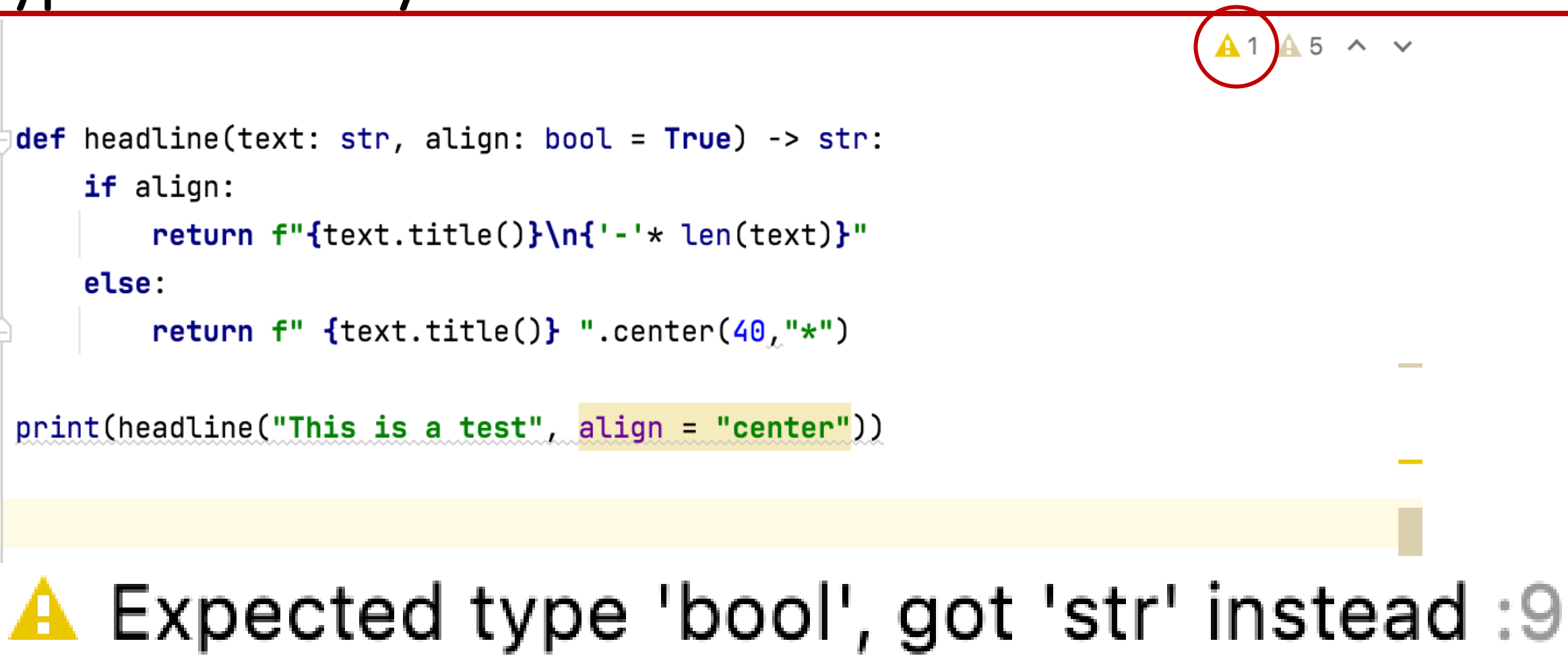
Type hints make it possible to do static typing (with the help of other tools). They are not enforced!

Introduced in PEP 484 / Python version 3.5+:

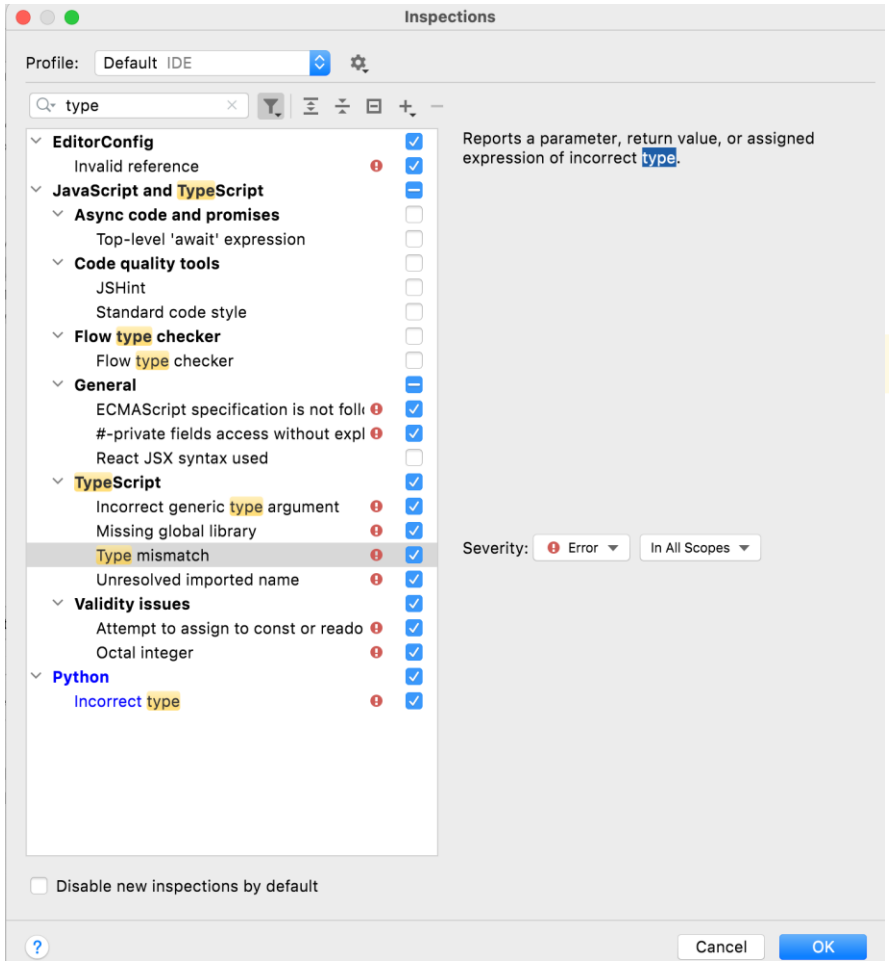
```
def greeting(name: str) -> str:
    return 'Hello ' + name
```

From the PEP: *Python will remain a dynamically typed language, and the authors have no desire to ever make type hints mandatory, even by convention.*

Type Hints in PyCharm



Type Hints in PyCharm can be converted to Errors




```
print(headline("This is a test", align = "left"))
```

Expected type 'bool', got 'str' instead

Parameter **align** of type_hints.headline
align: bool

Type Hint Convensions

PEP 8 Recommends

- **Use normal rules for colons, no space before and one space after a colon**

```
text: str
```

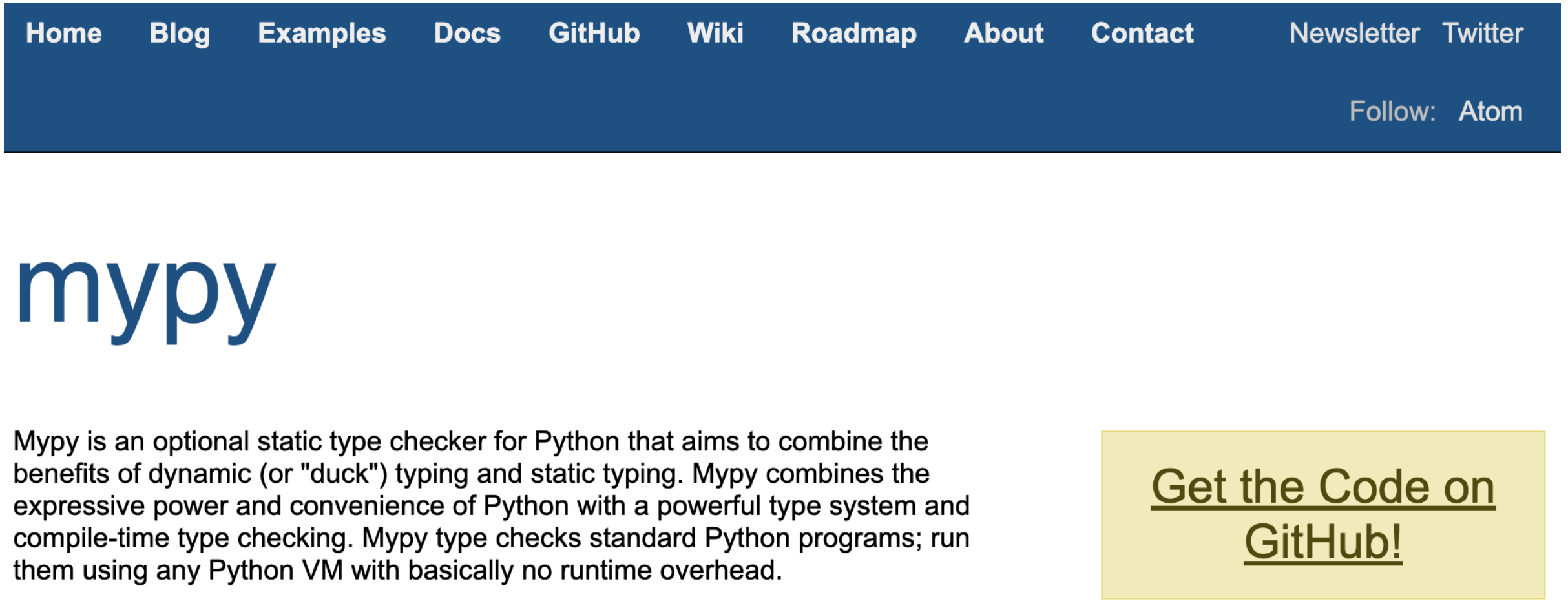
- **Use spaces around the `=` sign when combining an argument annotation with a default value**

```
align: bool = True
```

- **Use spaces around the `->` arrow**

```
def headline(...) -> str
```

Mypy (mypy-lang.org): A static type checker for python



Mypy (mypy-lang.org): A static type checker for python

```
(base) [522 uranus:10_patterns1 ] mypy type_hints.py
type_hints.py:8: error: Argument "align" to "headline" has incompatible
 type "str"; expected "bool"
Found 1 error in 1 file (checked 1 source file)
(base) [523 uranus:10_patterns1 ]
```

(running mypy from the command line.)

```
(base) [524 uranus:10_patterns1 ] mypy type_hints.py
Success: no issues found in 1 source file
(base) [525 uranus:10_patterns1 ]
```

Type Hints: Advantages and Disadvantages

ADVANTAGES:

- Extra pre-runtime checking
- Improved readability / documentation – better than docstrings
- Improved functionality in certain IDEs like PyCharm
- Improved code / architecture
- Improved reliability for published libraries

DISADVANTAGES:

- More verbose code
- Limited to Python 3+
- Advanced features require 3.6+
- Slightly slower program start-up
- Not useful for small throw-away code

Type Hints are a specific kind of Python Annotation

Annotations such as Type Hints can be *inspected* programmatically at runtime using the

```
.__annotations__
```

attribute or method

```
{'radius': <class 'float'>, 'return': <class 'float'>}
{'pi': <class 'float'>, 'nothing': <class 'str'>}
```

Type comments (older - used by some checkers)

```
>>> import math
>>>
>>> def circumference(radius):
...     # type: (float) -> float
...     return 2 * math.pi * radius
...
>>>
>>> circumference(4.5)
28.274333882308138
>>>
```

Not stored in __annotations__ but Mypy will pick them up.

Type comments (older - used by some checkers)

```
2
3  def headline1(text, width=80, fill_char="-"):
4      # type: (str, int, str) -> str
5      return f" {text.title()} ".center(width, fill_char)
6
7  print(headline1("type comments work", width=40))
8
9  def headline2(
10     text,           # type: str
11     width=80,       # type: int
12     fill_char='-',  # type: str
13 ):                  # type: (...) -> str
14     return f" {text.title()} ".center(width, fill_char)
15
16 print(headline2("these type comments also work", width=70))
17
```

Typing for sequences, maps, tuples, etc.

```
2   from typing import List, Dict, Tuple

10  mylist: List[int] = [1, 2, 3, 4]
11  mydictionary: Dict[str, int] = {'a': 1, 'b': 2, 'c': 3}
12  coord: Tuple[int, int, int] = (7, 12, 3)
13
14  # something fancier
15  coordinate_list: List[Tuple[int, int]] = []
16
```

```
mylist.append('a')
print(mylist)

mydictionary[12] = "hello"
print(mydictionary)

print(coord)
coord = (0, 0)
print(coord)

coordinate_list.append((1,2))
coordinate_list += [(2,3), (0,0)]
coordinate_list.append(("hello", "world"))
print(coordinate_list)
```

Typing for sequences, maps, tuples, etc.

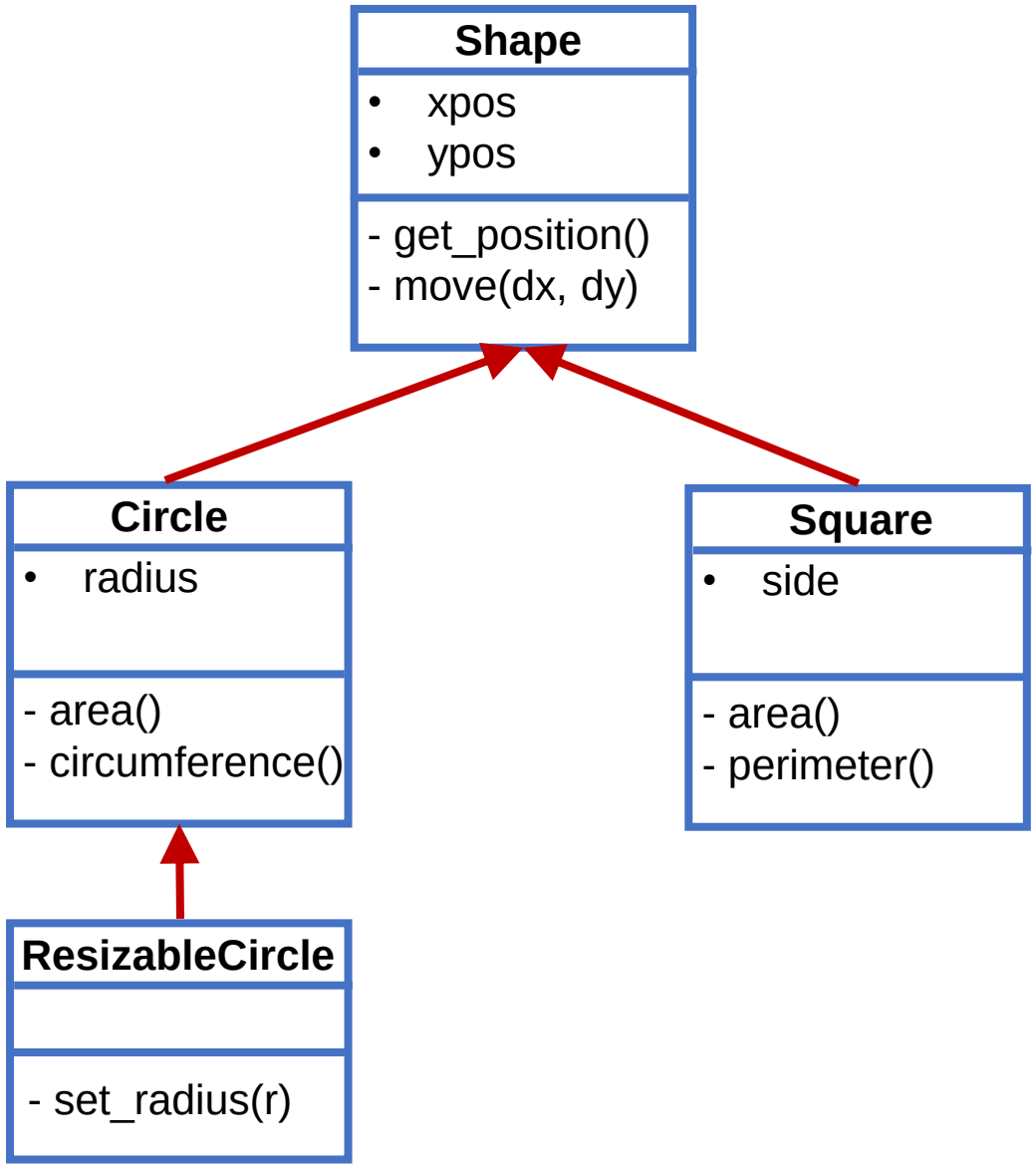
```
(base) [503 uranus:10_patterns1 ] mypy annotations2.py
annotations2.py:17: error: Argument 1 to "append" of "list" has incompatible type "str"; expected "int"
annotations2.py:20: error: Invalid index type "int" for "Dict[str, int]"; expected type "str"
annotations2.py:20: error: Incompatible types in assignment (expression has type "str", target has type "int")
annotations2.py:26: error: Incompatible types in assignment (expression has type "Tuple[int, int]", variable has type "Tuple[int, int, int]")
annotations2.py:31: error: Argument 1 to "append" of "list" has incompatible type "Tuple[str, str]"; expected "Tuple[int, int]"
Found 5 errors in 1 file (checked 1 source file)
```

Polymorphism

Polymorphism is the ability to treat a class differently depending on which subclass is being implemented.

For example, if we call the **area** method, the calculation performed depends on which subclass we are using - circle or square.

If we wanted to, a specific subclass of Shape could implement a different way of moving, or it could inherit its behavior from the Shape parent class.



Duck Typing

Duck typing is related to dynamic typing

- **The type or the class of an object is less important than the methods it defines**
- **Instead of checking for the class or type, check the object for the presence of specific methods and/or attributes**

Duck typing: *If it behaves like a duck, it's a duck!*

Python doesn't enforce the type hierarchy when passing parameters.
As long as the parameters support the behaviors we need, Python will not complain!

```
def draw_shape(sh):
    """ Draw a shape """
    sh.draw()
    .
    .

draw_shape(not_a_shape)  # This works as long
                         # as not_a_shape has
                         # a draw method!
```

Design Patterns

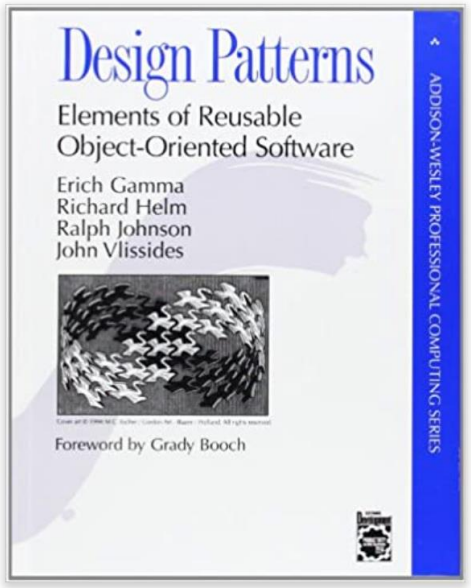
Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition

by Erich Gamma (Author), Richard Helm (Author), Ralph Johnson (Author), John Vlissides (Author), Grady Booch (Foreword)

4.5 out of 5 stars 1,548 ratings

#1 Best Seller in Software Reuse

Look inside



ISBN-13: 978-0201633610
ISBN-10: 0201633612
Why is ISBN important?

Kindle	Hardcover	Paperback	Other Sellers
$32.39	$27.99 - $37.85	from $32.97	See all 12 versions

Buy used: $27.99

Buy new: $37.85

In Stock.

Ships from and sold by Amazon.com.

May be available at a lower price from other sellers, potentially without free Prime shipping.

