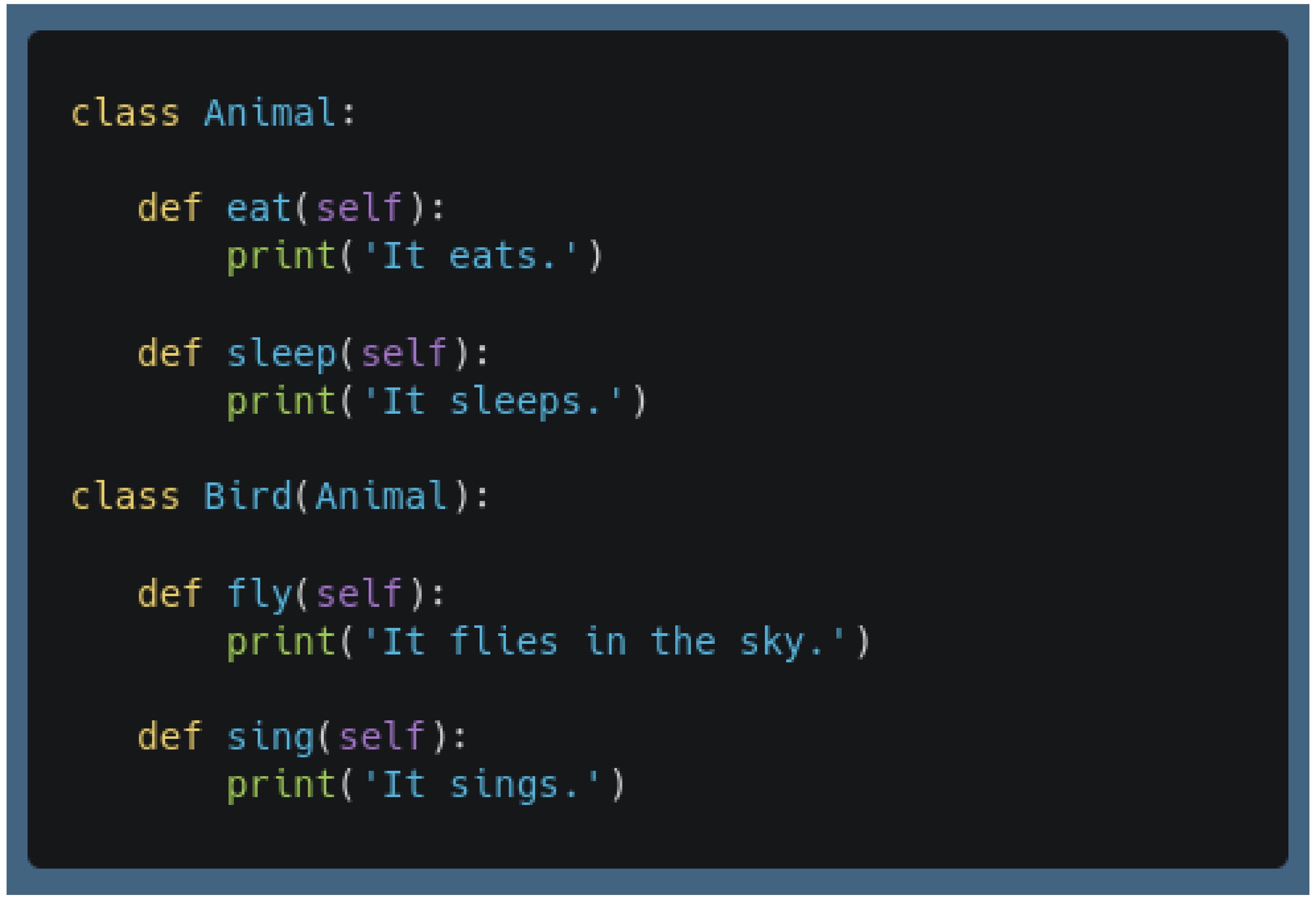
ADVANCED PYTHON

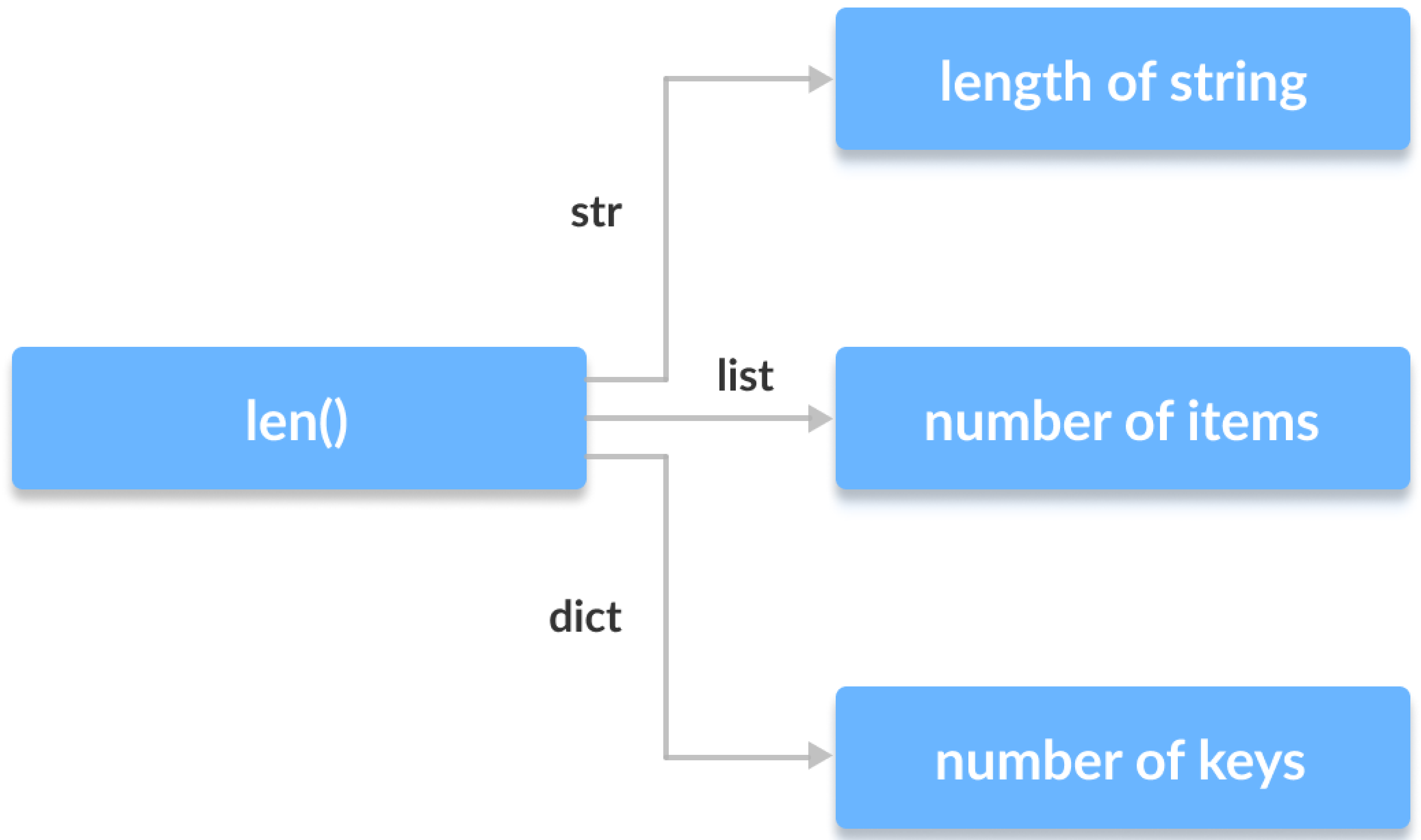
CONTENTS

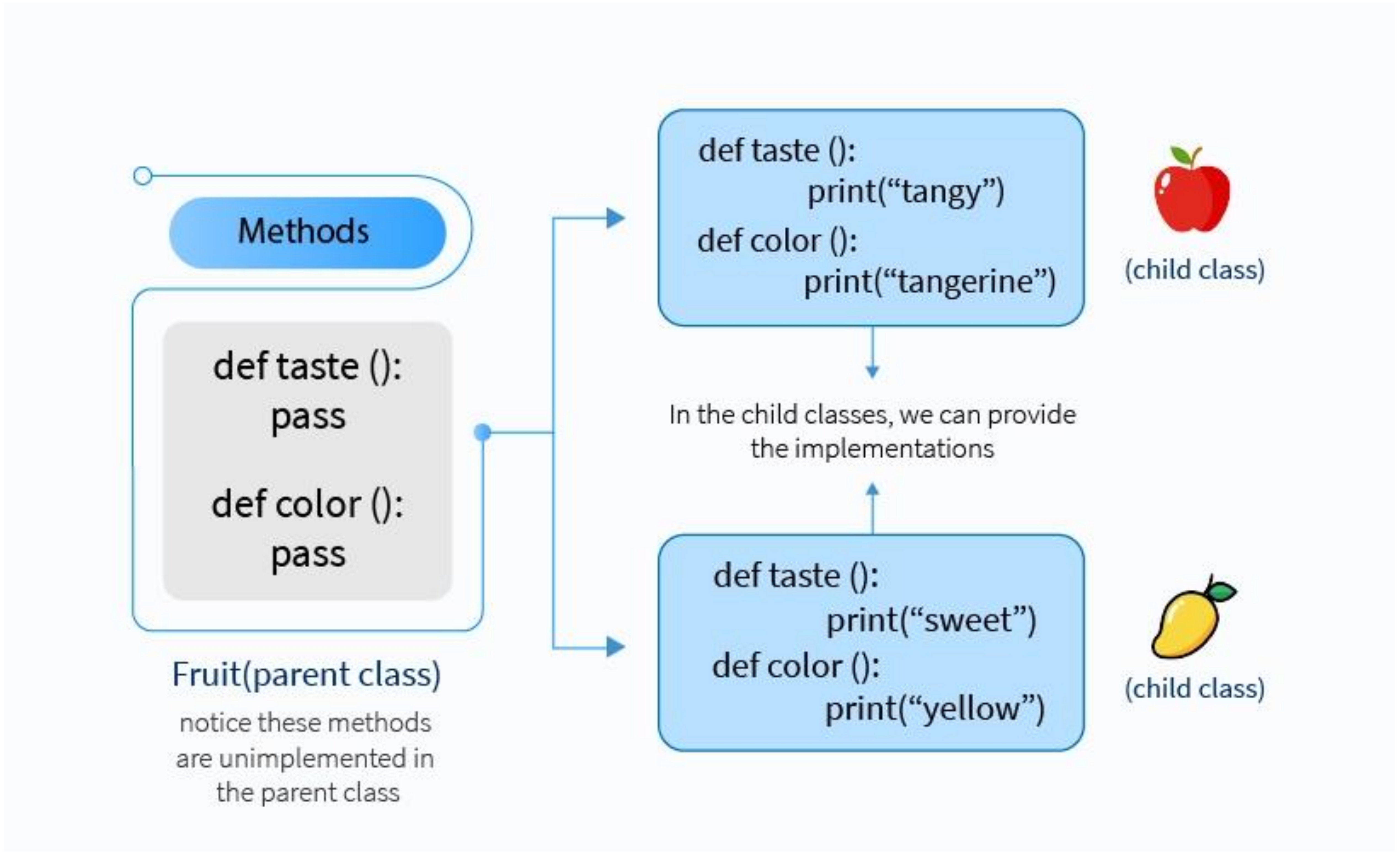
Tecnofor by SNGULAR

Advanced Object-Oriented Programming



```
class Animal:

   def eat(self):
       print('It eats.')

   def sleep(self):
       print('It sleeps.')

class Bird(Animal):

   def fly(self):
       print('It flies in the sky.')

   def sing(self):
       print('It sings.')
```


len()
str
list
dict
length of string