List Price: ~~$59.99~~ Details
Save: $22.14 (37%)
prime

FREE delivery **Tomorrow, October 15**. Order within 4 hrs 32 mins

Deliver to John - Fountain Hills 85268

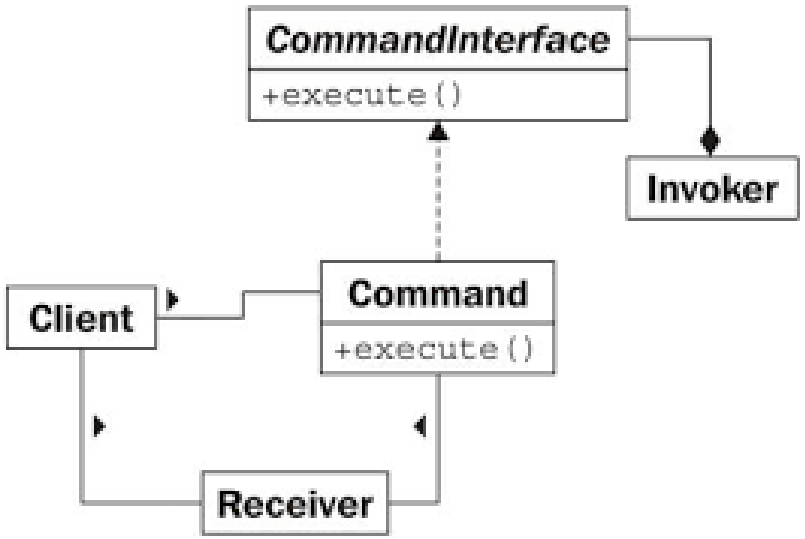
Qty: 1

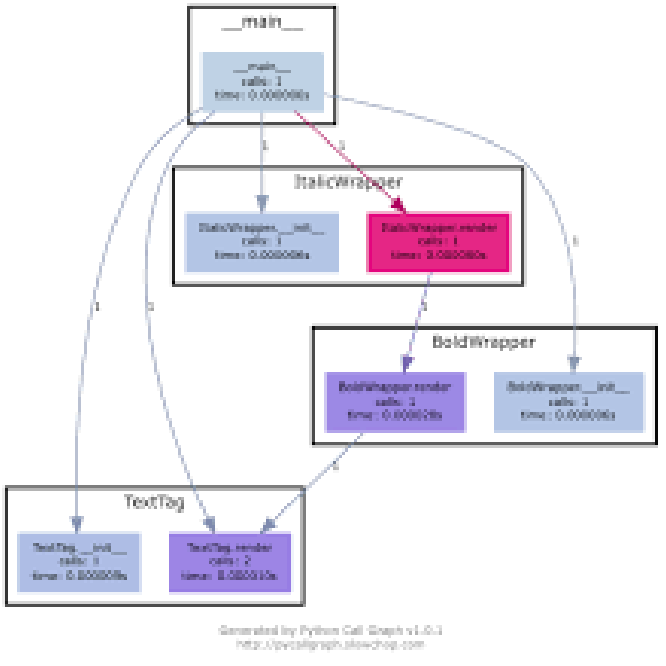
Add to Cart

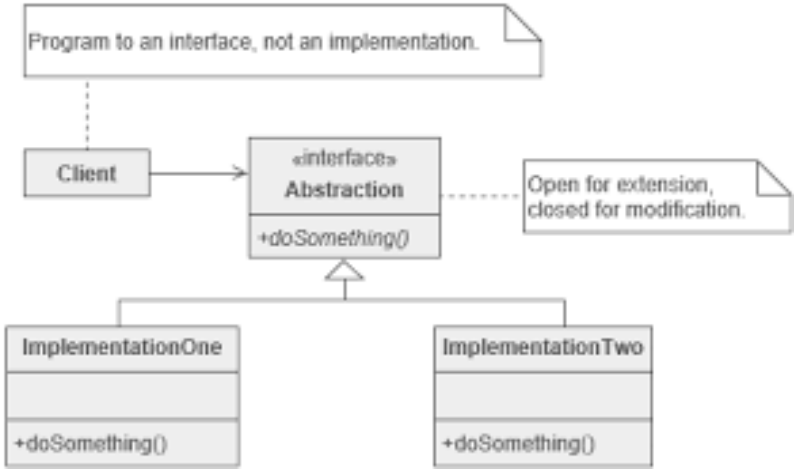
Buy Now

Design Pattern Objectives

Common software problems should be solved in standard ways.
(Don't re-invent the wheel.)







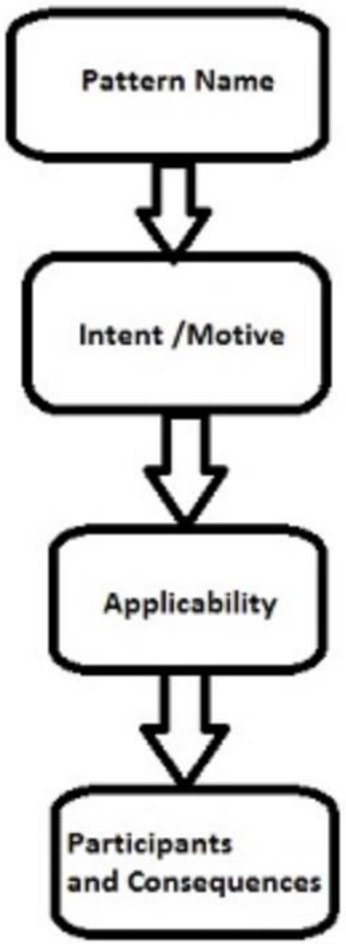
What is a Design Pattern?

- In software engineering, a **software design pattern** is a general, reusable solution to a commonly occurring problem within a given context in software design.

- It is not a finished design that can be transformed directly into source or machine code. Rather, it is a description or template for ***how to solve a problem*** that can be used in many different situations.

  Design patterns are **formalized best practices** that the programmer can use to solve common problems when designing an application or system.

Design Patterns



Pattern Name

It describes the pattern in short and effective manner.

Intent/Motive

It describes what the pattern does.

Applicability

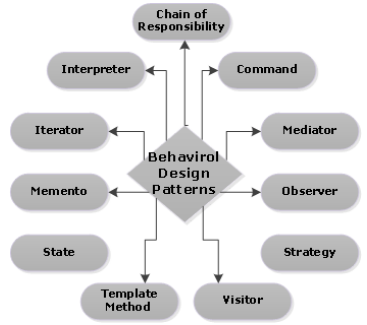
It describes the list of situations where pattern is applicable.

Participants and consequences

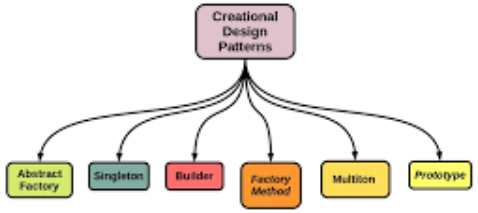
Participants include classes and objects that participate in the design pattern with a list of consequences that exist with the pattern.

Types of Design Patterns

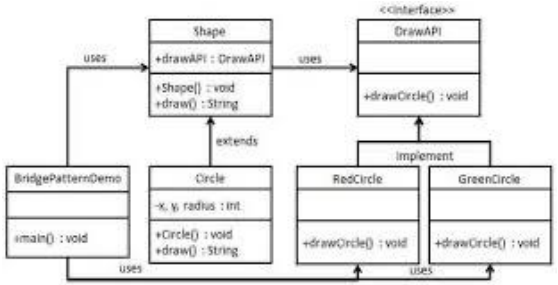
1. **Behavioral patterns** describe interactions between objects and focus on how objects communicate with each other. Behavioral patterns are also used to make the algorithm that a class uses simply another parameter that is adjustable at runtime.



2. **Creational patterns** are used to create objects for a suitable class that serves as a solution for a problem.

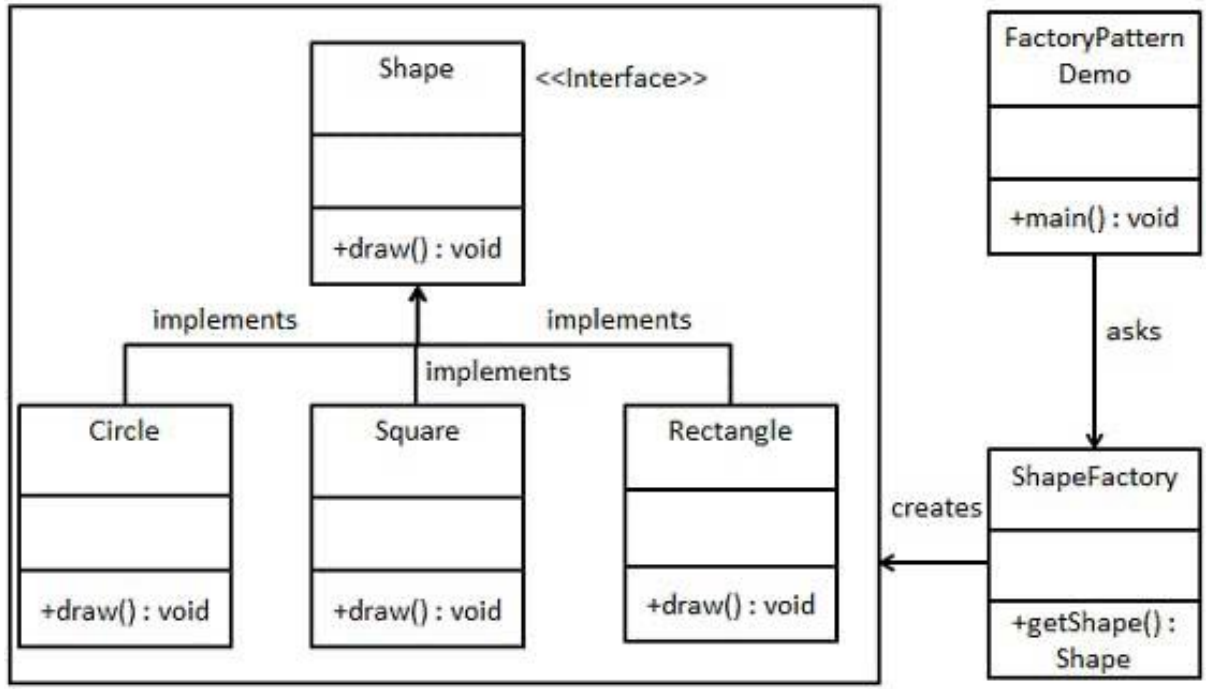


3. **Structural patterns** form larger structures from individual parts, generally of different classes. Structural patterns vary a great deal depending on what sort of structure is being created for what purpose.



Factory Method (A Creational Design Pattern)

- Create a concrete implementation of a common interface
- Separate creation process from the code that depends on the interface
- Delegate the creation of objects to the factory



When to use a Factory Pattern

18 Answers

Sorted by: Highest score (default)

419

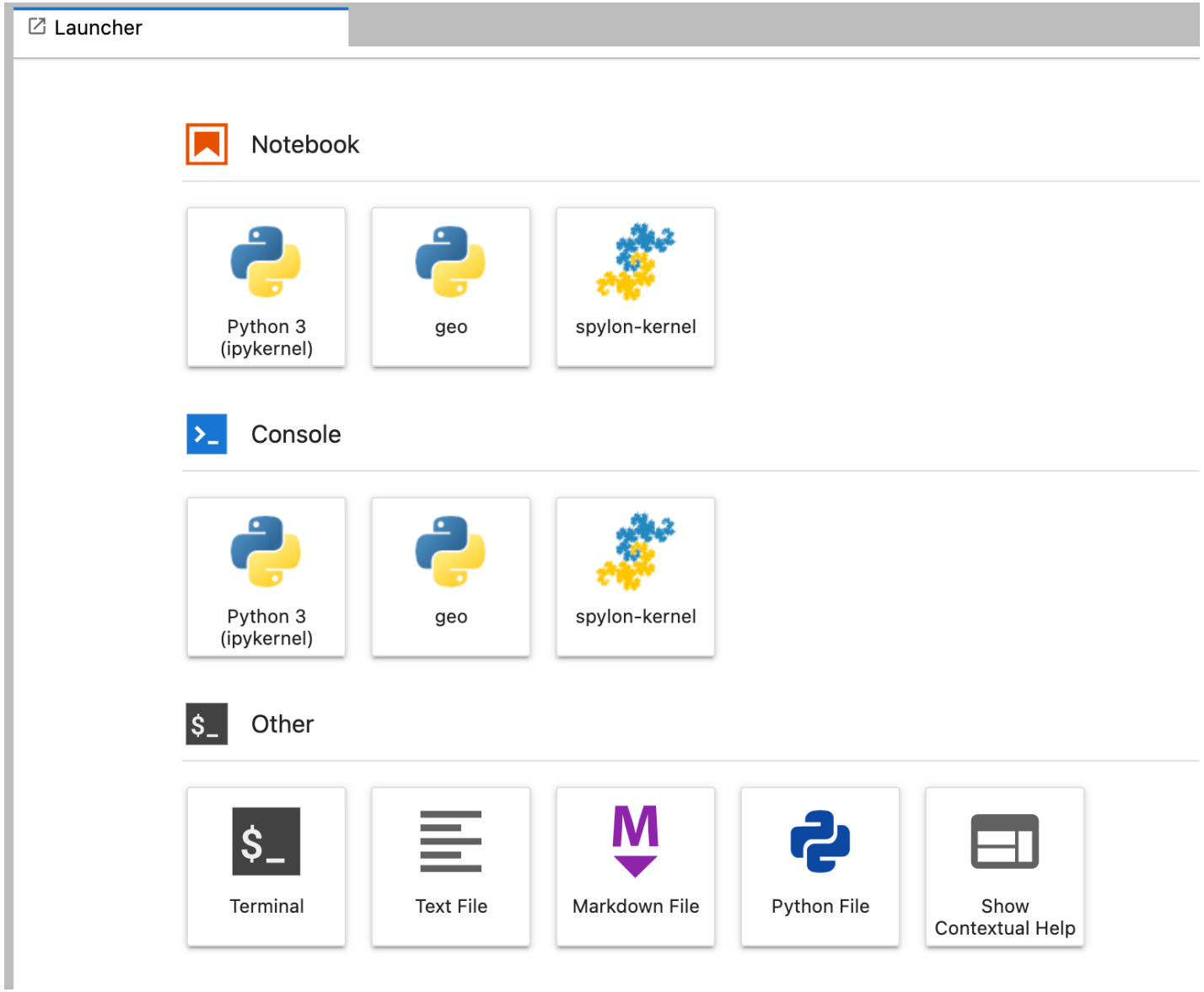
I like thinking about design pattens in terms of my classes being 'people,' and the patterns are the ways that the people talk to each other.

So, to me the factory pattern is like a hiring agency. You've got someone that will need a variable number of workers. This person may know some info they need in the people they hire, but that's it.

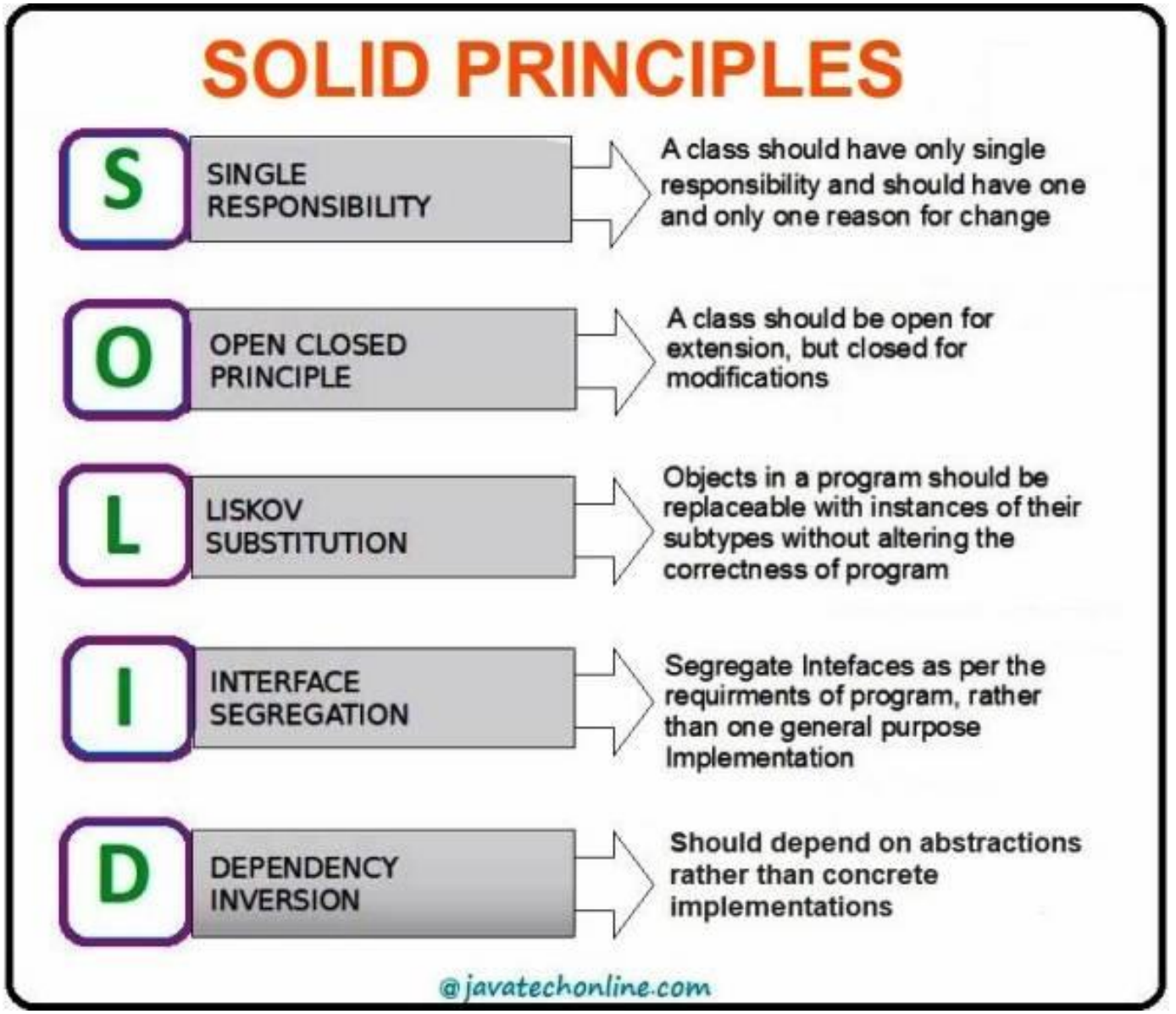
So, when they need a new employee, they call the hiring agency and tell them what they need. Now, to actually *hire* someone, you need to know a lot of stuff - benefits, eligibility verification, etc. But the person hiring doesn't need to know any of this - the hiring agency handles all of that.

In the same way, using a Factory allows the consumer to create new objects without having to know the details of how they're created, or what their dependencies are - they only have to give the information they actually want.