number of items
number of keys


Methods
Fruit(parent class)
notice these methods are unimplemented in the parent class
(child class)
In the child classes, we can provide the implementations
(child class)

```
def taste ():
    pass

def color ():
    pass
```

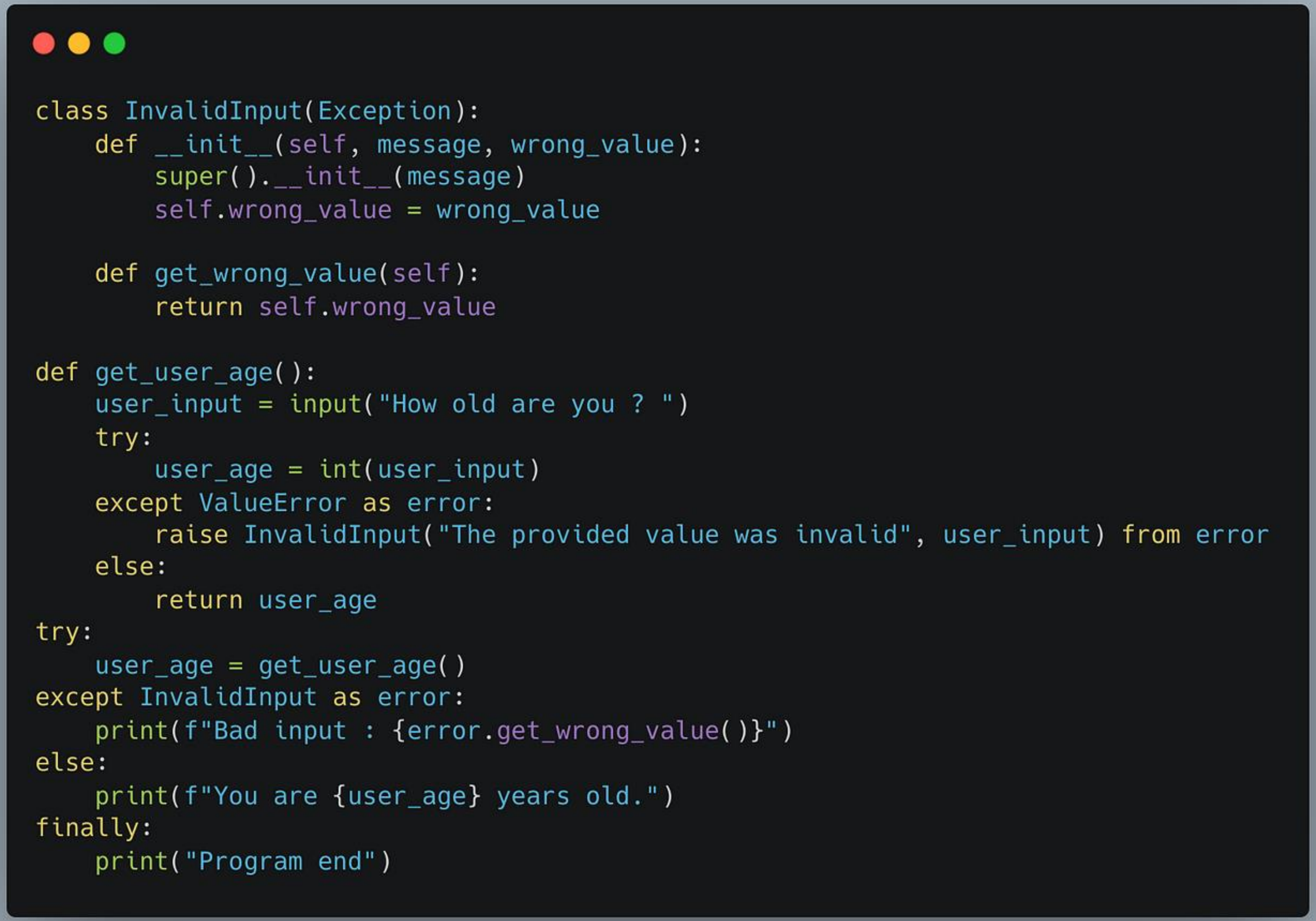
```
def taste ():
        print("tangy")
def color ():
        print("tangerine")
```

```
def taste ():
        print("sweet")
def color ():
        print("yellow")
```