The Jupyter Lab Launcher is a Factory



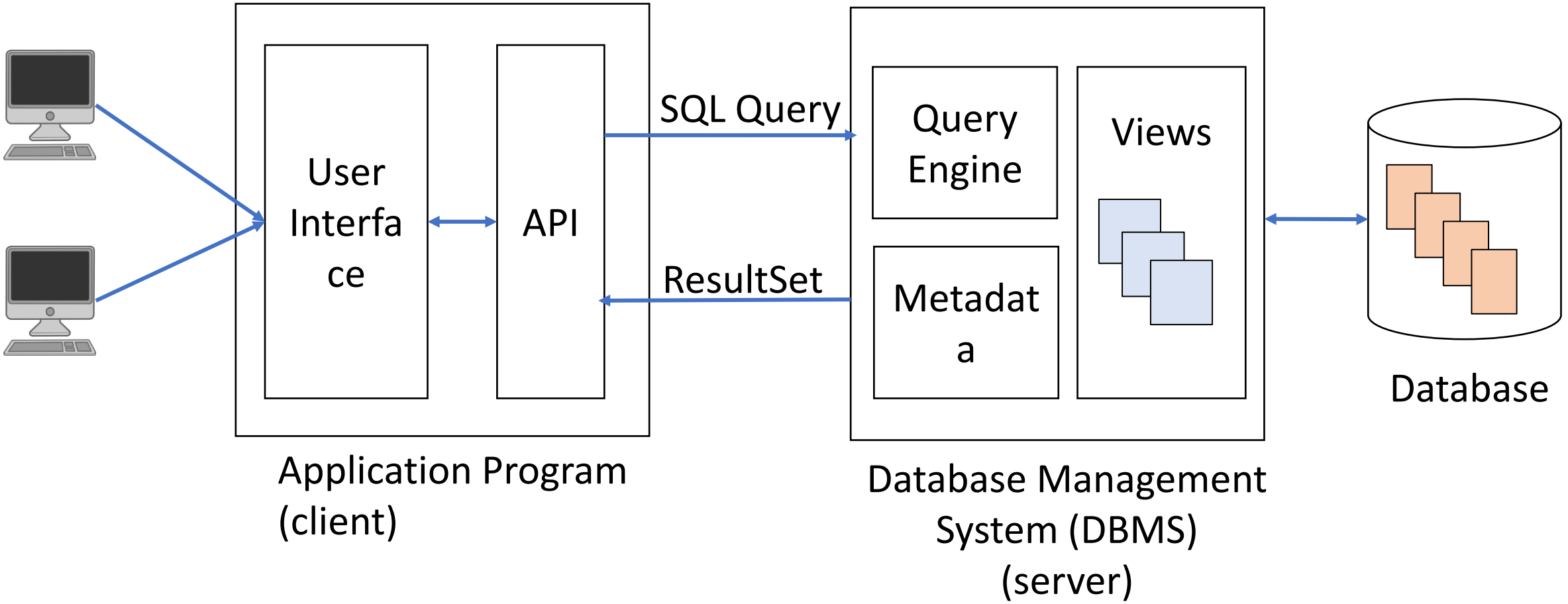
SOLID


SOLID PRINCIPLES
S
SINGLE RESPONSIBILITY
A class should have only single responsibility and should have one and only one reason for change
O
OPEN CLOSED PRINCIPLE
A class should be open for extension, but closed for modifications
L
LISKOV SUBSTITUTION
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of program
I
INTERFACE SEGREGATION
Segregate Intefaces as per the requirments of program, rather than one general purpose Implementation
D
DEPENDENCY INVERSION
Should depend on abstractions rather than concrete implementations
@javatechonline.com

SRP

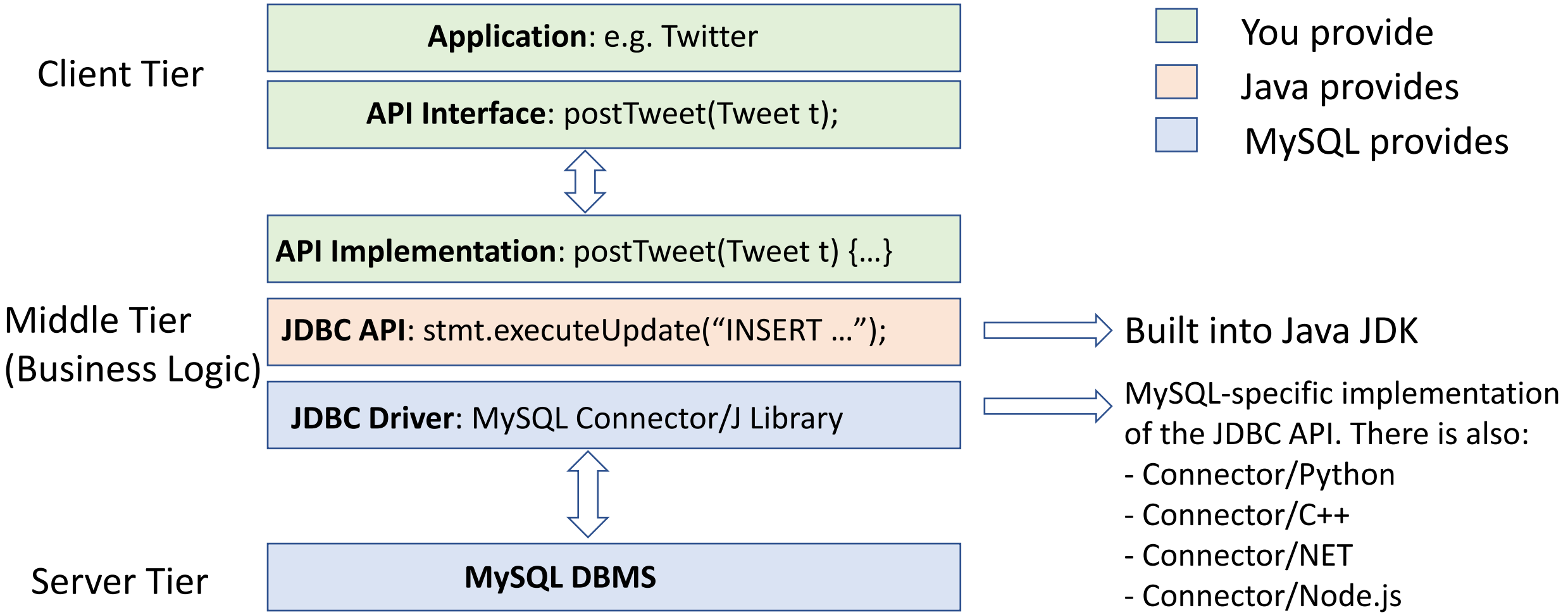
The **single-responsibility principle** (**SRP**) is a computer-programming principle that states that every module, class or function in a computer program should have responsibility over a single part of that program's functionality, and it should encapsulate that part. All of that module, class or function's services should be narrowly aligned with that responsibility. - *Wikipedia*

Two-Tier Client-Server Design Pattern

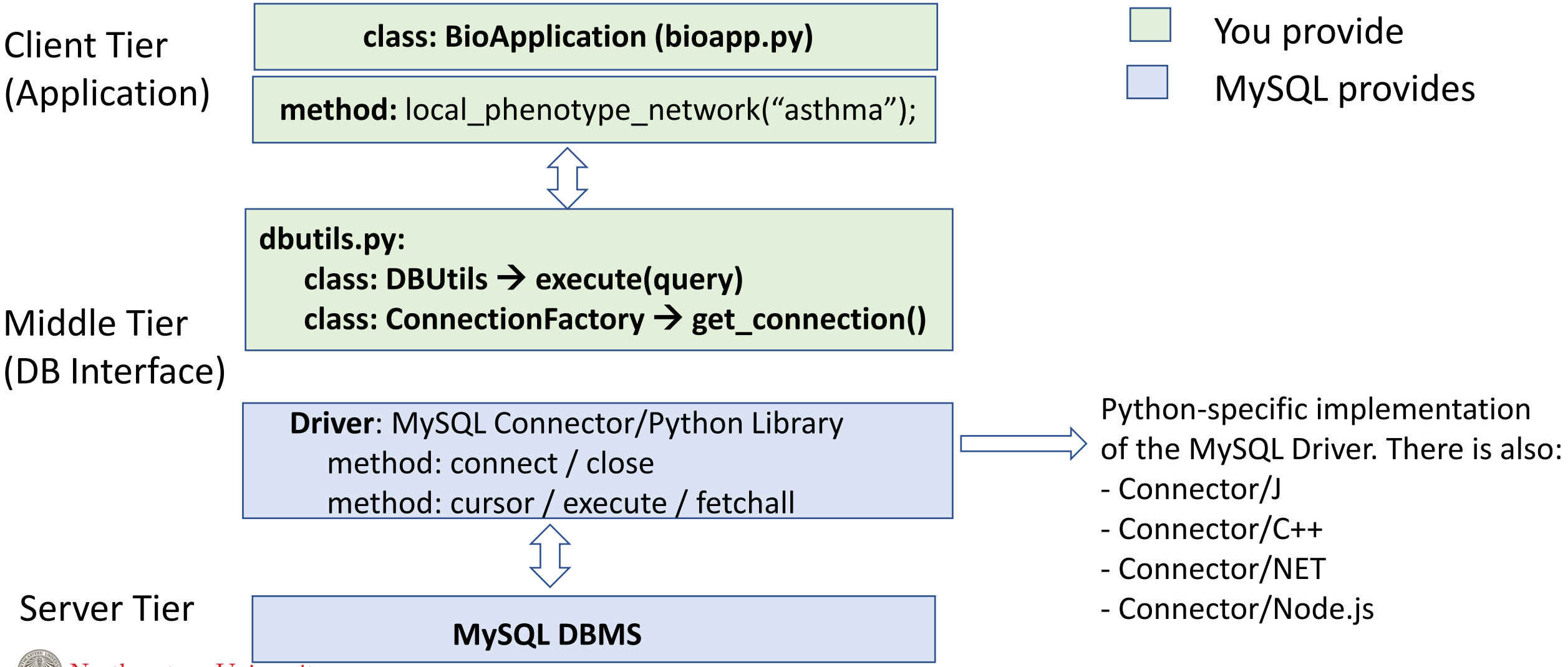


(Usually running on a separate server)

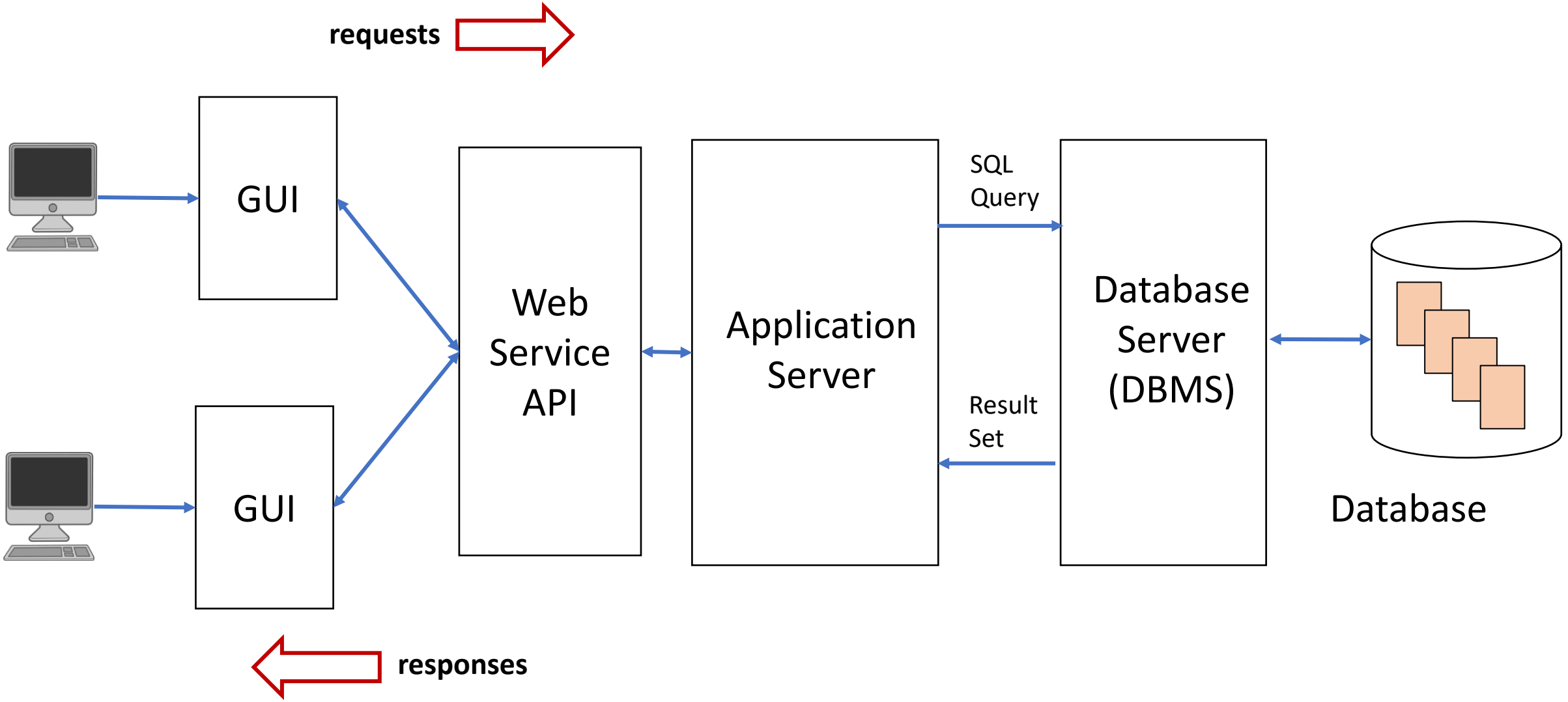
Client-Server Design Pattern (Twitter - Java)



Client-Server Design Pattern (BioApplication - Python)

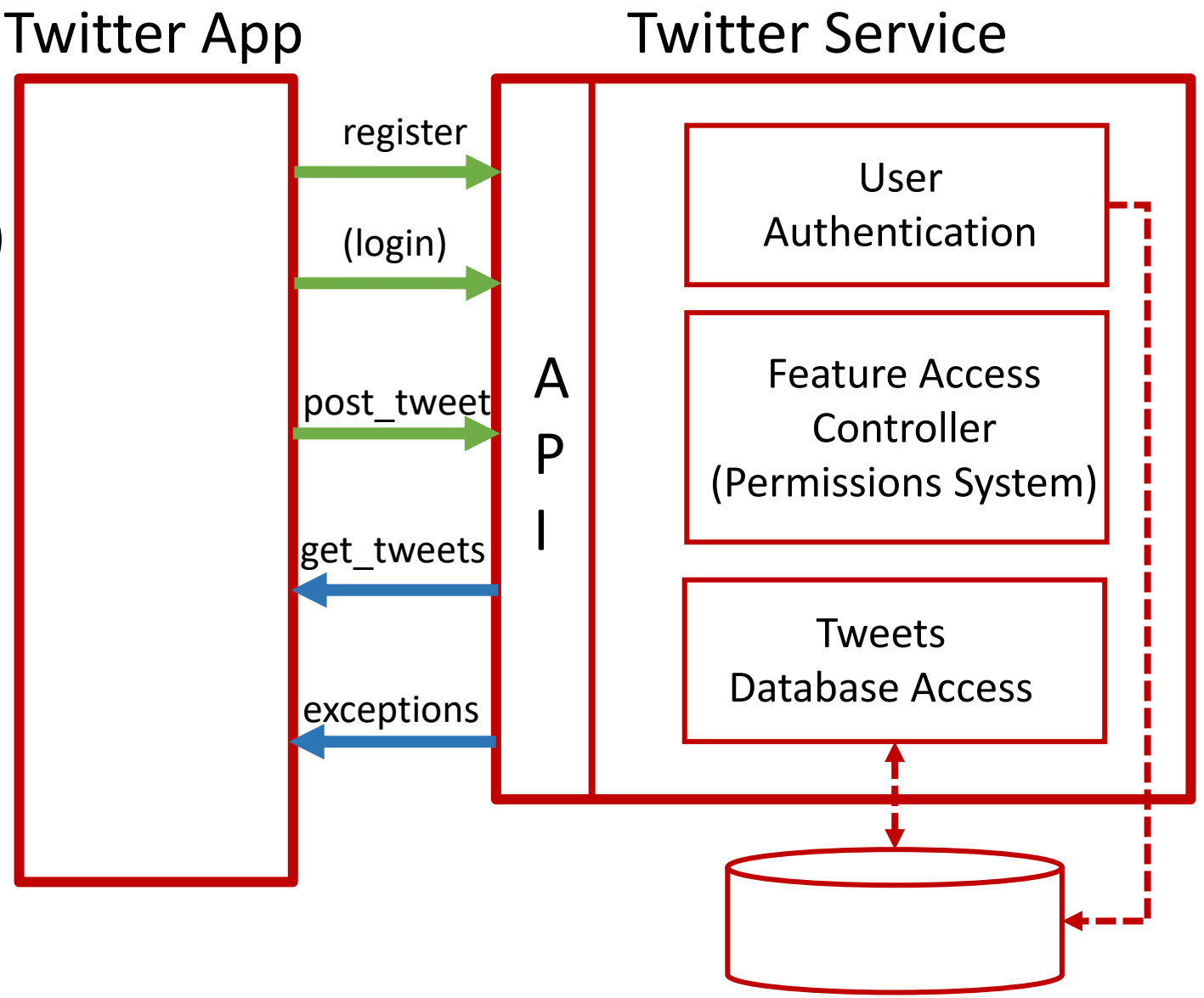


Three-Tier Client-Server Design Pattern



Monolith Twitter

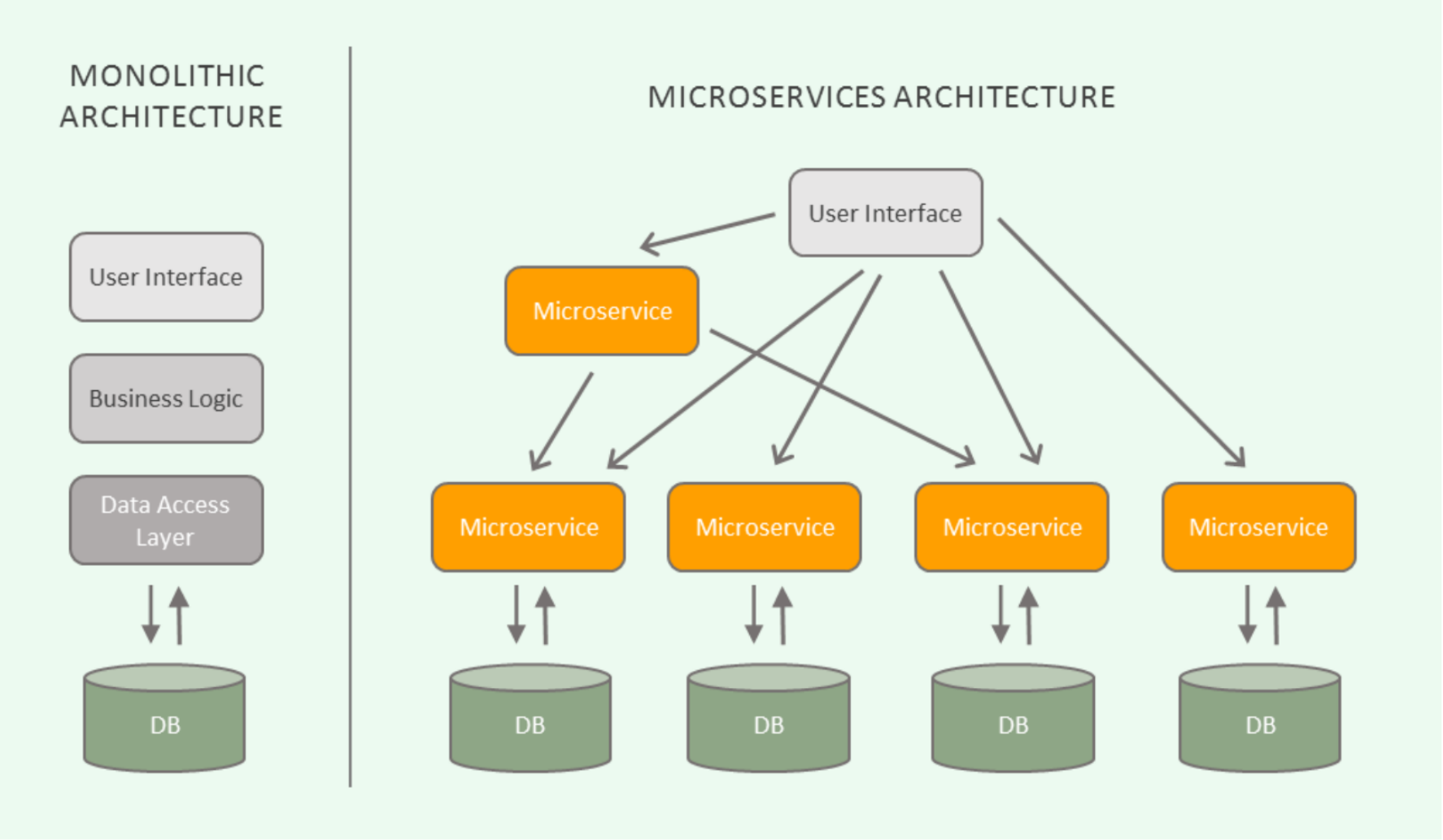
- **User authentication**: responsible for validating user credentials (login / logoff)
- **Permission System**: Keeps track of what features users have access to.
- **Tweets database**: Stores and retrieves user tweets



Disadvantages of Monolithic Architectures

- The application codebase can become very large and difficult to understand by any one person
- Difficult to maintain – requires careful testing at both the component and application level
- Longer deployment times

Monolith vs. Microservices



Source: DZone

Microservices have their challenges too!

- As a loosely-coupled distributed system, microservices tend to be more complex – there are simply more moving pieces that need to be setup, tested, and monitored.
- Failures due to cross-service communication delays: Developers must address problems such as network latency and load balancing
- Integration testing is more challenging. Microservices are siloed. So are the developers that manage the individual services.