TIME FOR PRACTICE

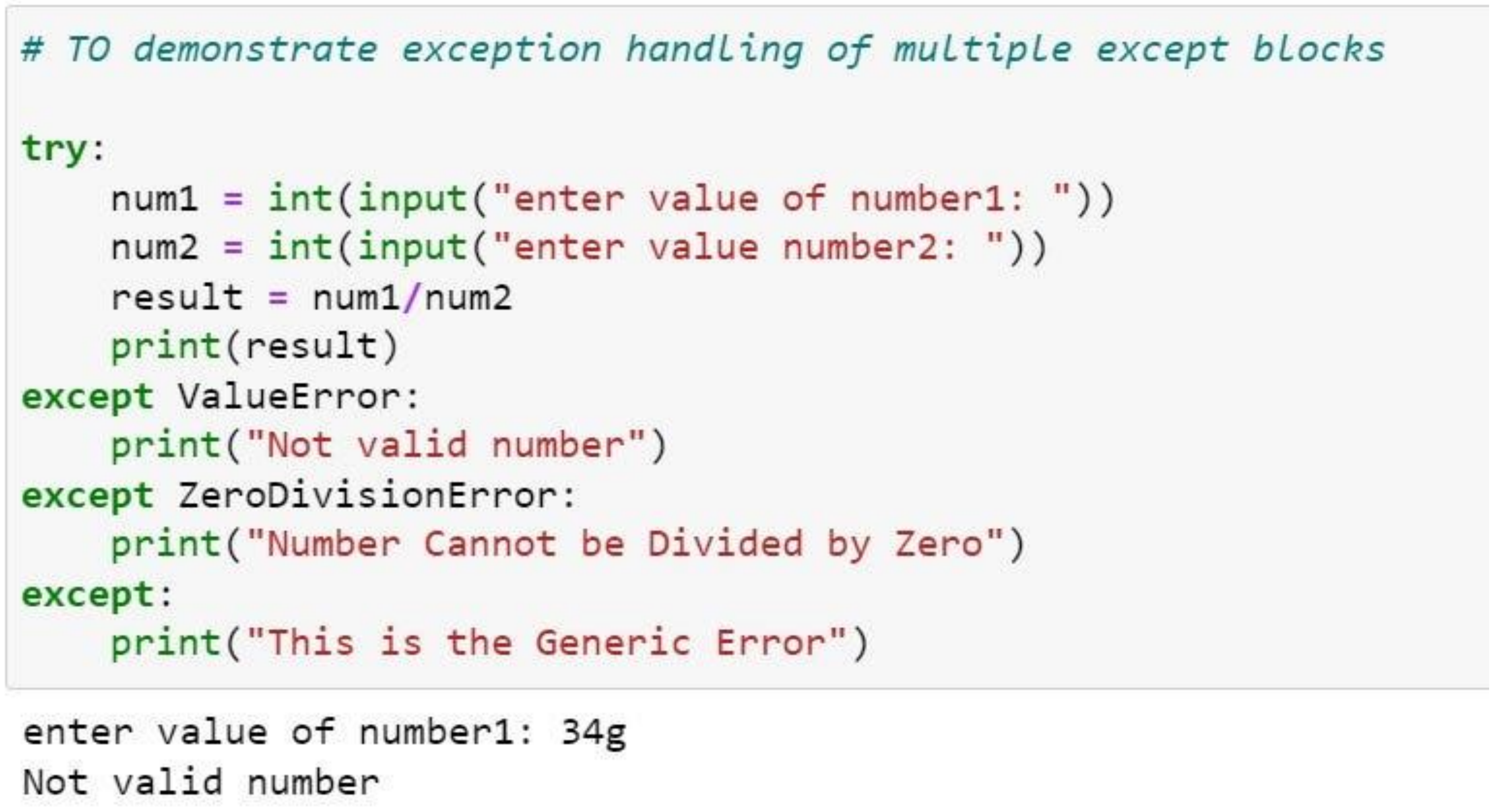
Tecnofor

Advanced Exception Handling



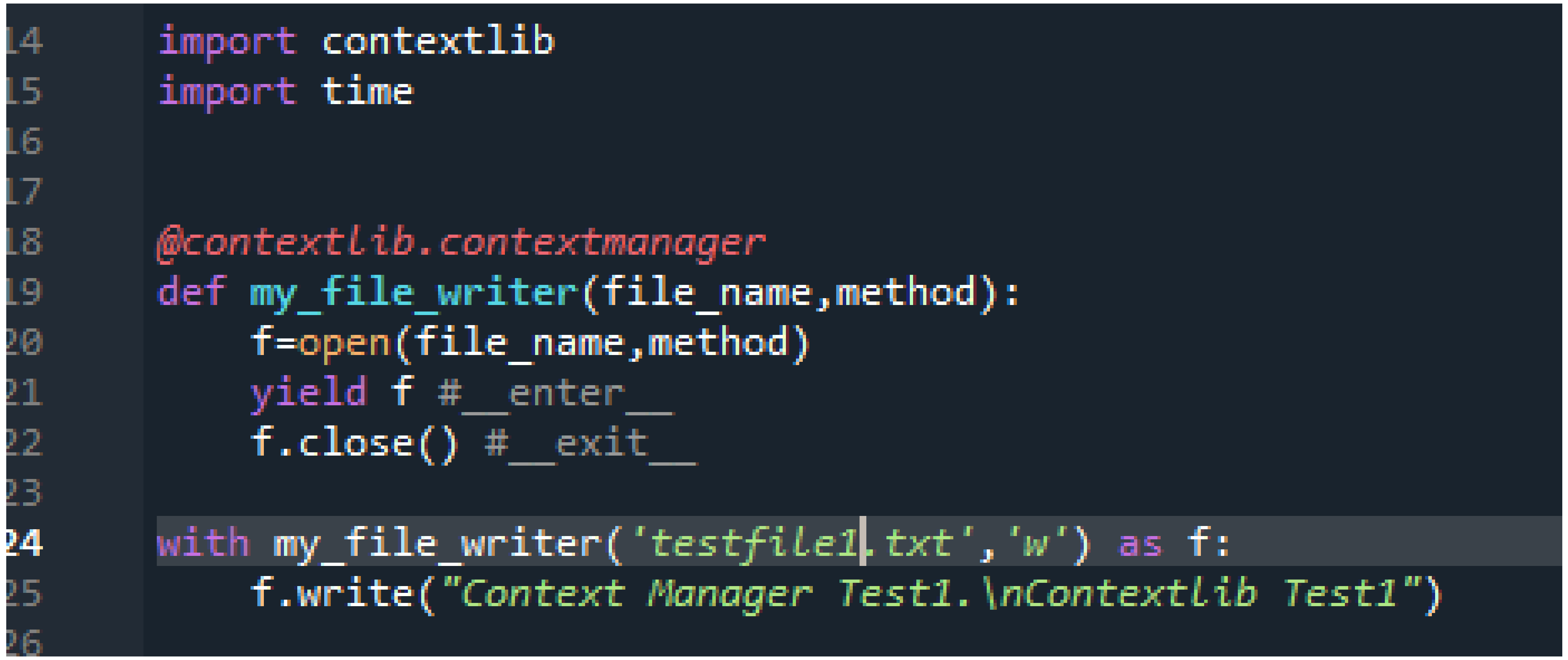
```python
class InvalidInput(Exception):
    def __init__(self, message, wrong_value):
        super().__init__(message)
        self.wrong_value = wrong_value

    def get_wrong_value(self):
        return self.wrong_value

def get_user_age():
    user_input = input("How old are you ? ")
    try:
        user_age = int(user_input)
    except ValueError as error:
        raise InvalidInput("The provided value was invalid", user_input) from error
    else:
        return user_age
try:
    user_age = get_user_age()
except InvalidInput as error:
    print(f"Bad input : {error.get_wrong_value()}")
else:
    print(f"You are {user_age} years old.")
finally:
    print("Program end")
```

Tecnofor
by SNGULAR



```
# TO demonstrate exception handling of multiple except blocks

try:
    num1 = int(input("enter value of number1: "))
    num2 = int(input("enter value number2: "))
    result = num1/num2
    print(result)
except ValueError:
    print("Not valid number")
except ZeroDivisionError:
    print("Number Cannot be Divided by Zero")
except:
    print("This is the Generic Error")
```

```
enter value of number1: 34g
Not valid number
```



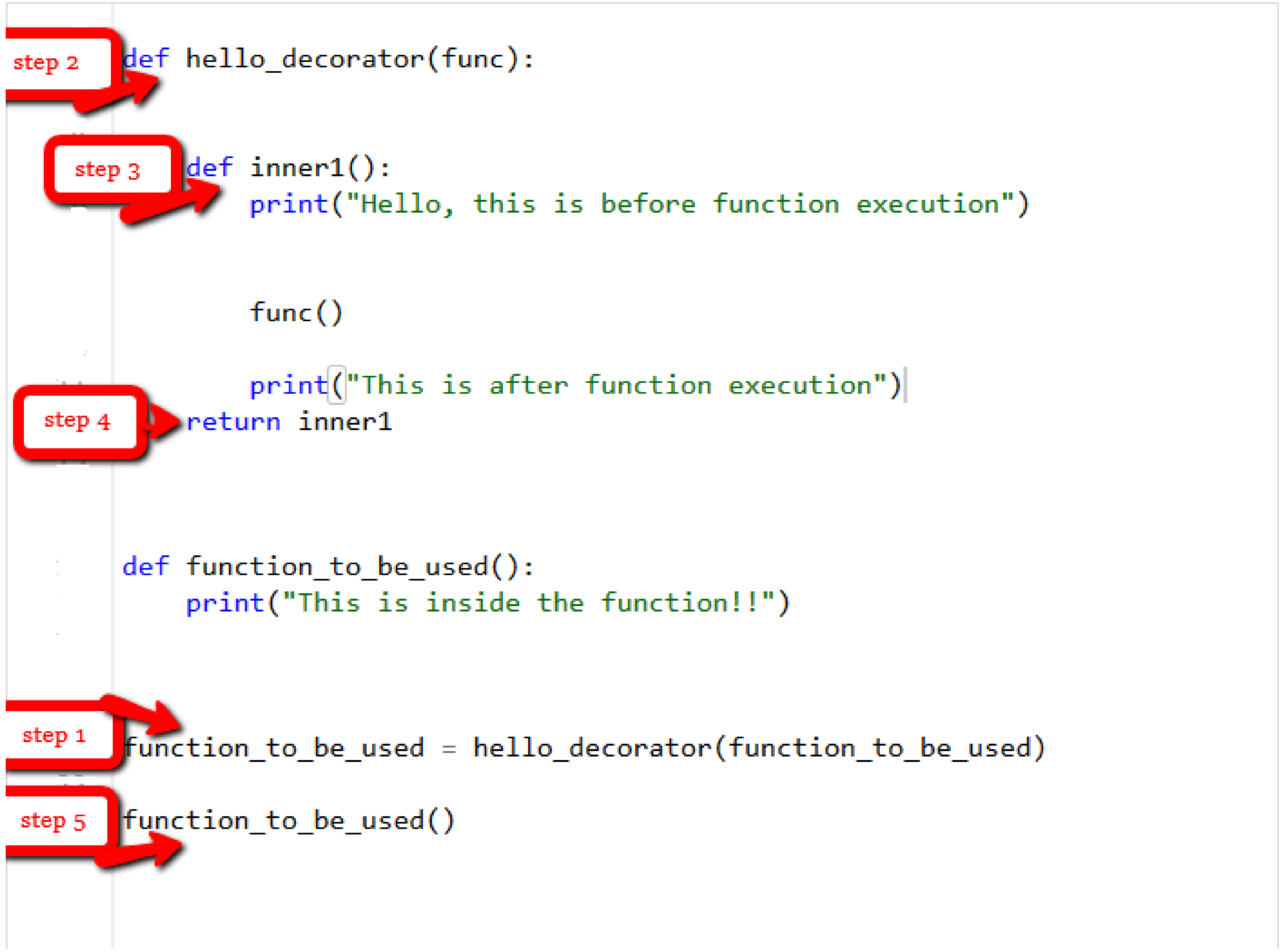
```python
import contextlib
import time


@contextlib.contextmanager
def my_file_writer(file_name,method):
    f=open(file_name,method)
    yield f #__enter__
    f.close() #__exit__

with my_file_writer('testfile1.txt','w') as f:
    f.write("Context Manager Test1.\nContextlib Test1")
```