Hacker News (news.ycombinator.com) takes up the debate

Response to the blob post: ***Give me back my monolith***
https://www.craigkerstiens.com/2019/03/13/give-me-back-my-monolith/

- **Ease of setup:**
  I'm not sure why you would start with microservices, unless you wanted to show that you could build a microservices application. Monoliths are quicker and easier to setup when you're talking about a small service in the first place.

- **Scalability considerations:**
  It's when an organization grows and the software grows and the monolith starts to get unwieldy that it makes sense to go to microservices. It's then that the advantage of microservices both at the engineering and organizational level really helps.

- **When microservices really shine**
  A team of three engineers orchestrating 25 microservices sounds insane to me.
  A team of of thirty turning one monolith into 10 microservices and splitting into 10 teams of three, each responsible for maintaining one service, is the scenario you want for microservices.

Monolith Twitter Learning Goals

- Composition of Objects
  to build higher-level components
- **Flexible** but **uniform** approach to
  context-sensitive exception handling
  in a non-trivial application architecture
- DO NOT STORE PASSWORDS!

What could possibly go wrong??

registration

- UsernameAlreadyExists
- InvalidPassword

authentication

- UnknownUsername
- IncorrectPassword

access control

- PermissionDenied

Literature Analysis

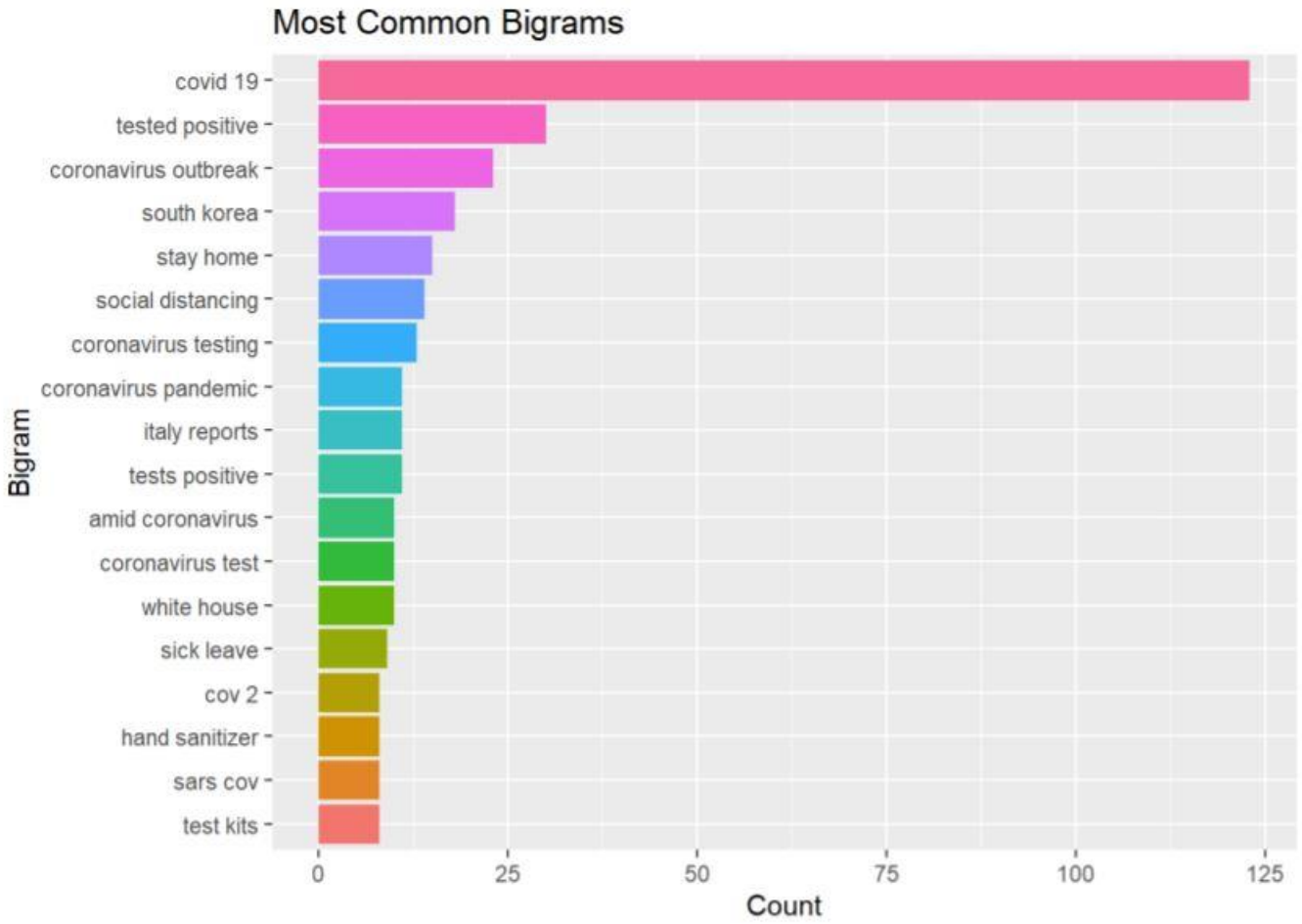
- Use text mining and data visualisation to compare authors and genres
- Possible workflow:

  raw text → clean and process → store → retrieve → run comparisons

- How do visualisations help?

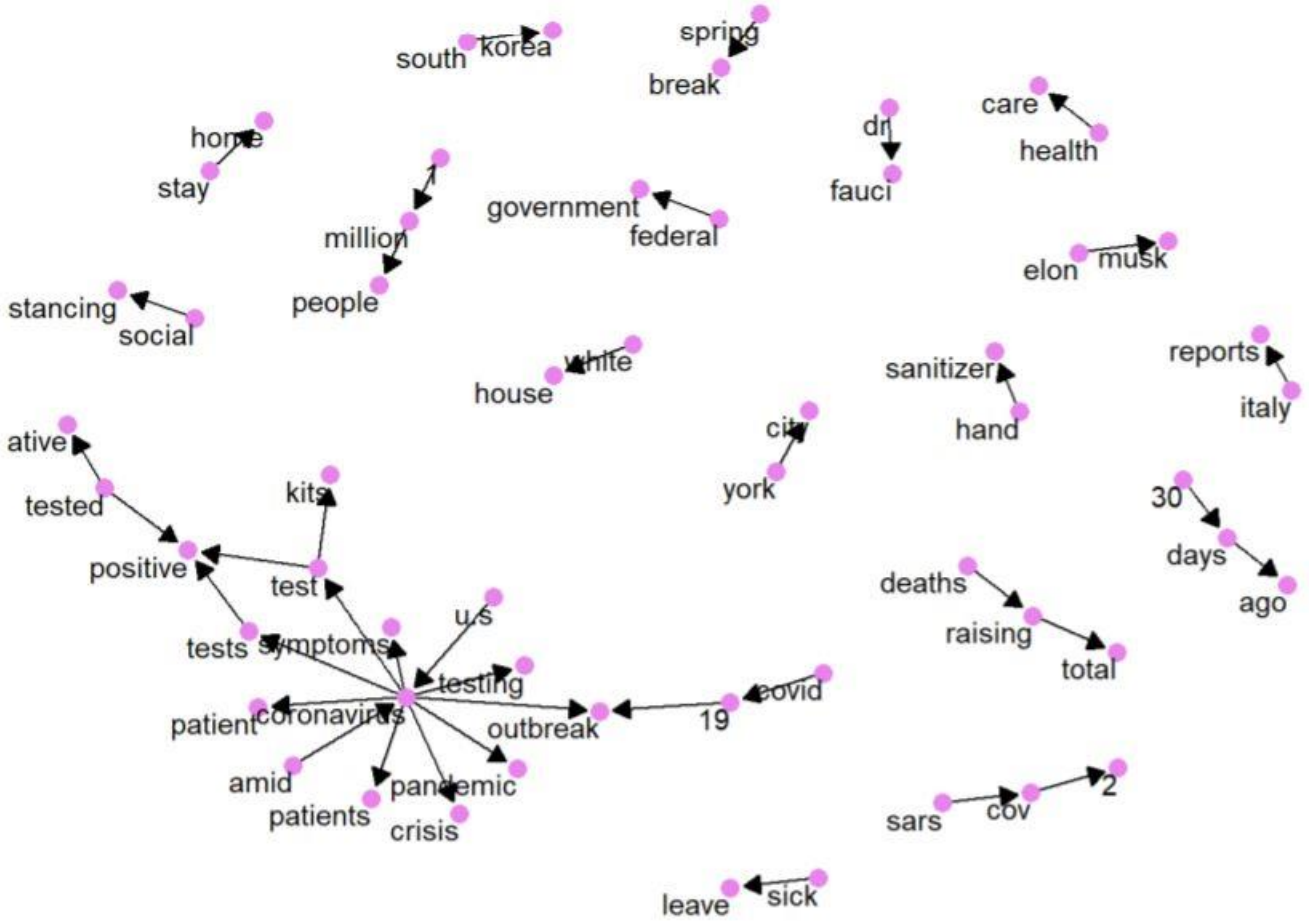
Visualisations – Non-comparative

- Most common bigrams from r/Coronavirus in Mar'20 [1]



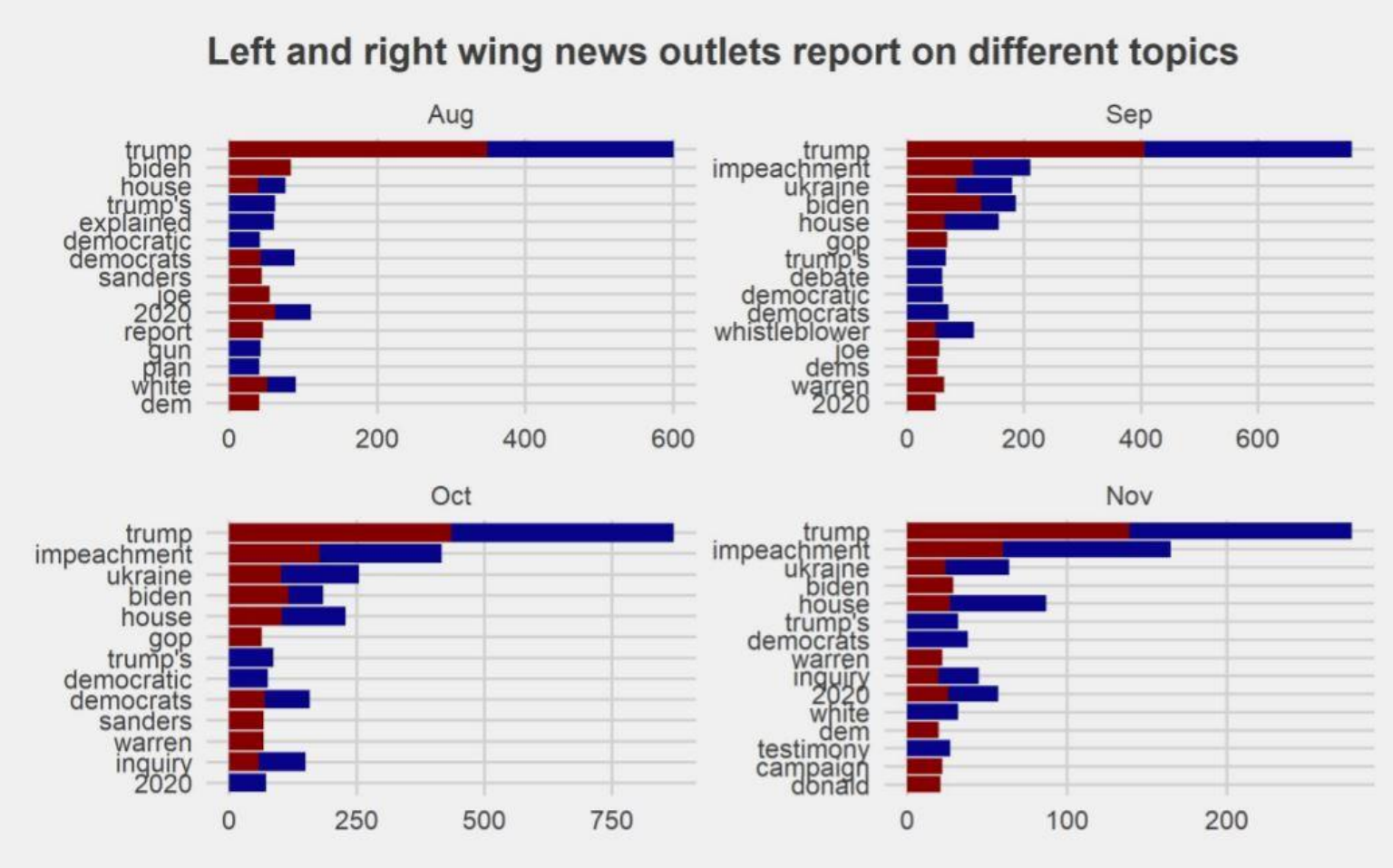
Word/Token Networks

- Model bigrams into a network [1]

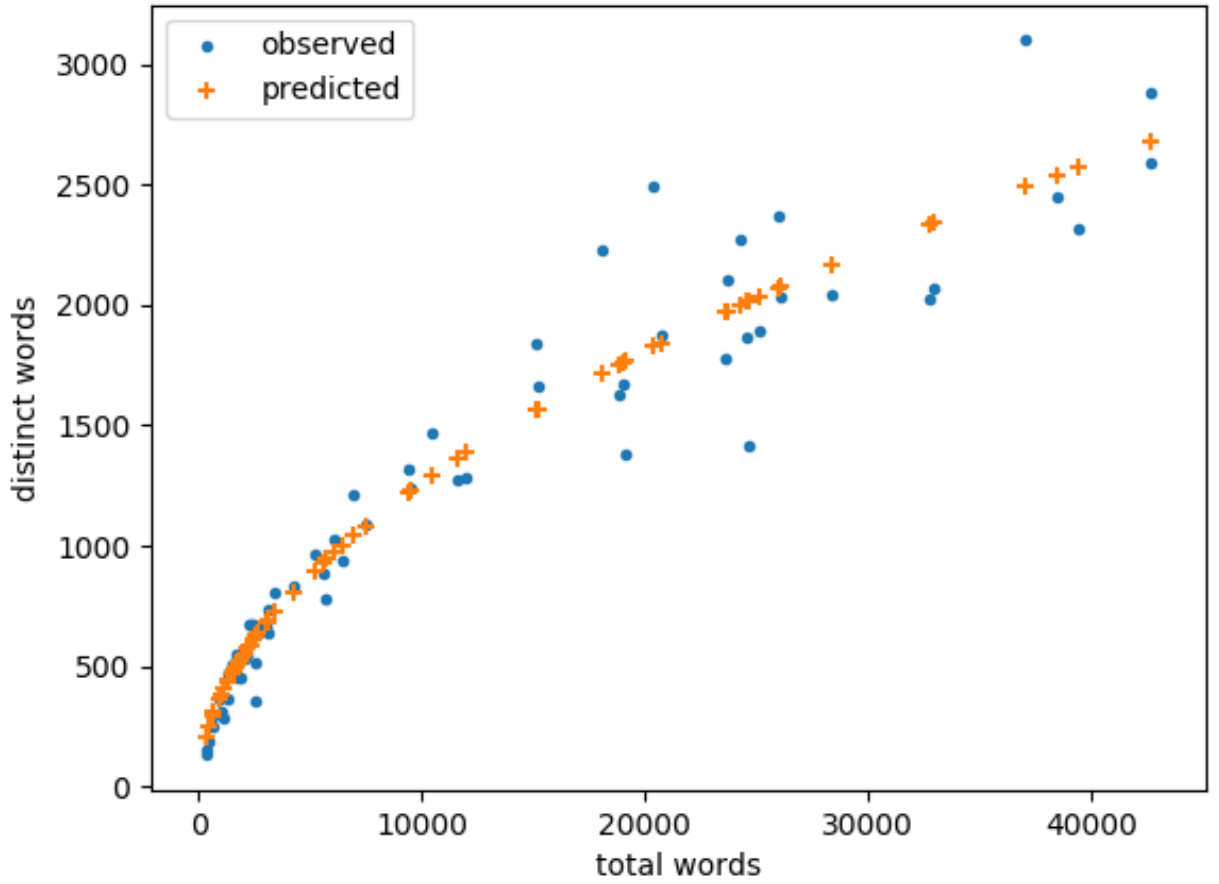


Visualisations – Comparative

- Similarities and differences between news sources in the US [2]



Heaps' Law: $|V| = KN^B$



Zipf's Law

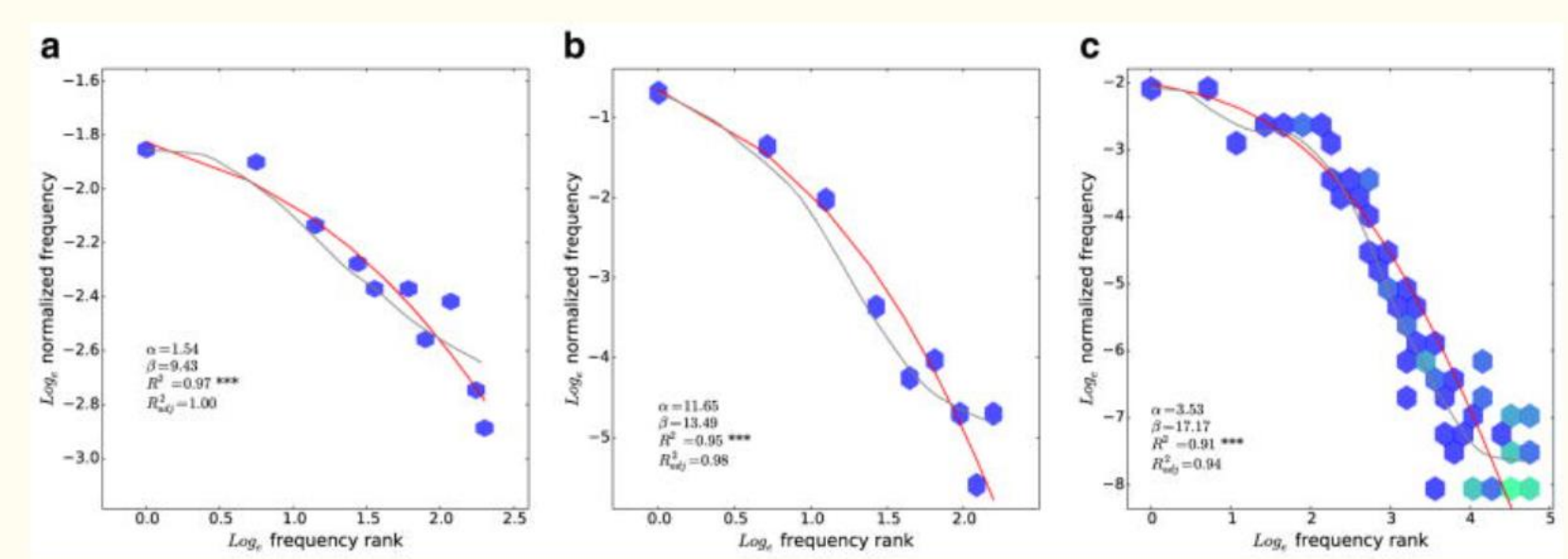
Zipf's law: The *r-th* most frequent word has a frequency $f(r)$ that scales according to:

$$f(r) \propto \frac{1}{r^{\alpha}}$$

or more generally:

$$f(r) \propto \frac{1}{(r+\beta)^{\alpha}}$$

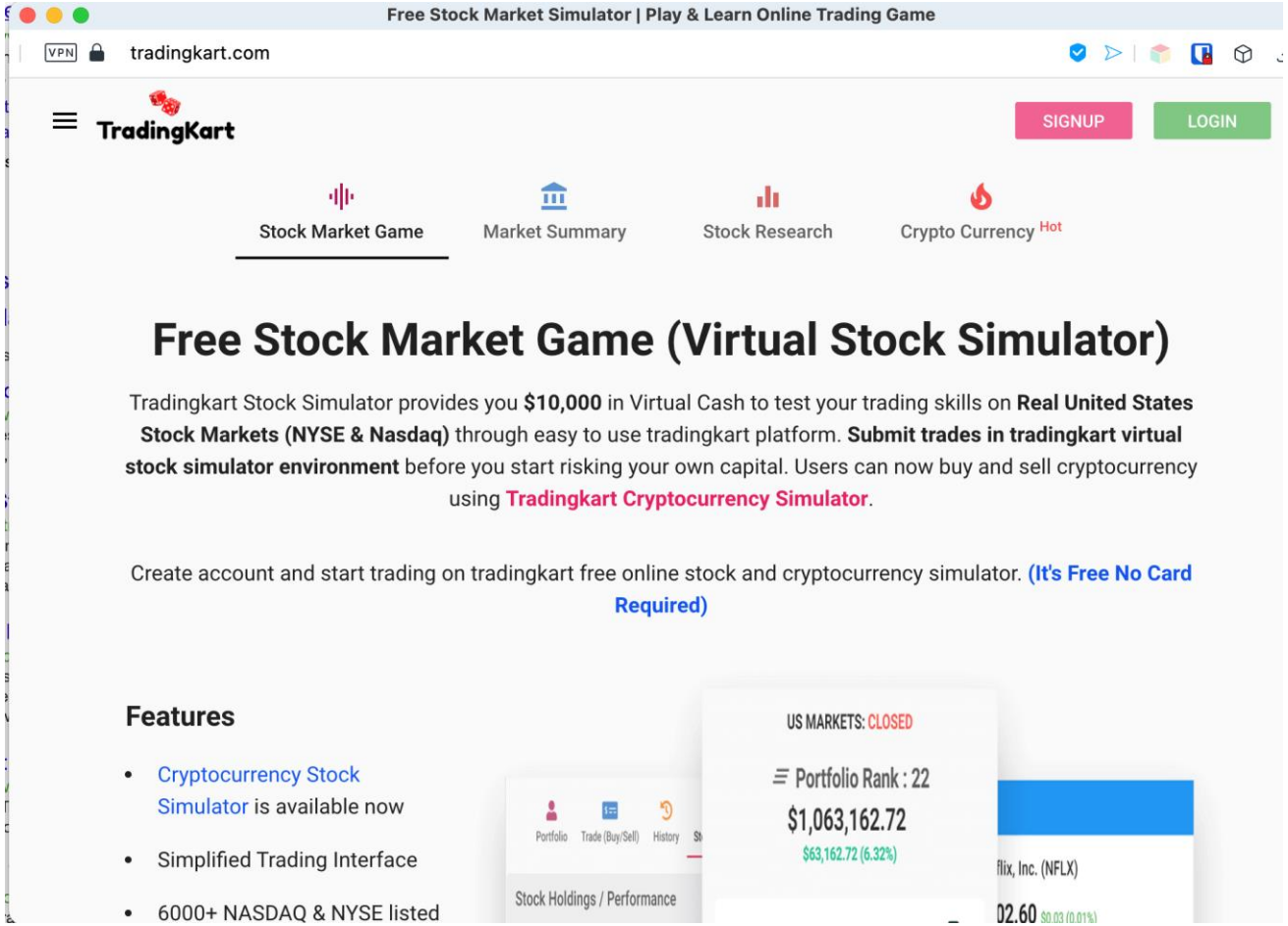
$\alpha \approx 1$ and $\beta \approx 2.7$



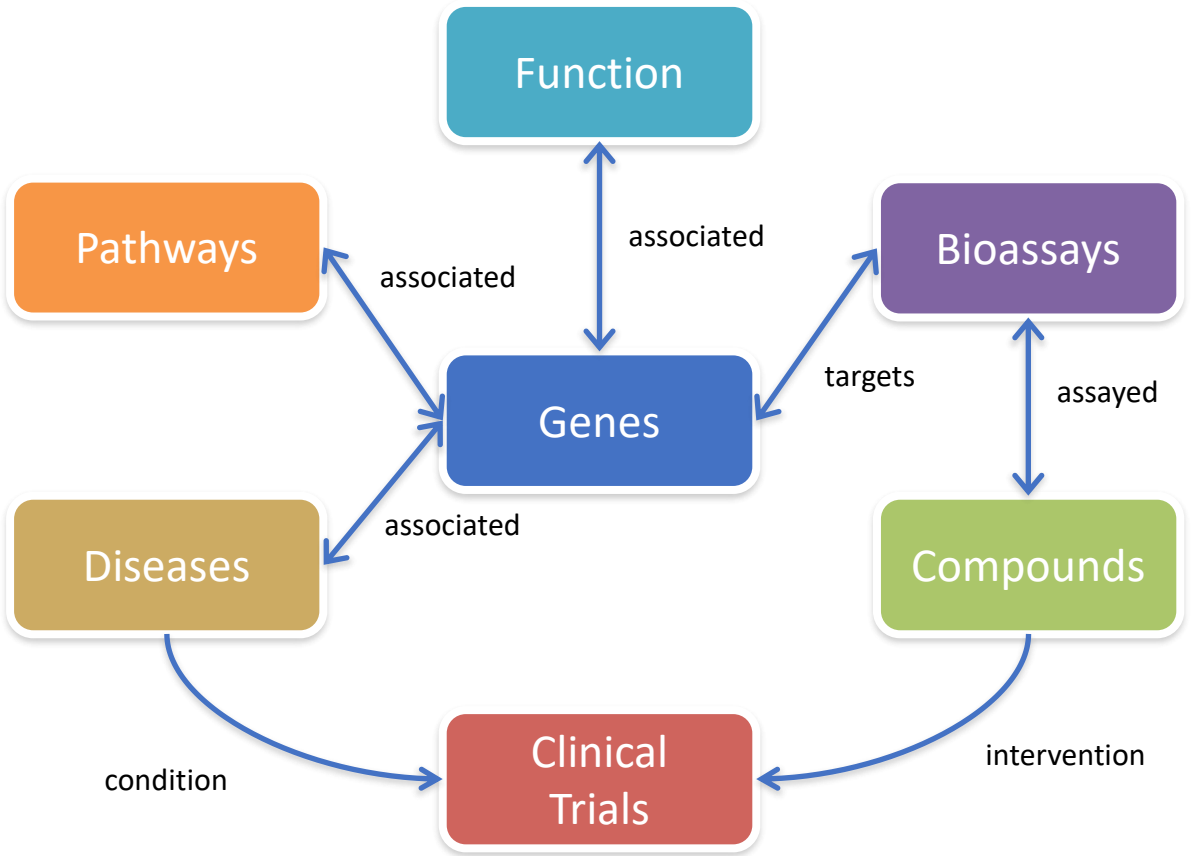
From: Piantadosi (2015). *Zipf's word frequency law in natural language: A critical review and future directions*. Psychon Bul Rev 21(5): 1112-1130

Stock market / Trading Bots

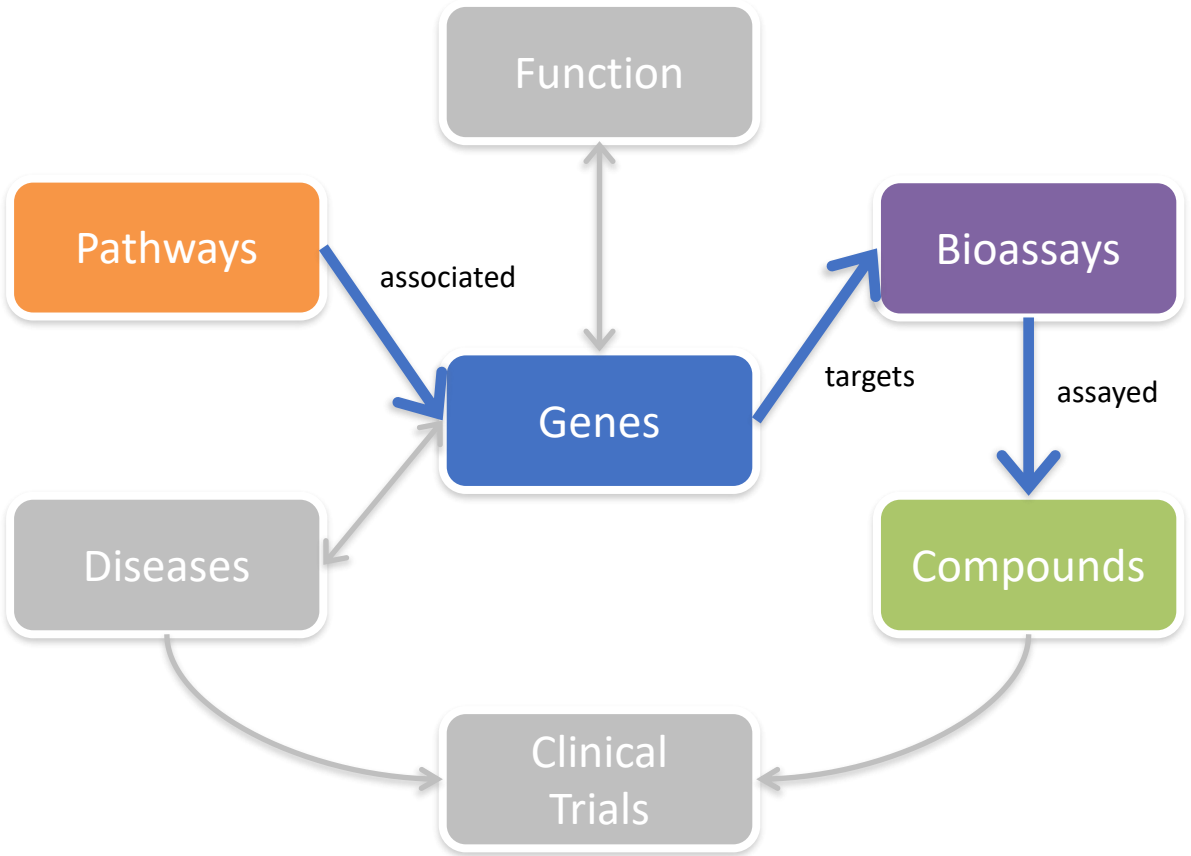
- Manage portfolios
- Buy / Sell / Trade
- Develop REST APIs
- Explore trading strategies with bots



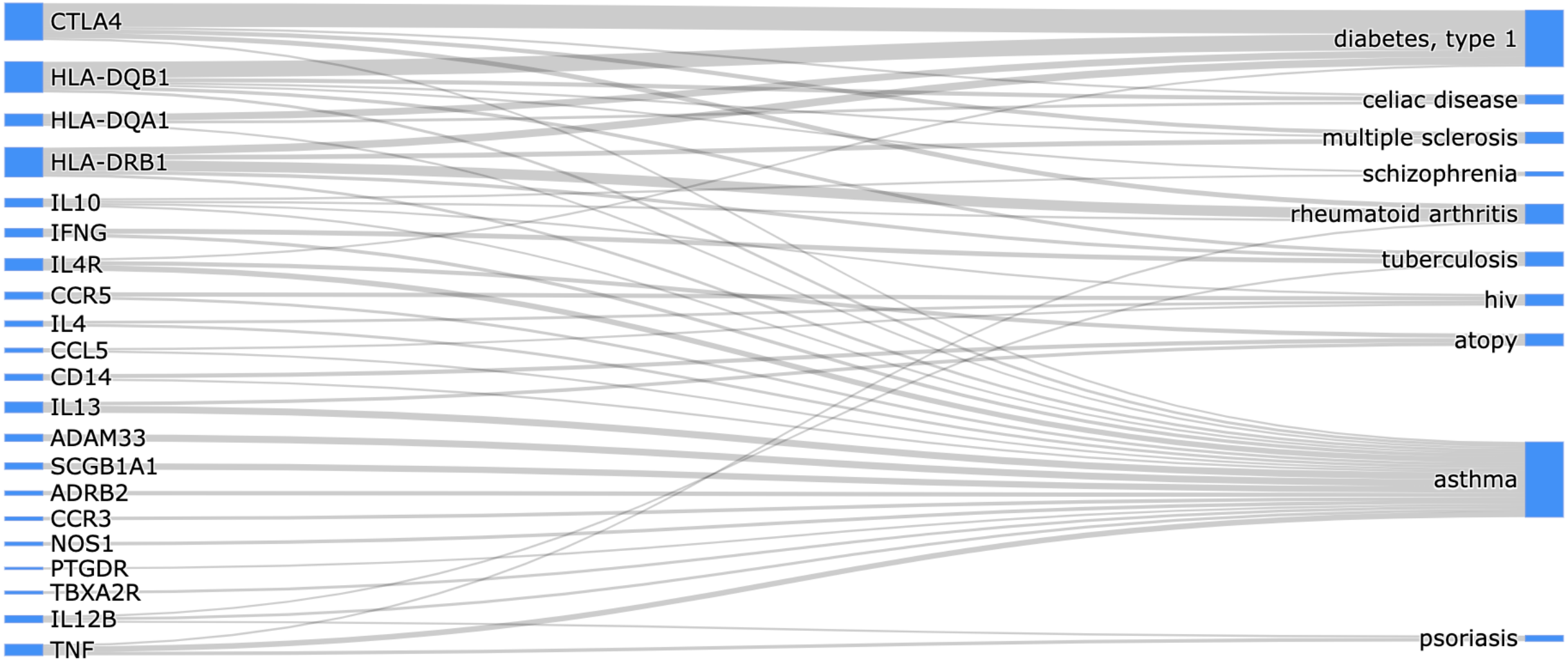
Biological Knowledge is Connected



Query: What compounds activate genes association with pathway X?



Biolink


CTLA4
HLA-DQB1
HLA-DQA1
HLA-DRB1
IL10
IFNG
IL4R
CCR5
IL4
CCL5
CD14
IL13
ADAM33
SCGB1A1
ADRB2
CCR3
NOS1
PTGDR
TBXA2R
IL12B
TNF
diabetes, type 1
celiac disease
multiple sclerosis
schizophrenia
rheumatoid arthritis
tuberculosis
hiv
atopy
asthma
psoriasis

The BioLink DataTable Widget

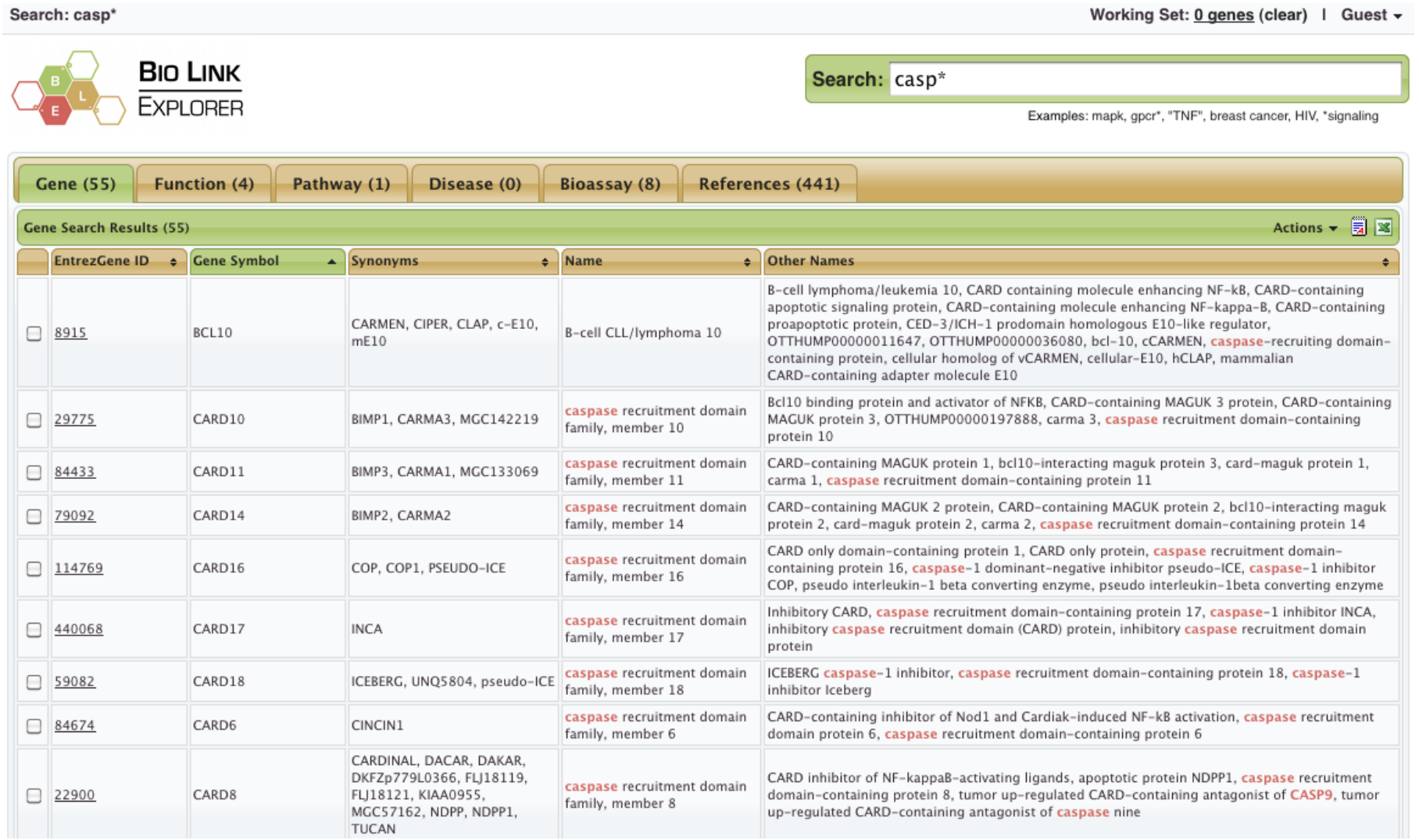
A configurable container of data

- Title
- Description (help)
- Default Sorting
- Selection ON/OFF
- Actions
- Column order
- Column headers
- URL configuration
- Text wrapping
- Export ON/OFF

Associated Genes (25) Actions

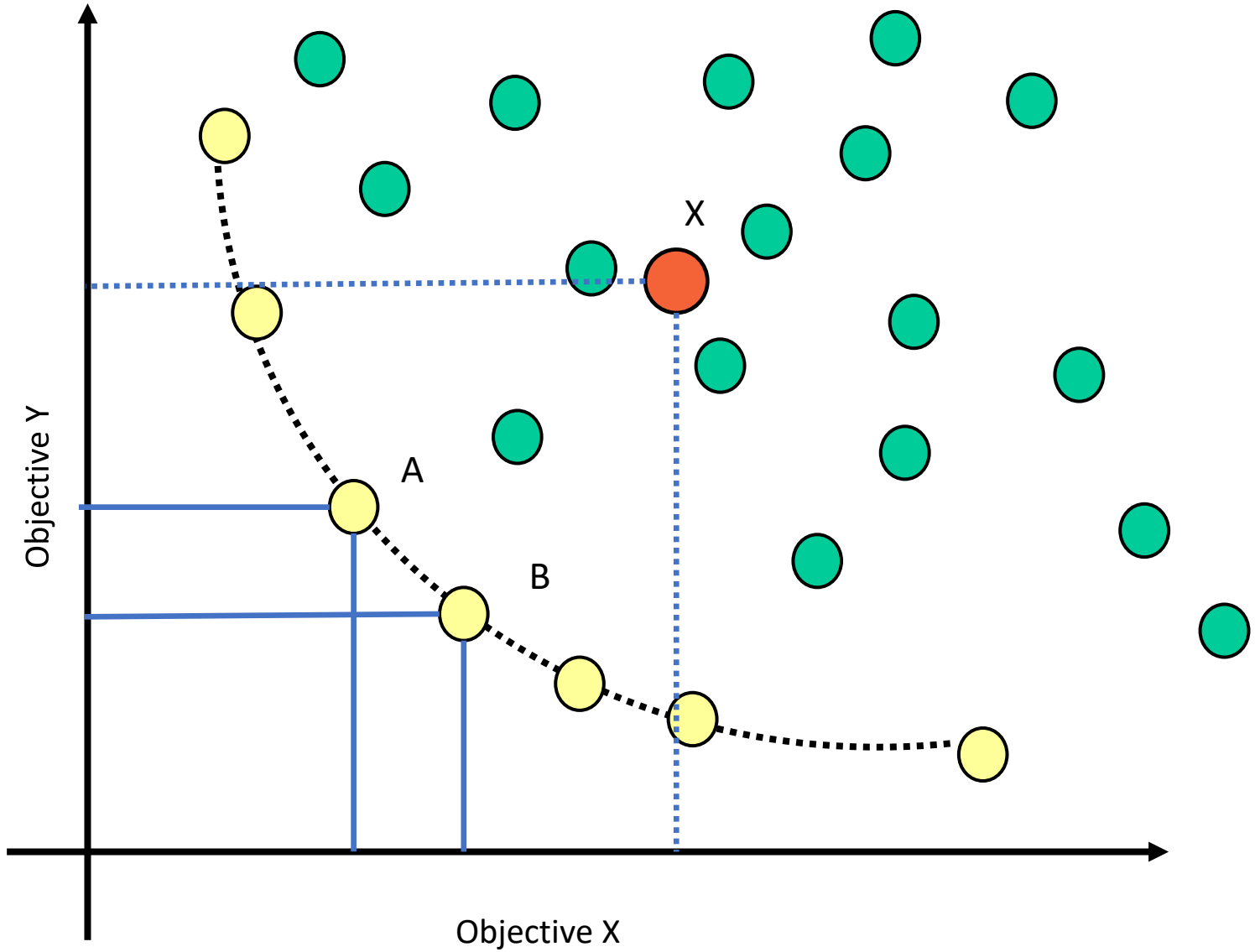
	EntrezGene ID	Gene Symbol	Gene Name
☐	329	BIRC2	baculoviral IAP repeat containing 2
☑	330	BIRC3	baculoviral IAP repeat containing 3
☐	841	CASP8	caspase 8, apoptosis-related cysteine peptidase
☑	857	CAV1	caveolin 1, caveolae protein, 22kDa
☑	1540	CYLD	cylindromatosis (turban tumor syndrome)
☐	4214	MAP3K1	mitogen-activated protein kinase kinase kinase 1
☐	4215	MAP3K3	mitogen-activated protein kinase kinase kinase 3
☑	4217	MAP3K5	mitogen-activated protein kinase kinase kinase 5
☐	5606	MAP2K3	mitogen-activated protein kinase kinase 3
☐	5609	MAP2K7	mitogen-activated protein kinase kinase 7
☐	6609	SMPD1	sphingomyelin phosphodiesterase 1, acid lysosomal
☐	6610	SMPD2	sphingomyelin phosphodiesterase 2, neutral membrane (neutral sphingomyelinase)
☑	6772	STAT1	signal transducer and activator of transcription 1, 91kDa
☑	6868	ADAM17	ADAM metallopeptidase domain 17
☐	7128	TNFAIP3	tumor necrosis factor, alpha-induced protein 3
☐	7186	TRAF2	TNF receptor-associated factor 2
☐	7295	TXN	thioredoxin