Tecnofor by SNGULAR

Tecnofor
by SNGULAR

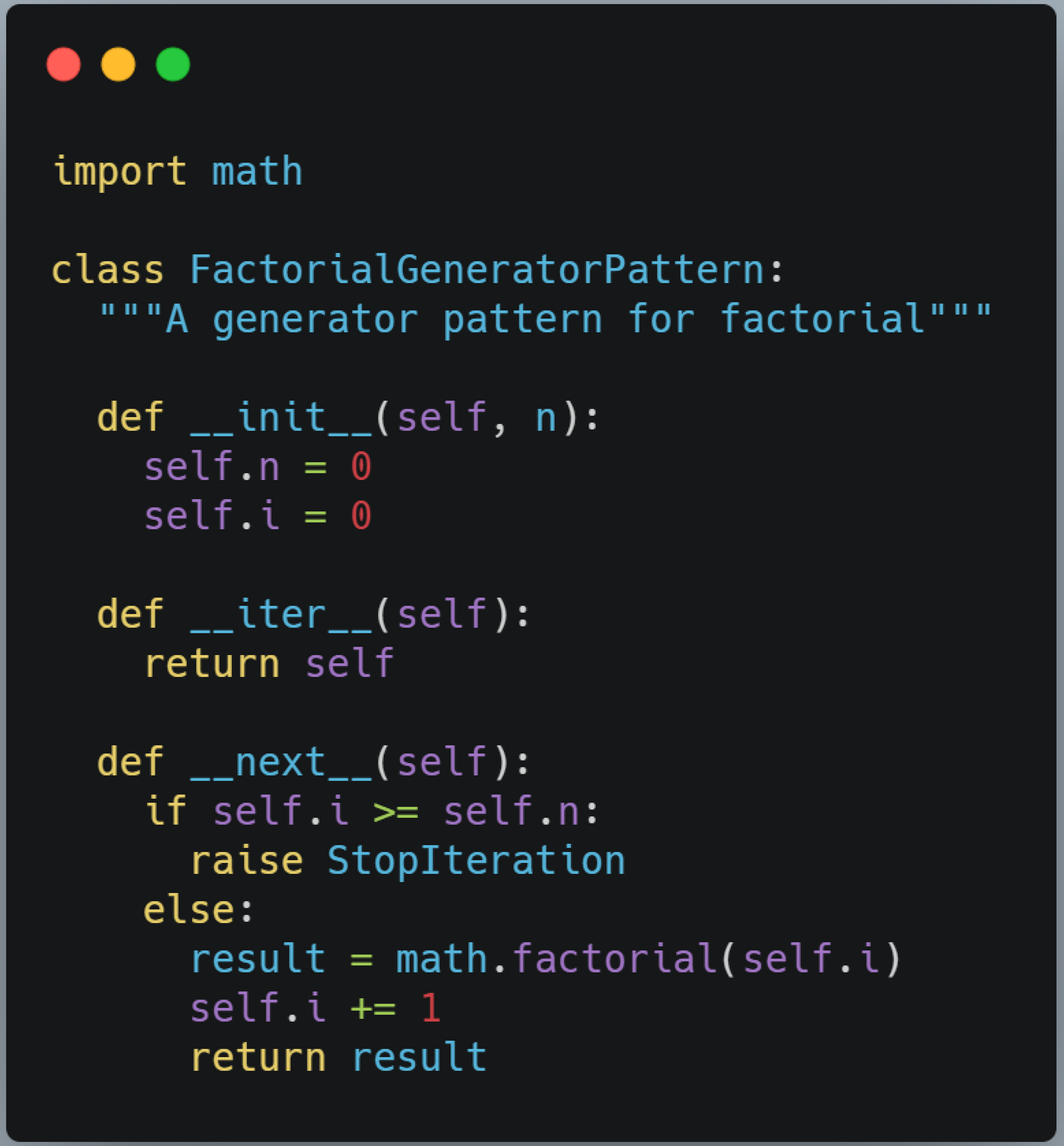
Tecnofor

Advanced Functional Programming

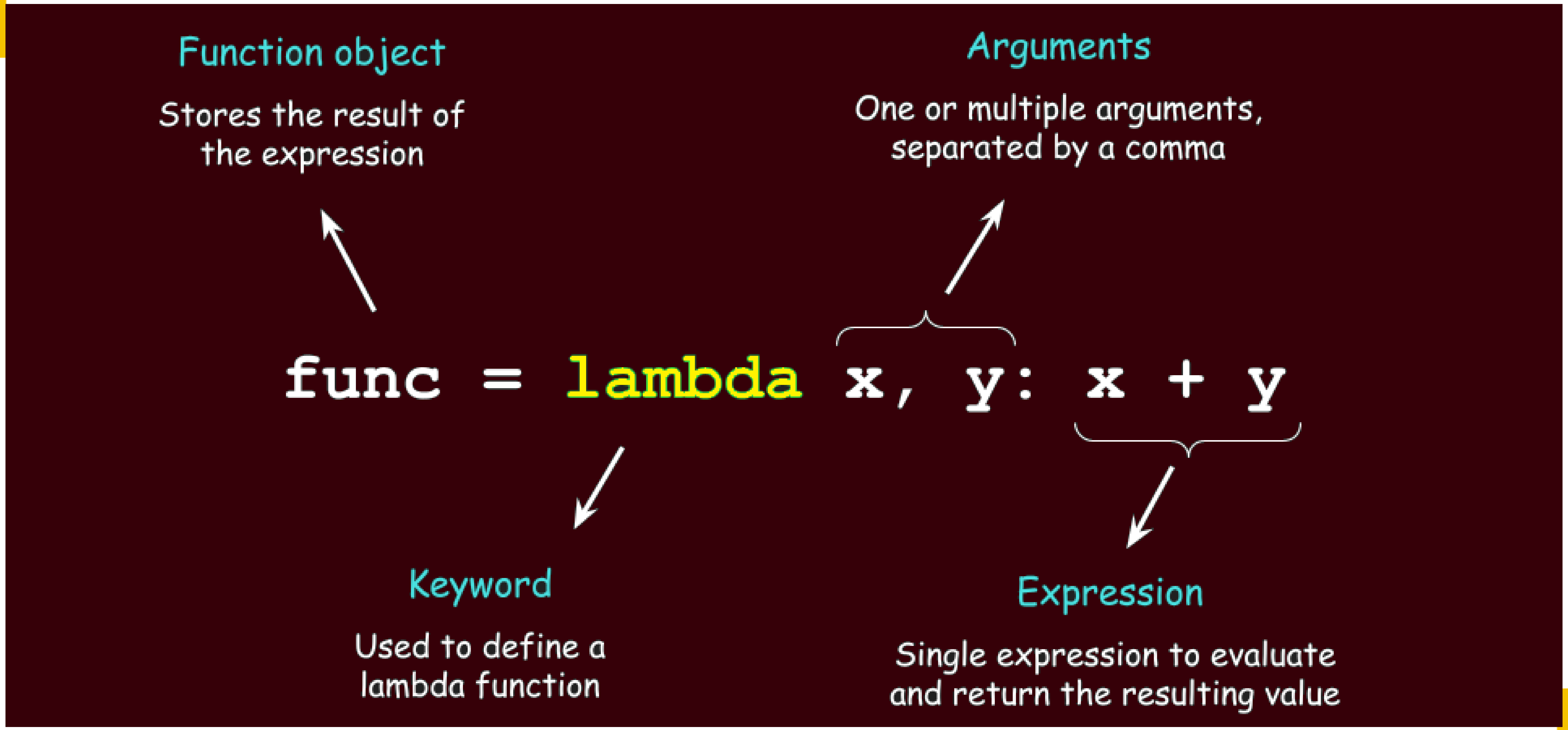


```
def hello_decorator(func):

    def inner1():
        print("Hello, this is before function execution")


        func()

        print("This is after function execution")
    return inner1



def function_to_be_used():
    print("This is inside the function!!")



function_to_be_used = hello_decorator(function_to_be_used)

function_to_be_used()
```