An Example Biolink Search



EntrezGene ID	Gene Symbol	Synonyms	Name	Other Names
8915	BCL10	CARMEN, CIPER, CLAP, c-E10, mE10	B-cell CLL/lymphoma 10	B-cell lymphoma/leukemia 10, CARD containing molecule enhancing NF-kB, CARD-containing apoptotic signaling protein, CARD-containing molecule enhancing NF-kappa-B, CARD-containing proapoptotic protein, CED-3/ICH-1 prodomain homologous E10-like regulator, OTTHUMP00000011647, OTTHUMP00000036080, bcl-10, cCARMEN, caspase-recruiting domain-containing protein, cellular homolog of vCARMEN, cellular-E10, hCLAP, mammalian CARD-containing adapter molecule E10
29775	CARD10	BIMP1, CARMA3, MGC142219	caspase recruitment domain family, member 10	Bcl10 binding protein and activator of NFKB, CARD-containing MAGUK 3 protein, CARD-containing MAGUK protein 3, OTTHUMP00000197888, carma 3, caspase recruitment domain-containing protein 10
84433	CARD11	BIMP3, CARMA1, MGC133069	caspase recruitment domain family, member 11	CARD-containing MAGUK protein 1, bcl10-interacting maguk protein 3, card-maguk protein 1, carma 1, caspase recruitment domain-containing protein 11
79092	CARD14	BIMP2, CARMA2	caspase recruitment domain family, member 14	CARD-containing MAGUK 2 protein, CARD-containing MAGUK protein 2, bcl10-interacting maguk protein 2, card-maguk protein 2, carma 2, caspase recruitment domain-containing protein 14
114769	CARD16	COP, COP1, PSEUDO-ICE	caspase recruitment domain family, member 16	CARD only domain-containing protein 1, CARD only protein, caspase recruitment domain-containing protein 16, caspase-1 dominant-negative inhibitor pseudo-ICE, caspase-1 inhibitor COP, pseudo interleukin-1 beta converting enzyme, pseudo interleukin-1beta converting enzyme
440068	CARD17	INCA	caspase recruitment domain family, member 17	Inhibitory CARD, caspase recruitment domain-containing protein 17, caspase-1 inhibitor INCA, inhibitory caspase recruitment domain (CARD) protein, inhibitory caspase recruitment domain protein
59082	CARD18	ICEBERG, UNQ5804, pseudo-ICE	caspase recruitment domain family, member 18	ICEBERG caspase-1 inhibitor, caspase recruitment domain-containing protein 18, caspase-1 inhibitor Iceberg
84674	CARD6	CINCIN1	caspase recruitment domain family, member 6	CARD-containing inhibitor of Nod1 and Cardiak-induced NF-kB activation, caspase recruitment domain protein 6, caspase recruitment domain-containing protein 6
22900	CARD8	CARDINAL, DACAR, DAKAR, DKFZp779L0366, FLJ18119, FLJ18121, KIAA0955, MGC57162, NDPP, NDPP1, TUCAN	caspase recruitment domain family, member 8	CARD inhibitor of NF-kappaB-activating ligands, apoptotic protein NDPP1, caspase recruitment domain-containing protein 8, tumor up-regulated CARD-containing antagonist of CASP9, tumor up-regulated CARD-containing antagonist of caspase nine

Evolving Tradeoffs



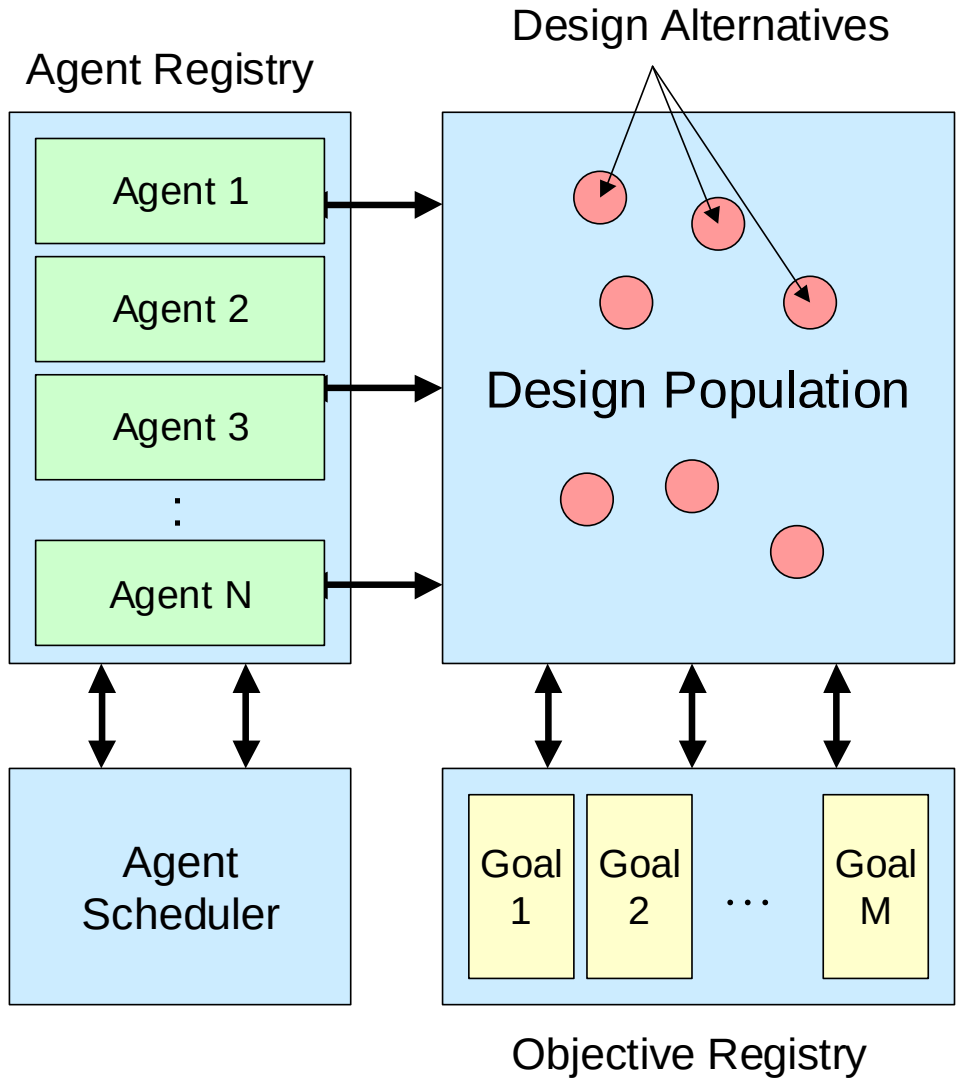
An architecture for multi-objective optimization using evolutionary computing

Classes of agents:

Creator Agent: Adds a design constructed from scratch

Improver Agent: Copies and then modifies an existing design

Destroyer Agent: Removes one or more unpromising designs



General Distance Measure

$$\text{diff} = w_1|\Delta A_1|^r + w_2|\Delta A_2|^r + \ldots + w_n|\Delta A_n|^r$$

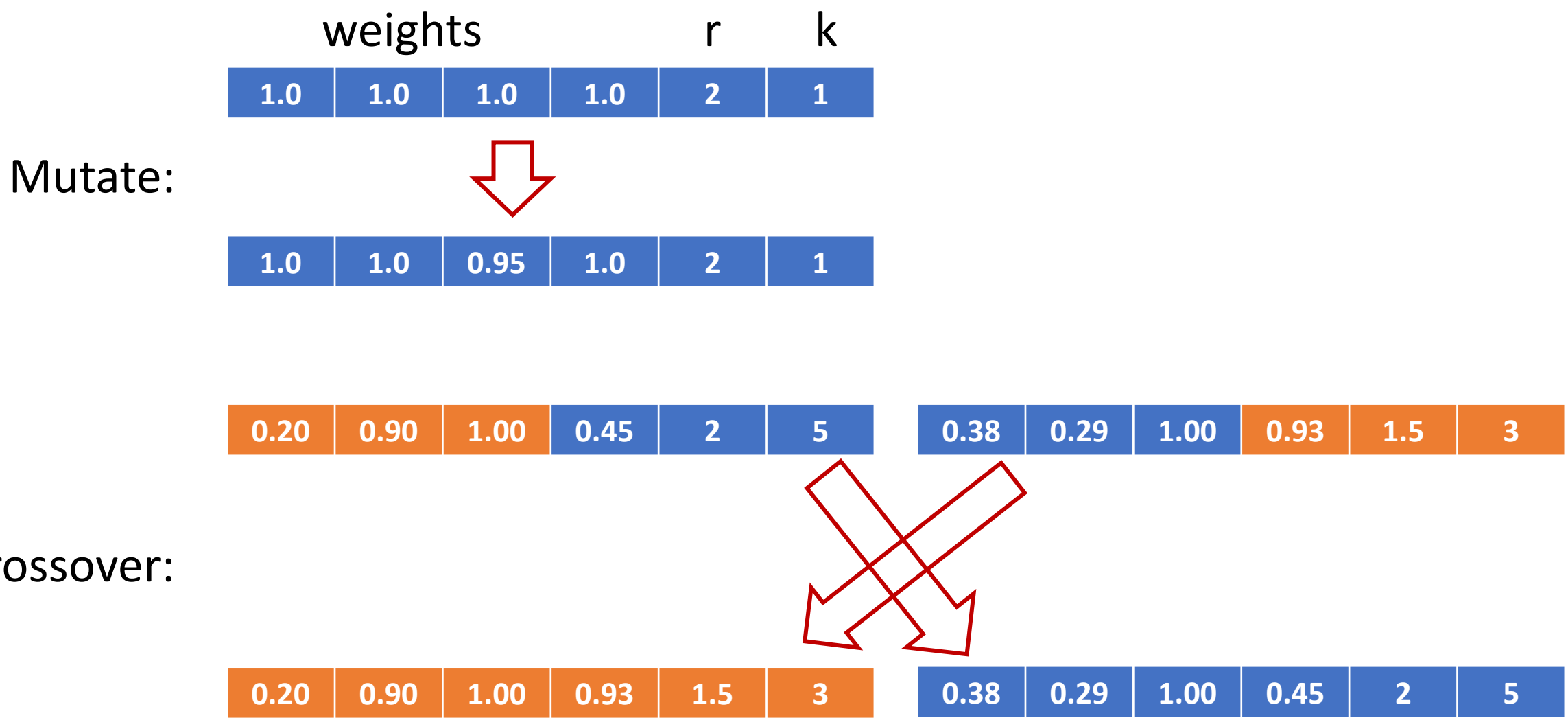
where

ΔA_i is the difference with respect to feature i,

w_k is a feature weighting factor (0.0 to 1.0), and

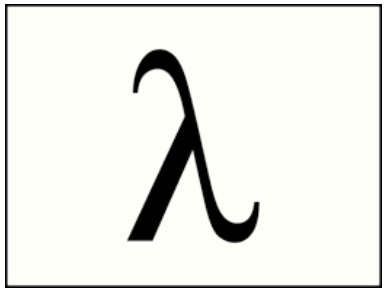
r is an exponent (> 0.0)

Tweaking / modifying a solution



An overview of functional programming

- What is functional programming?
- Immutable data structures
- Building blocks: *filter(), map(), reduce()*
- Parallel processing, multi-processing, futures
- Emphasis on recursion rather than looping
- list comprehensions everywhere!
- Functions are first-class objects: Can be input to and output from other functions or be elements of a collection.



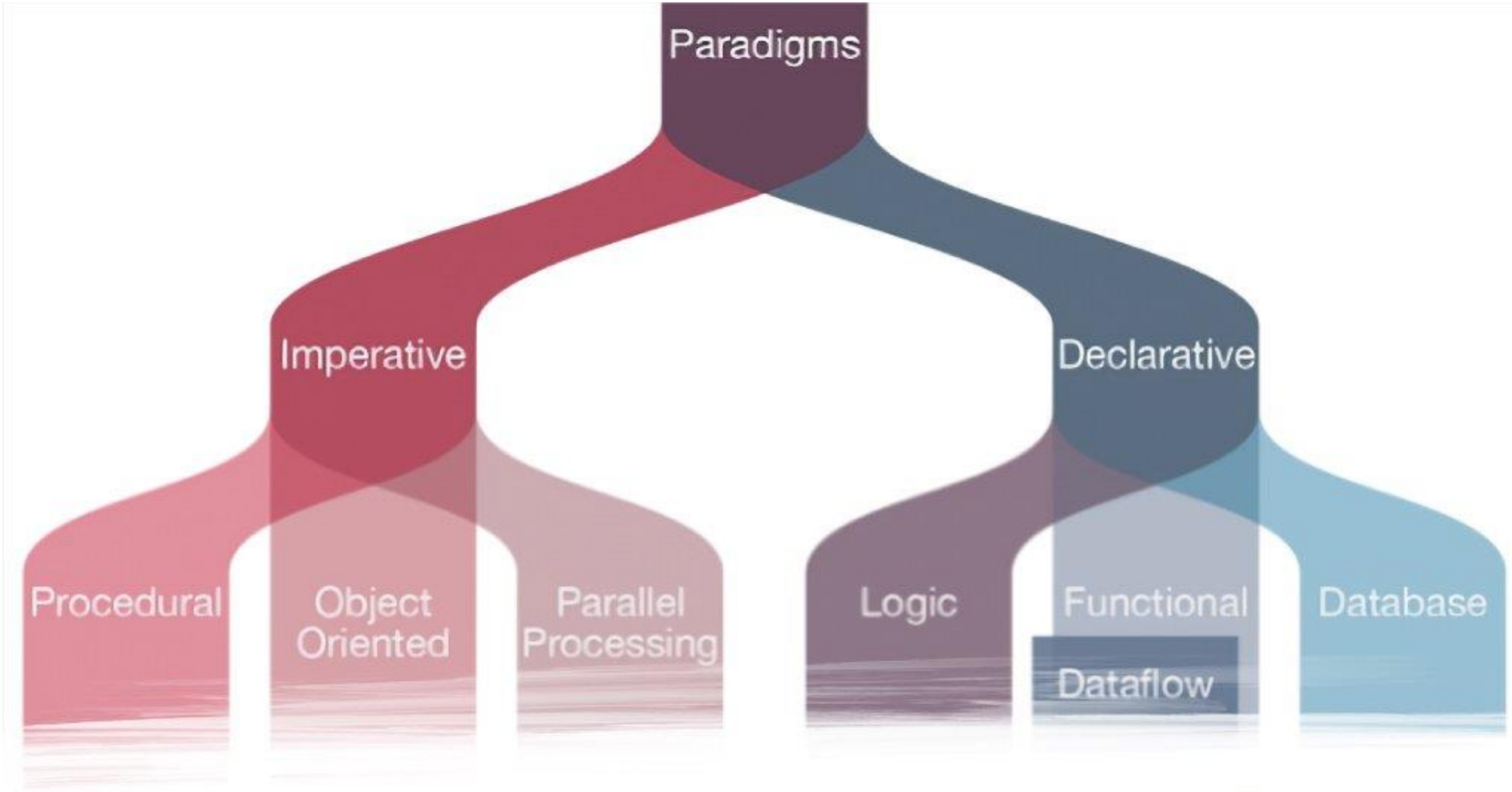
What is functional programming?

Functional programming is a programming *paradigm* or programming *style* that emphasizes computation on *immutable (unchanging)* data structures through the evaluation of functions while avoiding side-effects.

Advantages:

- Reduced number of bugs
- More maintainable
- Elegant expression of computation
- Easier to reason about multi-threaded / parallel programs because interactions with data are via immutable data structures.