Tecnofor by SNGULAR



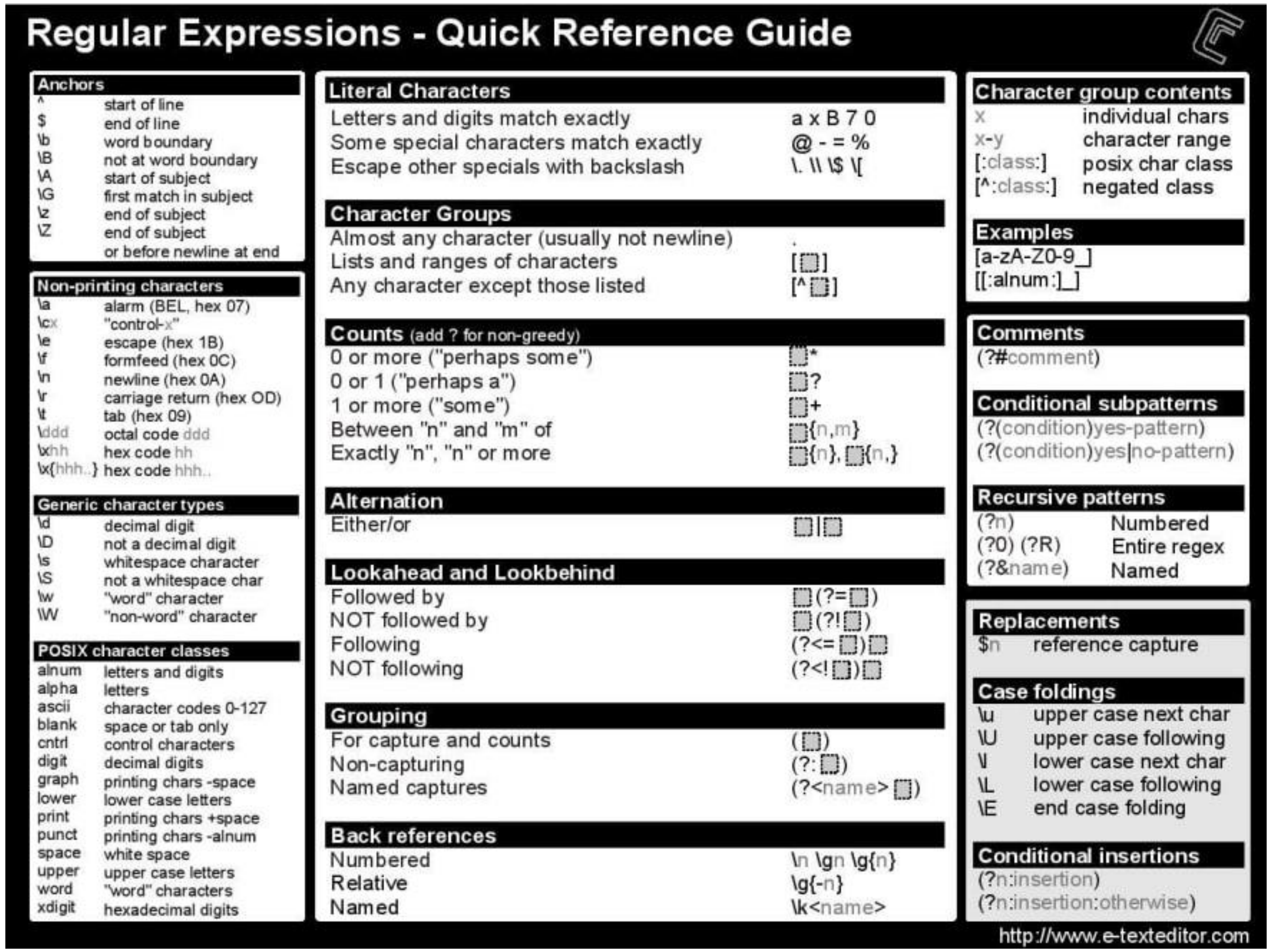
```
import math

class FactorialGeneratorPattern:
  """A generator pattern for factorial"""

  def __init__(self, n):
    self.n = 0
    self.i = 0

  def __iter__(self):
    return self

  def __next__(self):
    if self.i >= self.n:
      raise StopIteration
    else:
      result = math.factorial(self.i)
      self.i += 1
      return result
```


Function object
Stores the result of the expression
Arguments
One or multiple arguments, separated by a comma
func = lambda x, y: x + y
Keyword
Used to define a lambda function
Expression
Single expression to evaluate and return the resulting value

TIME FOR PRACTICE

Tecnofor

Regular Expressions



Regular Expressions - Quick Reference Guide

Anchors

^	start of line
$	end of line
\b	word boundary
\B	not at word boundary
\A	start of subject
\G	first match in subject
\z	end of subject
\Z	end of subject or before newline at end

Non-printing characters

\a	alarm (BEL, hex 07)
\cx	"control-x"
\e	escape (hex 1B)
\f	formfeed (hex 0C)
\n	newline (hex 0A)
\r	carriage return (hex OD)
\t	tab (hex 09)
\ddd	octal code ddd
\xhh	hex code hh
\x{hhh..}	hex code hhh..

Generic character types

\d	decimal digit
\D	not a decimal digit
\s	whitespace character
\S	not a whitespace char
\w	"word" character
\W	"non-word" character

POSIX character classes

alnum	letters and digits
alpha	letters
ascii	character codes 0-127
blank	space or tab only
cntrl	control characters
digit	decimal digits
graph	printing chars -space
lower	lower case letters
print	printing chars +space
punct	printing chars -alnum
space	white space
upper	upper case letters
word	"word" characters
xdigit	hexadecimal digits

Literal Characters

Letters and digits match exactly	a x B 7 0
Some special characters match exactly	@ - = %
Escape other specials with backslash	\. \\ \$ \[

Character Groups

Almost any character (usually not newline)	.
Lists and ranges of characters	[⬚]
Any character except those listed	[^⬚]

Counts (add ? for non-greedy)

0 or more ("perhaps some")	⬚*
0 or 1 ("perhaps a")	⬚?
1 or more ("some")	⬚+
Between "n" and "m" of	⬚{n,m}
Exactly "n", "n" or more	⬚{n}, ⬚{n,}

Alternation

Either/or	⬚|⬚

Lookahead and Lookbehind

Followed by	⬚(?=⬚)
NOT followed by	⬚(?!⬚)
Following	(?<=⬚)⬚
NOT following	(?<!⬚)⬚

Grouping

For capture and counts	(⬚)
Non-capturing	(?:⬚)
Named captures	(?<name>⬚)

Back references

Numbered	\n \gn \g{n}
Relative	\g{-n}
Named	\k<name>

Character group contents

x	individual chars
x-y	character range
[:class:]	posix char class
[^:class:]	negated class

Examples

[a-zA-Z0-9_]

[[:alnum:]_]

Comments

(?#comment)

Conditional subpatterns

(?(condition)yes-pattern)

(?(condition)yes|no-pattern)

Recursive patterns

(?n)	Numbered
(?0) (?R)	Entire regex
(?&name)	Named

Replacements

$n	reference capture

Case foldings

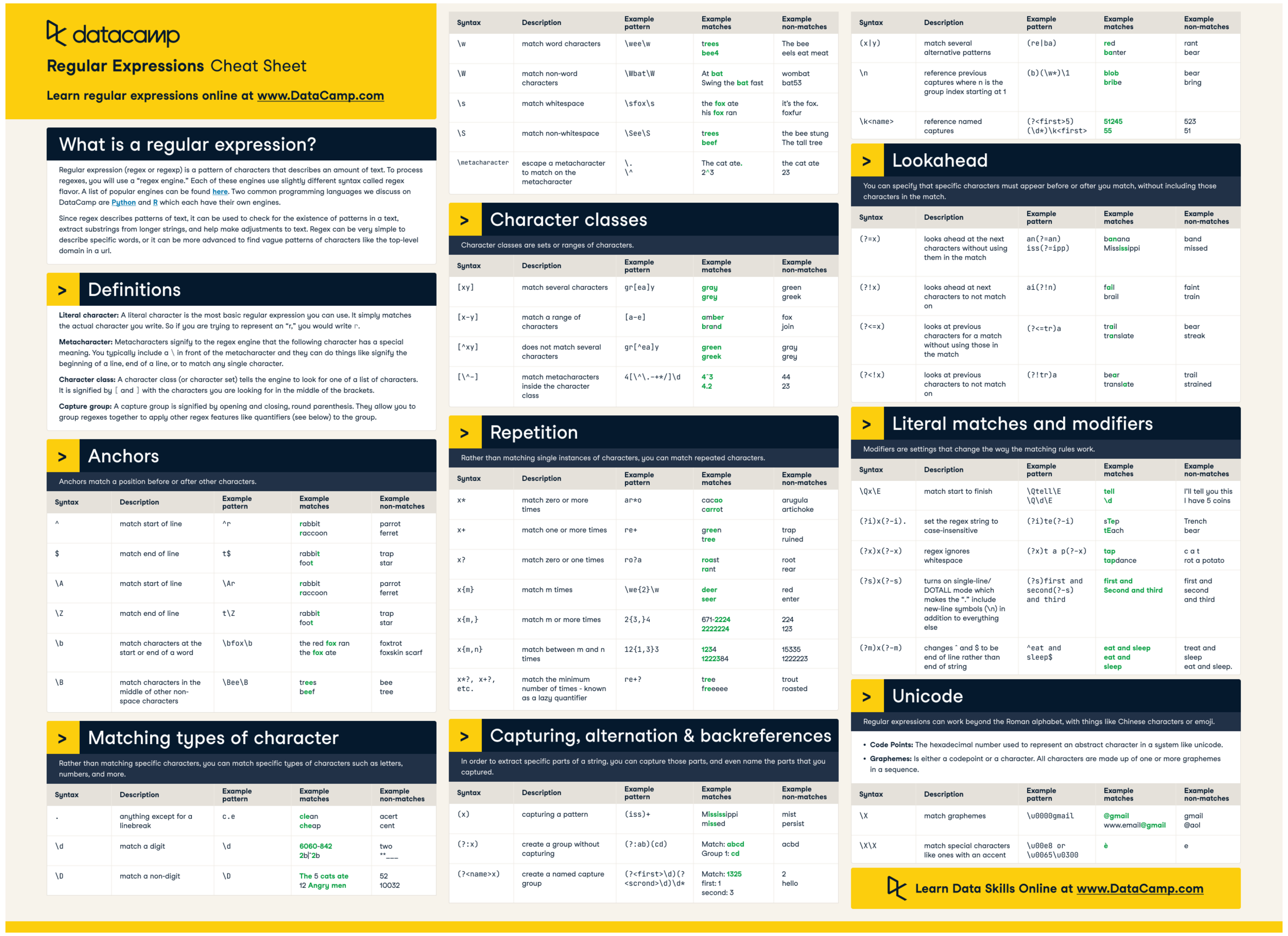
\u	upper case next char
\U	upper case following
\l	lower case next char
\L	lower case following
\E	end case folding

Conditional insertions

(?n:insertion)

(?n:insertion:otherwise)

http://www.e-texteditor.com



datacamp

Regular Expressions Cheat Sheet

Learn regular expressions online at www.DataCamp.com

What is a regular expression?

Regular expression (regex or regexp) is a pattern of characters that describes an amount of text. To process regexes, you will use a "regex engine." Each of these engines use slightly different syntax called regex flavor. A list of popular engines can be found here. Two common programming languages we discuss on DataCamp are Python and R which each have their own engines.

Since regex describes patterns of text, it can be used to check for the existence of patterns in a text, extract substrings from longer strings, and help make adjustments to text. Regex can be very simple to describe specific words, or it can be more advanced to find vague patterns of characters like the top-level domain in a url.

> Definitions

Literal character: A literal character is the most basic regular expression you can use. It simply matches the actual character you write. So if you are trying to represent an "r," you would write `r`.

Metacharacter: Metacharacters signify to the regex engine that the following character has a special meaning. You typically include a `\` in front of the metacharacter and they can do things like signify the beginning of a line, end of a line, or to match any single character.

Character class: A character class (or character set) tells the engine to look for one of a list of characters. It is signified by `[` and `]` with the characters you are looking for in the middle of the brackets.

Capture group: A capture group is signified by opening and closing, round parenthesis. They allow you to group regexes together to apply other regex features like quantifiers (see below) to the group.

> Anchors

Anchors match a position before or after other characters.

Syntax	Description	Example pattern	Example matches	Example non-matches
^	match start of line	^r	rabbit raccoon	parrot ferret
$	match end of line	t$	rabbit foot	trap star
\A	match start of line	\Ar	rabbit raccoon	parrot ferret
\Z	match end of line	t\Z	rabbit foot	trap star
\b	match characters at the start or end of a word	\bfox\b	the red fox ran the fox ate	foxtrot foxskin scarf
\B	match characters in the middle of other non-space characters	\Bee\B	trees beef	bee tree

> Matching types of character

Rather than matching specific characters, you can match specific types of characters such as letters, numbers, and more.

Syntax	Description	Example pattern	Example matches	Example non-matches
.	anything except for a linebreak	c.e	clean cheap	acert cent
\d	match a digit	\d	6060-842 2b|^2b	two **___
\D	match a non-digit	\D	The 5 cats ate 12 Angry men	52 10032
\w	match word characters	\wee\w	trees bee4	The bee eels eat meat
\W	match non-word characters	\Wbat\W	At bat Swing the bat fast	wombat bat53
\s	match whitespace	\sfox\s	the fox ate his fox ran	it's the fox. foxfur
\S	match non-whitespace	\See\S	trees beef	the bee stung The tall tree
\metacharacter	escape a metacharacter to match on the metacharacter	\. \^	The cat ate. 2^3	the cat ate 23

> Character classes

Character classes are sets or ranges of characters.

Syntax	Description	Example pattern	Example matches	Example non-matches
[xy]	match several characters	gr[ea]y	gray grey	green greek
[x-y]	match a range of characters	[a-e]	amber brand	fox join
[^xy]	does not match several characters	gr[^ea]y	green greek	gray grey
[\^-]	match metacharacters inside the character class	4[\^\.-+*/]\d	4^3 4.2	44 23

> Repetition

Rather than matching single instances of characters, you can match repeated characters.