Programming Paradigms



Source: https://www.watelectronics.com/types-of-programming-languages-with-differences/

Mutable vs. immutable data structures

Mutable	Immutable
list	string
dictionary	tuple
	named tuple

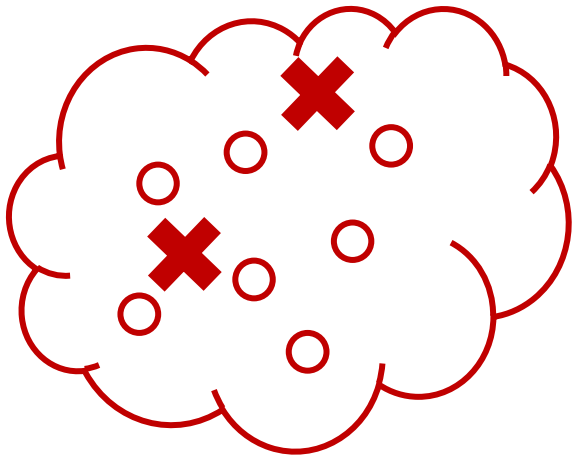
filter()

filter: Filter a collection according to which elements satisfy a function

filter(function, collection) ➔ *{elem* ∈ Collection | function(elem) = True}

L = [1,2,3,4,5]

filter(lambda x: x%2 == 0, L) → [2,4] # or rather an iterator

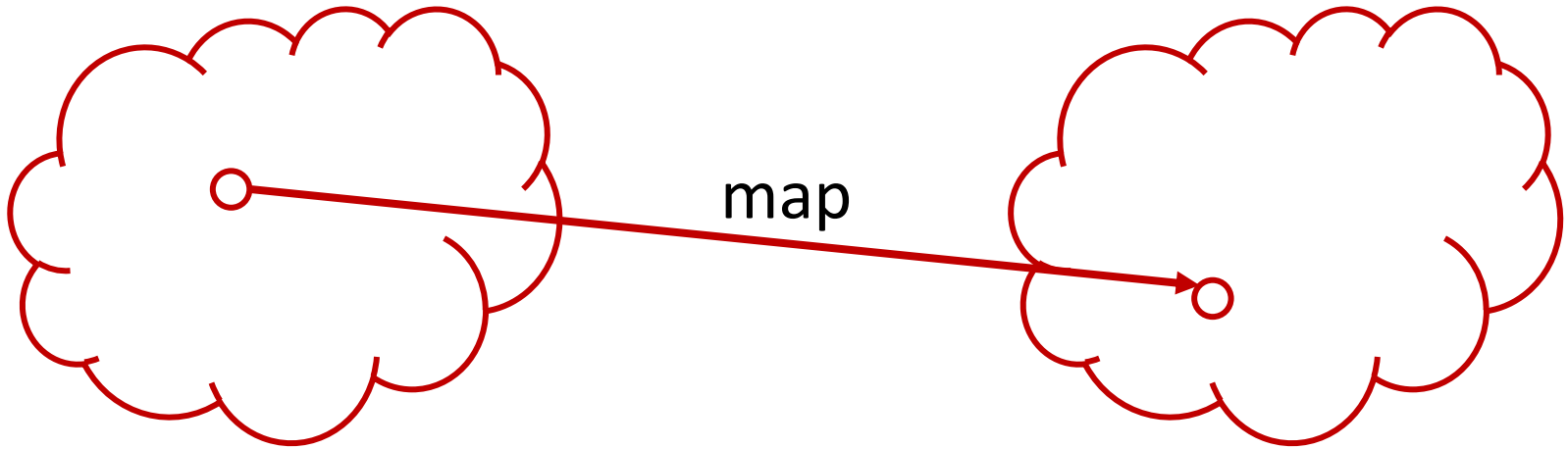


filter

map()

map: Maps elements of a collection to a new collection by applying a function to each element.

map(lambda x: x2, L) → [1,4,9,16,25]**

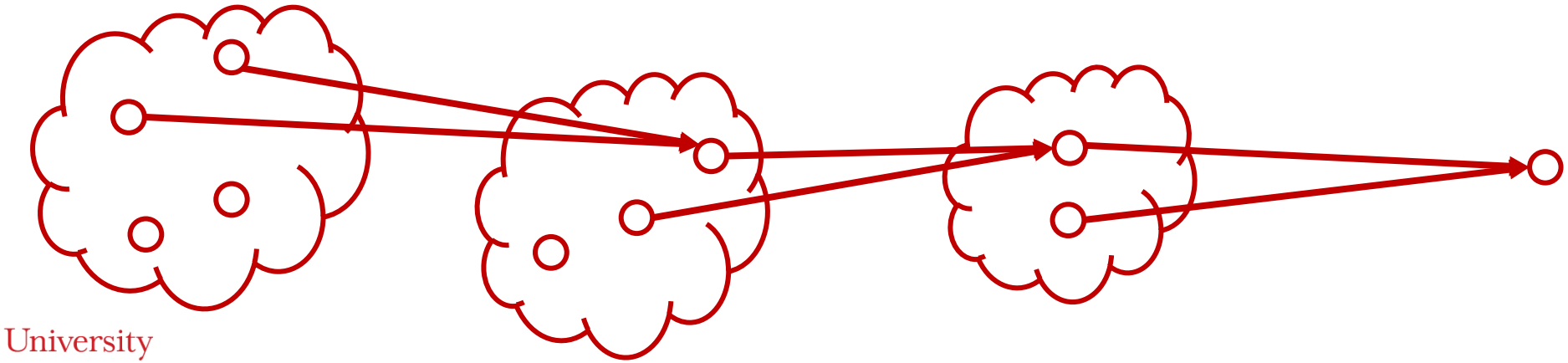


reduce()

reduce: Apply a function of two arguments cumulatively to the items of a sequence, from left to right, so as to reduce the sequence to a single value..

L = [1,2,3,4,5]
reduce(function, collection, initial_value)
reduce(lambda x,y: x+y, L) → [**3**,3,4,5] → [**6**,4,5] → [**10**,5] → **15**

NOTE: from functools import reduce



Intro to Parallel Programming

What is machine learning?

- Can we really make our machines (computers) learn?
- "Secret sauce" is **data, and lots of it**
- **Rather than programming expertise into our applications, we program them to learn from data**
- Build working machine-learning models then use them to make **remarkably accurate predictions**

Prediction

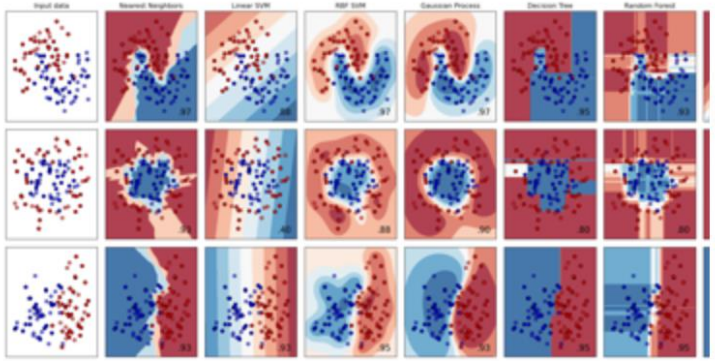
- Improve **weather forecasting** to save lives, minimize injuries and property damage
- Improve **cancer diagnoses** and **treatment regimens** to save lives
- Improve **business forecasts** to maximize profits and secure people's jobs
- **Detect fraudulent credit-card purchases** and **insurance claims**
- Predict **customer "churn"**, what prices houses are likely to sell for, ticket sales of new movies, and anticipated revenue of new products and services
- Predict the **best strategies for coaches and players** to use to win more games and championships
- All of these kinds of predictions are happening today with machine learning.

Machine Learning Approaches

Classification

Identifying which category an object belongs to.

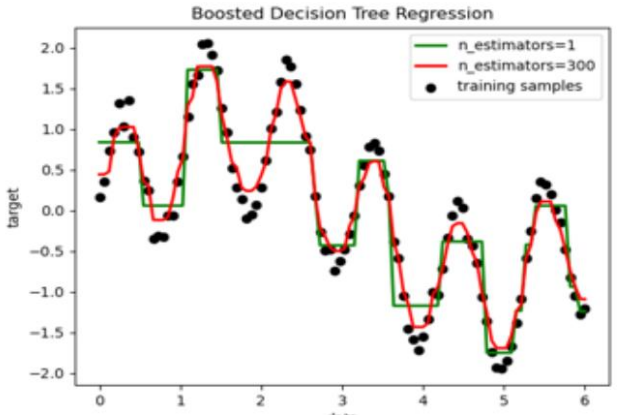
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

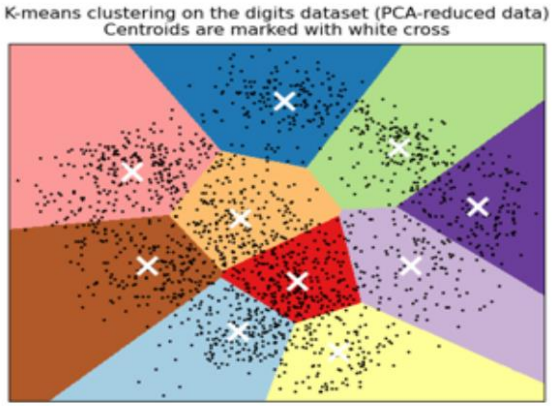
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



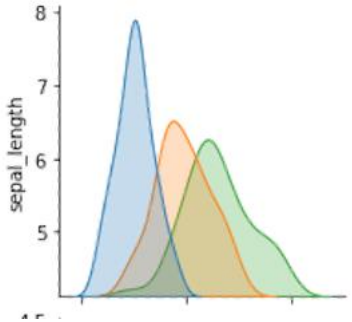
Popular Machine Learning Applications

Anomaly detection	Data mining social media (like Facebook, Twitter, LinkedIn)	Predict mortgage loan defaults
Chatbots	Detecting objects in scenes	Natural language translation (English to Spanish, French to Japanese, etc.)
Classifying emails as spam or not spam	Detecting patterns in data	Recommender systems ("people who bought this product also bought…")
Classifying news articles as sports, financial, politics, etc.	Diagnostic medicine	Self-Driving cars (more generally, autonomous vehicles)
Computer vision and image classification	Facial recognition	Sentiment analysis (like classifying movie reviews as positive, negative or neutral)
Credit-card fraud detection	Handwriting recognition	Spam filtering
Customer churn prediction	Insurance fraud detection	Time series predictions like stock-price forecasting and weather forecasting
Data compression	Intrusion detection in computer networks	Voice recognition
Data exploration	Marketing: Divide customers into clusters	

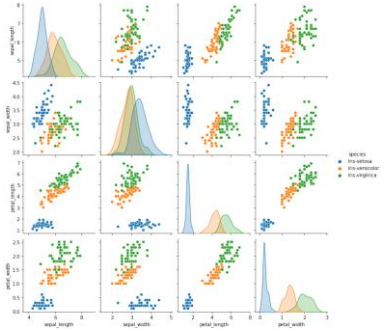
Machine Learning Recipe

```
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
```

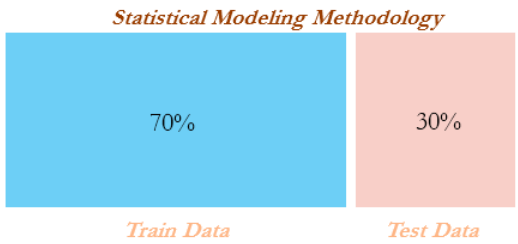
load
data



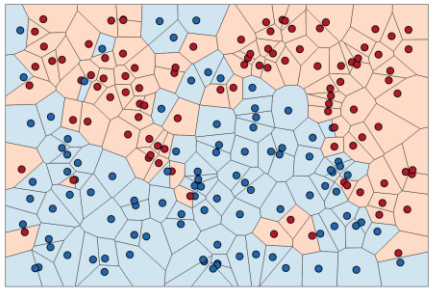
explore



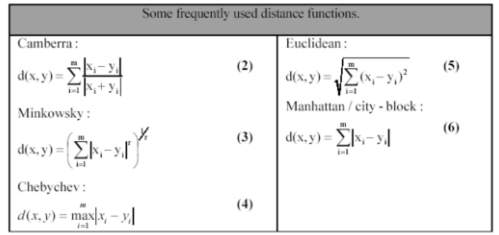
transform



training &
test split



model
building

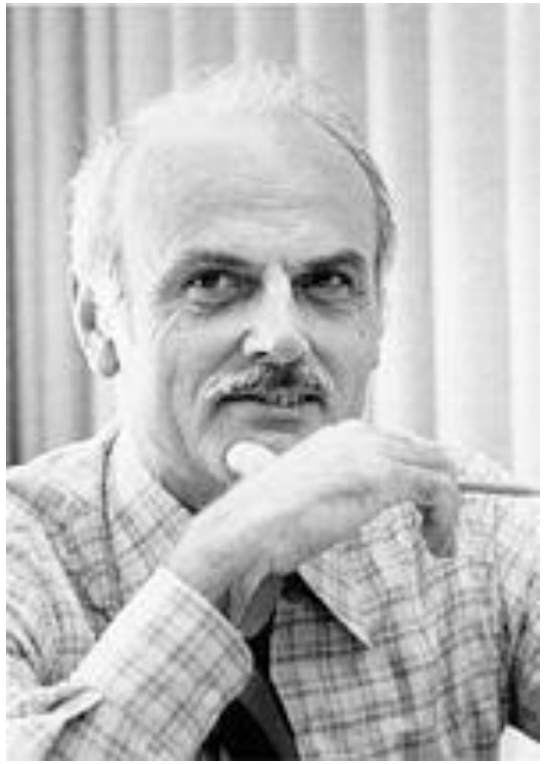


model
tuning

Databases

Relational and Non-Relational (NoSQL) Data Storage and Retrieval

Codd, 1970. The origins of the relational model



Edgar Frank Codd
1923-2003

Information Retrieval

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Relational Data Model

Entities

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

A column or ***attribute*** or **field** providing more details about the particular entity.

- Name
- Datatype
- Other attributes

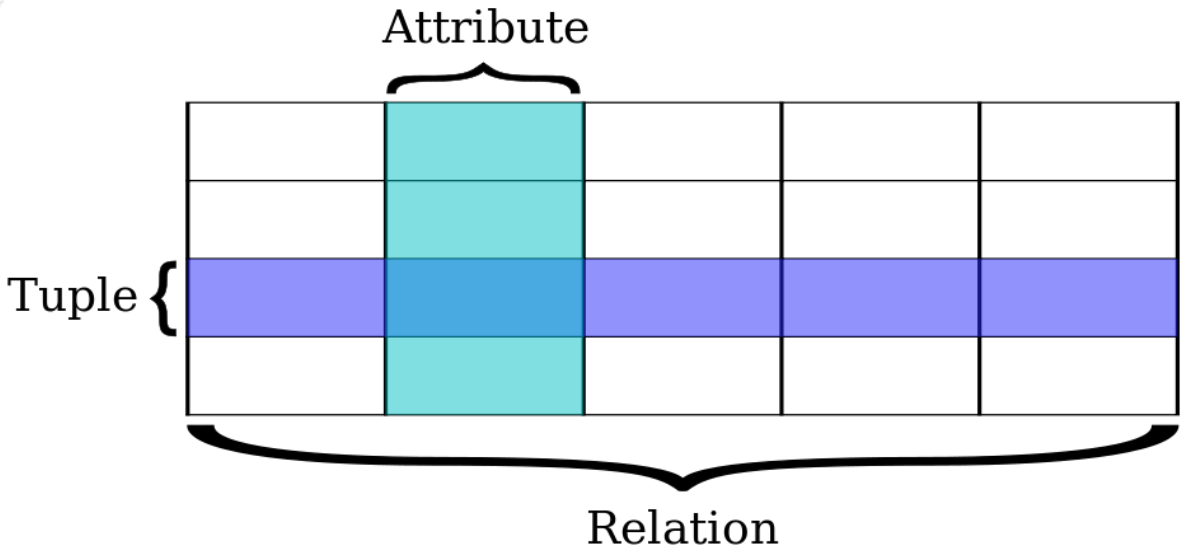
A row or ***record*** describing one instance of the entity

Three pillars of Relational Databases

relational model

SQL

ACID



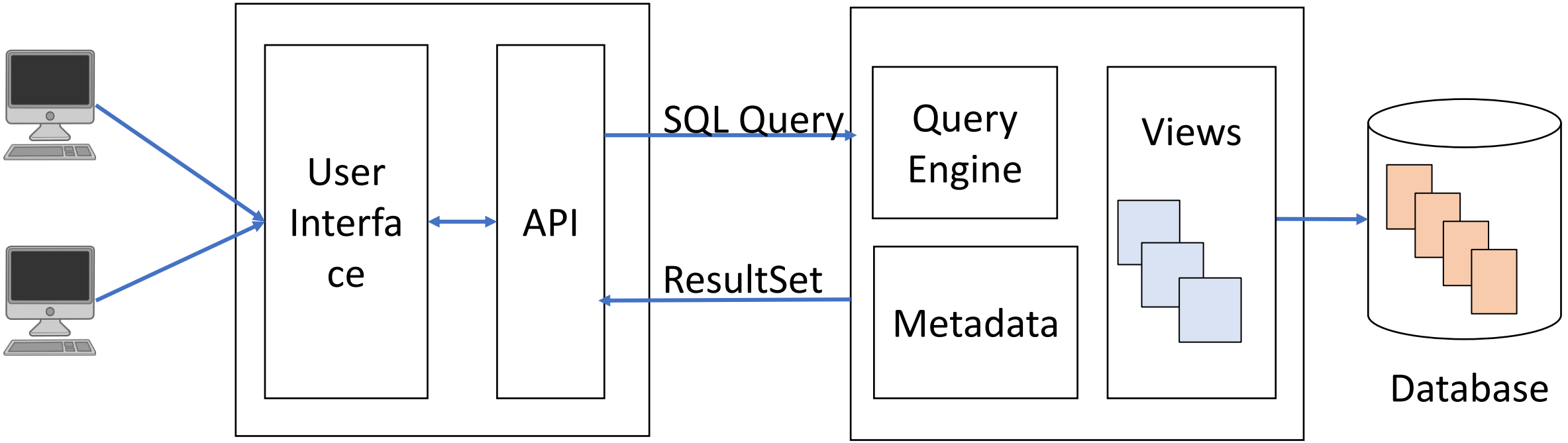
DB-Engines Ranking

371 systems in ranking, May 2021

Rank May 2021	Rank Apr 2021	Rank May 2020	DBMS	Database Model	Score May 2021	Score Apr 2021	Score May 2020
1.	1.	1.	Oracle	Relational, Multi-model	1269.94	-4.98	-75.50
2.	2.	2.	MySQL	Relational, Multi-model	1236.38	+15.69	-46.26
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	992.66	-15.30	-85.64
4.	4.	4.	PostgreSQL	Relational, Multi-model	559.25	+5.73	+44.45
5.	5.	5.	MongoDB	Document, Multi-model	481.01	+11.04	+42.02
6.	6.	6.	IBM Db2	Relational, Multi-model	166.66	+8.88	+4.02
7.	7.	↑ 8.	Redis	Key-value, Multi-model	162.17	+6.28	+18.69
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	155.35	+3.18	+6.23
9.	9.	9.	SQLite	Relational	126.69	+1.64	+3.66
10.	10.	10.	Microsoft Access	Relational	115.40	-1.33	-4.50

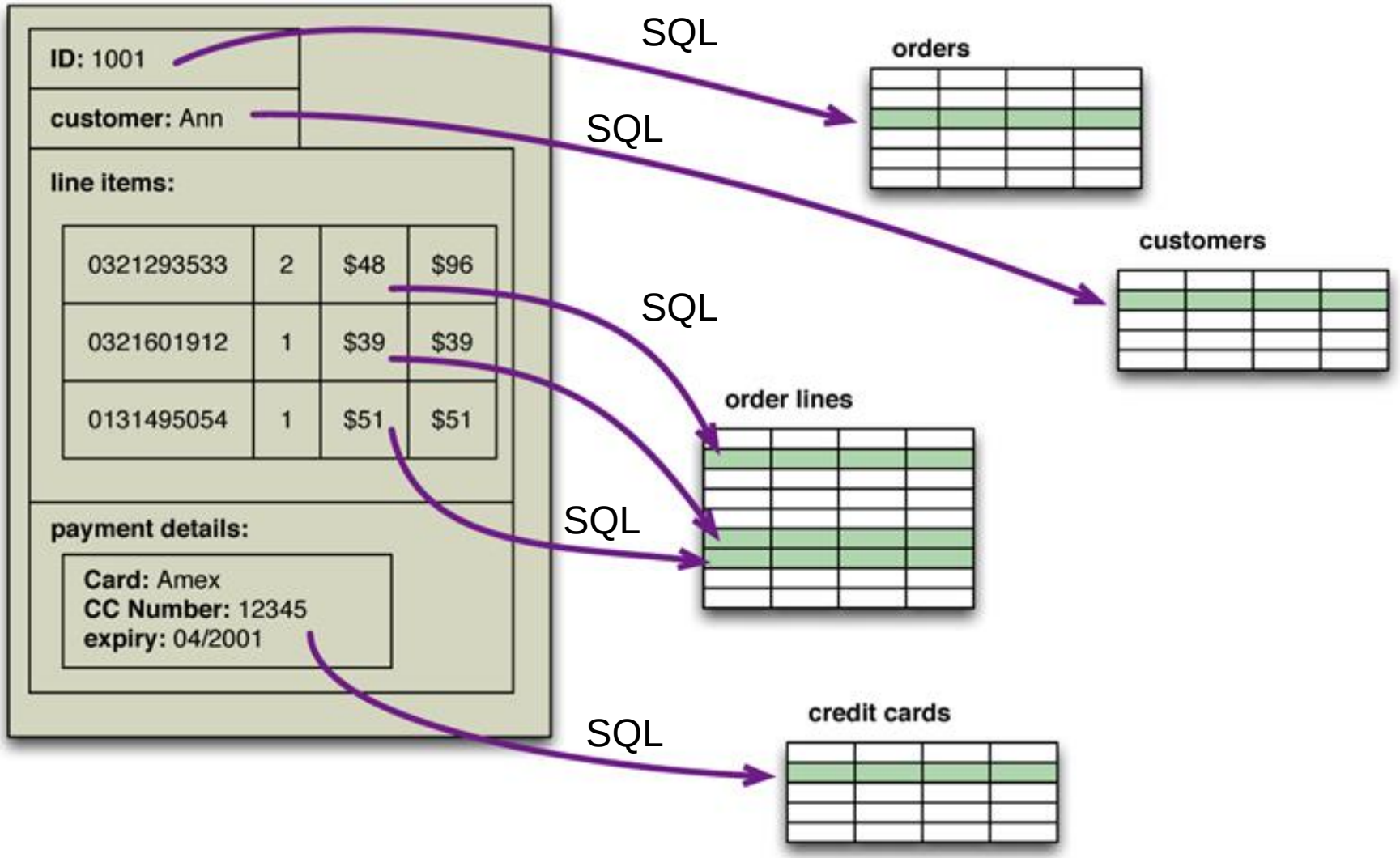
https://db-engines.com/en/ranking

Database Application

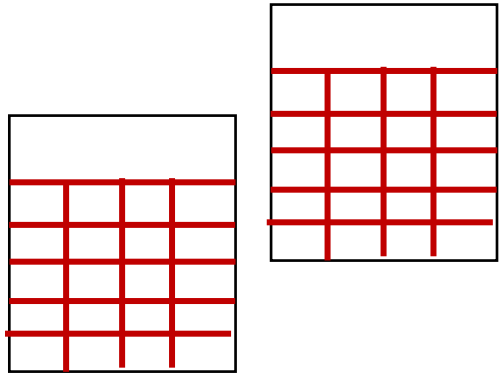


(Usually running on a separate server)

Why NoSQL?: Impedance mismatch problem



Types of models

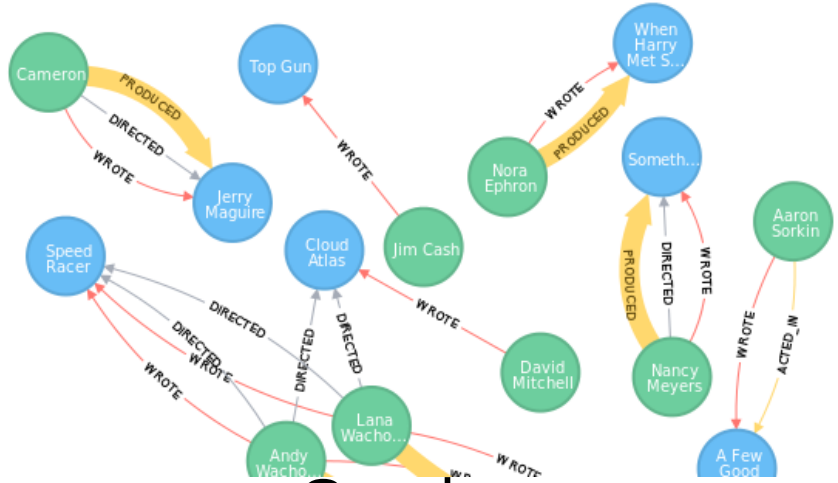


Relational

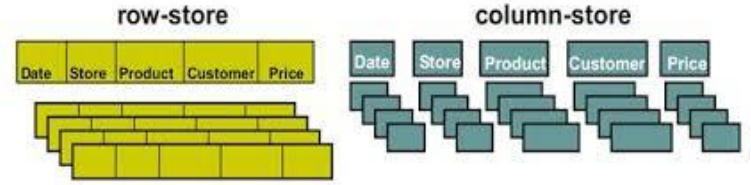
Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

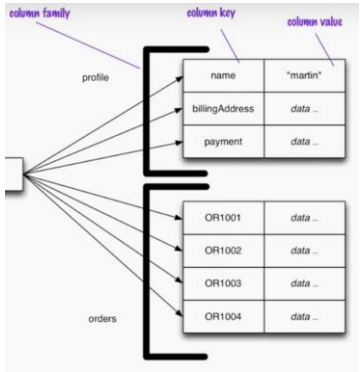
Key-Value

Document



Graphs

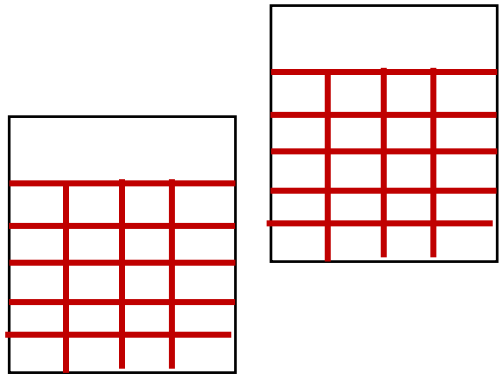




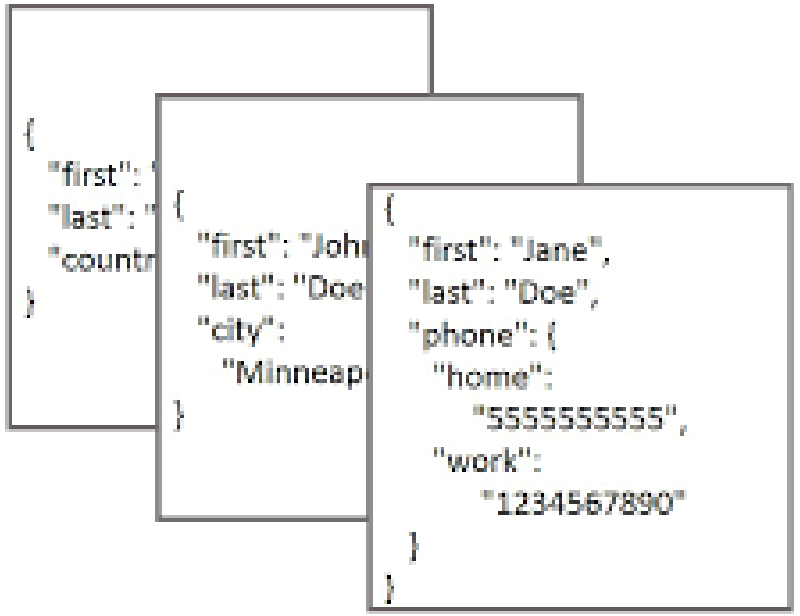
Column-oriented storage
Column-family storage

Document databases (MongoDB): a happy medium

Relational Databases
Pre-defined / rigid schema
SQL: Structured Query Language



Document Databases
Data is self-describing and flexible
Query by content



Key-Value Stores
Data is simple hash tables
Query by key only (with exceptions)

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

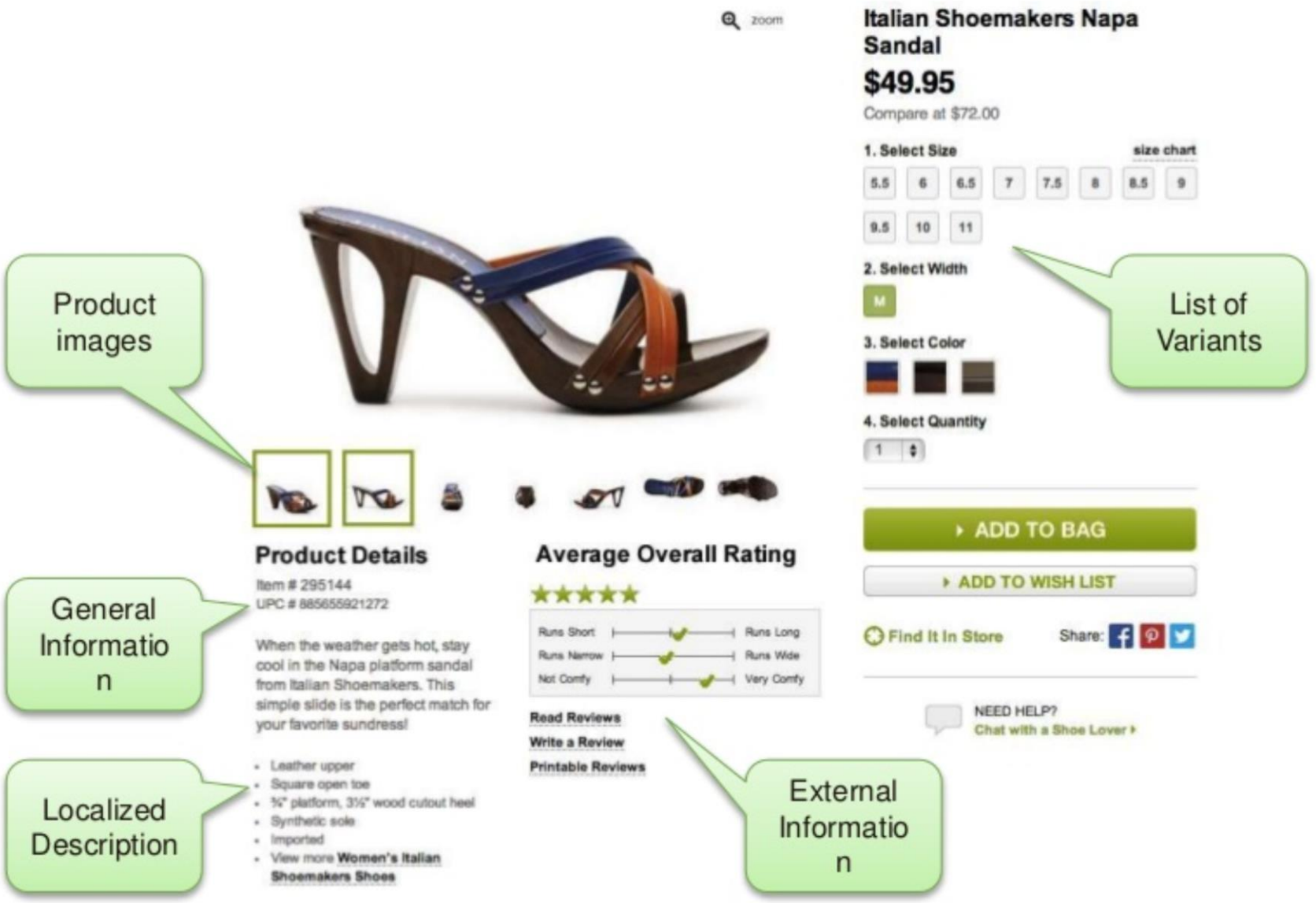
Document Databases

A **document database** is a non-relational database that stores data as structured documents, usually XML or JSON formats.

Designed to be *simple*, *flexible*, and *scalable*

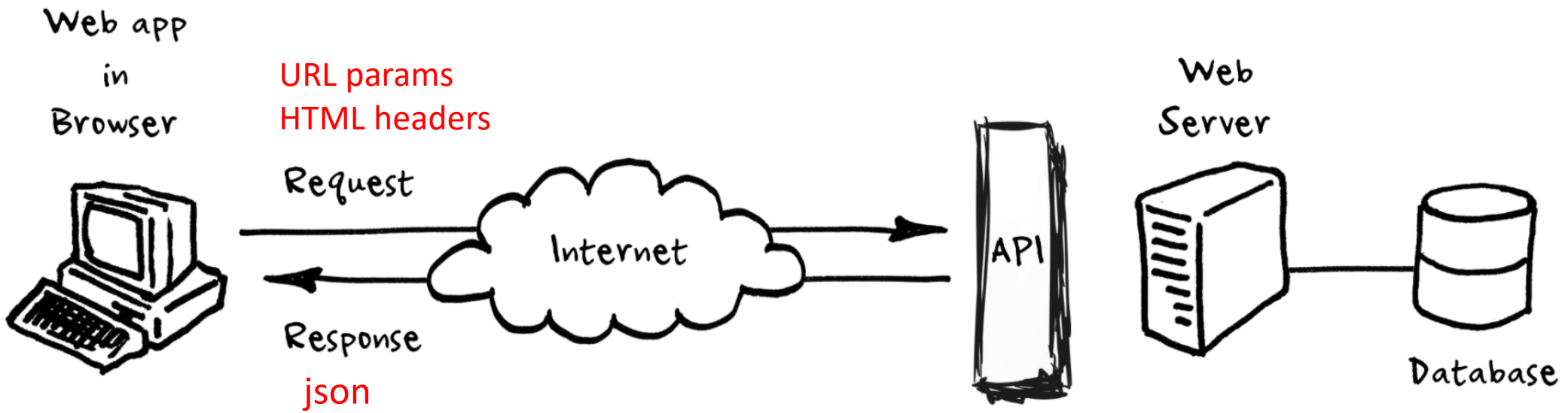
```
{ "users":[
        {
            "firstName":"Ray",
            "lastName":"Villalobos",
            "joined": {
                "month":"January",
                "day":12,
                "year":2012
            }
        },
        {
            "firstName":"John",
            "lastName":"Jones",
            "joined": {
                "month":"April",
                "day":28,
                "year":2010
            }
        }
]}
```

Product Catalogs / e-commerce websites



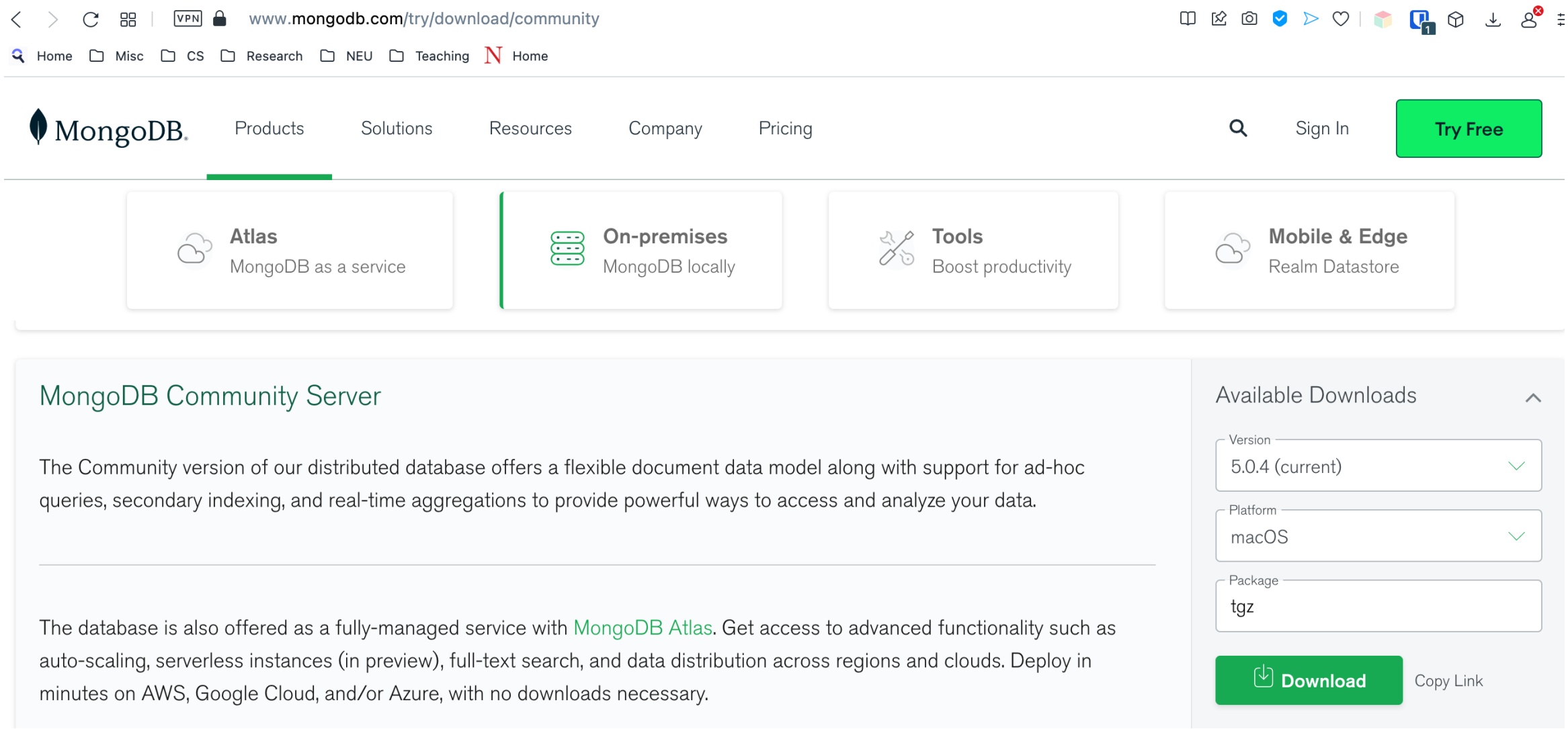
Web-Service APIs

Application Programming Interfaces (APIs) provide a general means of interacting with other programs. More specifically, many websites will provide data via a web API -- this means we can make *requests* to remote applications / websites for data. Here's what this looks like schematically.



The way that we do this in Python is via the `requests` module. The idea is that the Response (in the above depiction) is nicely formatted `json`, which feels a lot like the dictionaries you now know and love.

MongoDB Installation



Or on Mac OSX: $ brew install mongodb

MongoDB Detailed Instructions

https://docs.mongodb.com/manual/administration/install-community/

Install MongoDB > Install MongoDB Community Edition

Install MongoDB Community Edition

These documents provide instructions to install MongoDB Community Edition.

Install on Linux

Install MongoDB Community Edition and required dependencies on Linux.

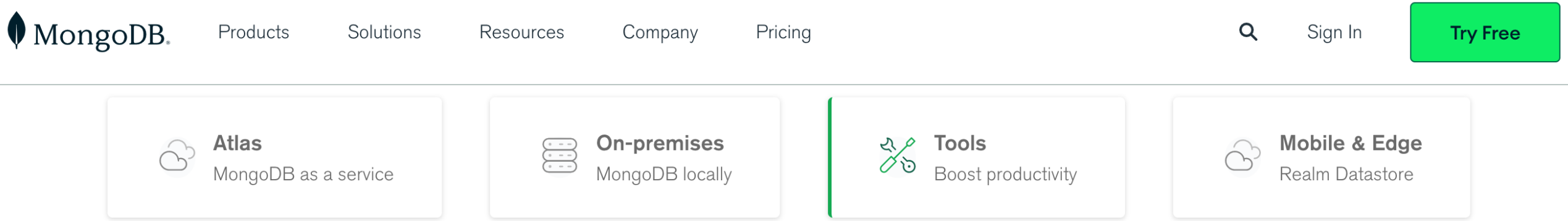
Install on macOS

Install MongoDB Community Edition on macOS systems from Homebrew packages or from MongoDB archives.

Install on Windows

Install MongoDB Community Edition on Windows systems and optionally start MongoDB as a Windows service.

MongoDB Compass: https://www.mongodb.com/try/download/compass



MongoDB Compass

Easily explore and manipulate your database with Compass, the GUI for MongoDB. Intuitive and flexible, Compass provides detailed schema visualizations, real-time performance metrics, sophisticated querying abilities, and much more.

Please note that MongoDB Compass comes in three versions: **a full version** with all features, **a read-only version** without write or delete capabilities, and **an isolated edition**, whose sole network connection is to the MongoDB instance.

For more information, see our documentation pages.

- **Compass**

  The full version of MongoDB Compass, with all features and capabilities.

- **Readonly Edition**

  This version is limited strictly to read operations, with all write and delete capabilities removed.

Available Downloads

Version: 1.29.4 (Stable)

Platform: OS X 64-bit (10.10+)

Package: dmg

Download | Copy Link

Documentation

Archived releases

Running MongoDB

Add $MONGO_HOME to your path

```
export APPS=/Users/rachlin/apps
export MONGO_HOME=$APPS/mongo/mongodb-4.0.7
export PATH=$MONGO_HOME/bin:$PATH
```

Set up a directory for the database storage and run server

```
$ mkdir -p $HOME/data/mongodb
$ mongod --dbpath $HOME/data/mongodb
```

Run the command-line client (or launch Compass)

```
$ mongo
```

Object Serialization

Using Pickle

Serialization / Deserialization

- Convert data type into a linear byte stream
- Useful for storing data or sending over network
- Python has a number of built-in modules for this
  - `marshall`
  - `json`
  - `pickle`

source: realpython.com

The marshall module (No!)

- Oldest of the three serialization modules
- Primarily reads and writes compiled bytecode from Python modules
- `.pyc` files
- Don’t use - mainly used by the interpreter and can have breaking changes

JSON (human-readable serialization)

- Newest of the three serialization modules
- Produces standard JSON output: human-readable
- Works very well with other languages
- Only works with certain data types

pickle (binary serialization)

- Serializes in binary format, not human-readable
- Works out of the box with many Python data types - including custom
- Very fast

pickle protocols

- Six different protocols for `pickle`
- Tied to Python interpreter version (2.3, 3.0, 3.4, 3.8)
- Newer protocols need newer interpreter
- Highest level available will be used, except `protocol=parameter`