Syntax	Description	Example pattern	Example matches	Example non-matches
x*	match zero or more times	ar*o	cacao carrot	arugula artichoke
x+	match one or more times	re+	green tree	trap ruined
x?	match zero or one times	ro?a	roast rant	root rear
x{m}	match m times	\we{2}\w	deer seer	red enter
x{m,}	match m or more times	2{3,}4	671-2224 2222224	224 123
x{m,n}	match between m and n times	12{1,3}3	1234 1222384	15335 1222223
x*?, x+?, etc.	match the minimum number of times - known as a lazy quantifier	re+?	tree freeeee	trout roasted

> Capturing, alternation & backreferences

In order to extract specific parts of a string, you can capture those parts, and even name the parts that you captured.

Syntax	Description	Example pattern	Example matches	Example non-matches
(x)	capturing a pattern	(iss)+	Mississippi missed	mist persist
(?:x)	create a group without capturing	(?:ab)(cd)	Match: abcd Group 1: cd	acbd
(?<name>x)	create a named capture group	(?<first>\d)(?<scrond>\d)\d*	Match: 1325 first: 1 second: 3	2 hello
(x|y)	match several alternative patterns	(re|ba)	red banter	rant bear
\n	reference previous captures where n is the group index starting at 1	(b)(\w*)\1	blob bribe	bear bring
\k<name>	reference named captures	(?<first>5)(\d*)\k<first>	51245 55	523 51

> Lookahead

You can specify that specific characters must appear before or after you match, without including those characters in the match.

Syntax	Description	Example pattern	Example matches	Example non-matches
(?=x)	looks ahead at the next characters without using them in the match	an(?=an) iss(?=ipp)	banana Mississippi	band missed
(?!x)	looks ahead at next characters to not match on	ai(?!n)	fail brail	faint train
(?<=x)	looks at previous characters for a match without using those in the match	(?<=tr)a	trail translate	bear streak
(?<!x)	looks at previous characters to not match on	(?!tr)a	bear translate	trail strained

> Literal matches and modifiers

Modifiers are settings that change the way the matching rules work.

Syntax	Description	Example pattern	Example matches	Example non-matches
\Qx\E	match start to finish	\Qtell\E \Q\d\E	tell \d	I'll tell you this I have 5 coins
(?i)x(?-i).	set the regex string to case-insensitive	(?i)te(?-i)	sTep tEach	Trench bear
(?x)x(?-x)	regex ignores whitespace	(?x)t a p(?-x)	tap tapdance	c a t rot a potato
(?s)x(?-s)	turns on single-line/DOTALL mode which makes the "." include new-line symbols (\n) in addition to everything else	(?s)first and second(?-s) and third	first and Second and third	first and second and third
(?m)x(?-m)	changes ^ and $ to be end of line rather than end of string	^eat and sleep$	eat and sleep eat and sleep	treat and sleep eat and sleep.

> Unicode

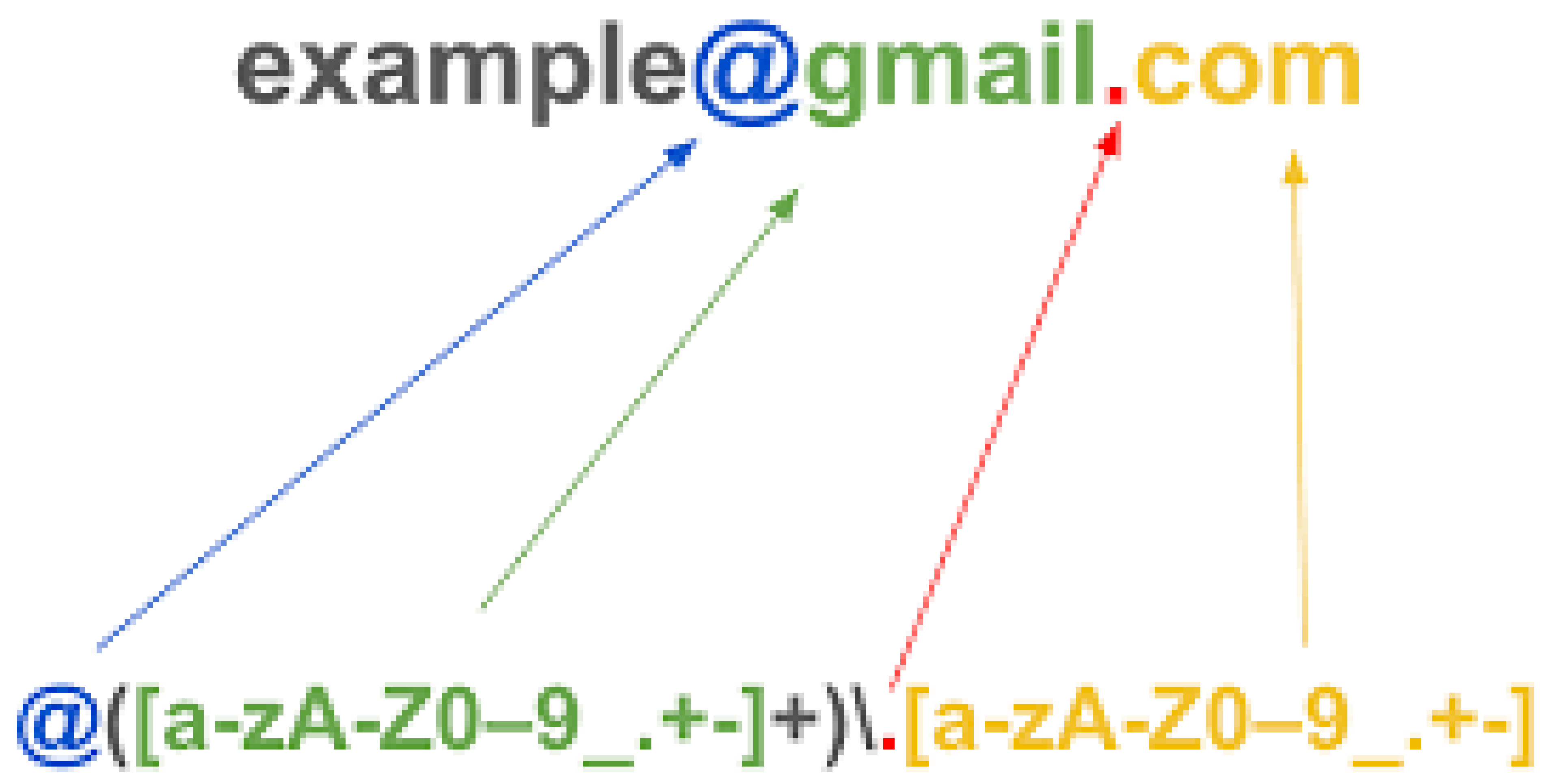
Regular expressions can work beyond the Roman alphabet, with things like Chinese characters or emoji.

- **Code Points:** The hexadecimal number used to represent an abstract character in a system like unicode.
- **Graphemes:** Is either a codepoint or a character. All characters are made up of one or more graphemes in a sequence.

Syntax	Description	Example pattern	Example matches	Example non-matches
\X	match graphemes	\u0000gmail	@gmail www.email@gmail	gmail @aol
\X\X	match special characters like ones with an accent	\u00e8 or \u0065\u0300	è	e

Learn Data Skills Online at www.DataCamp.com

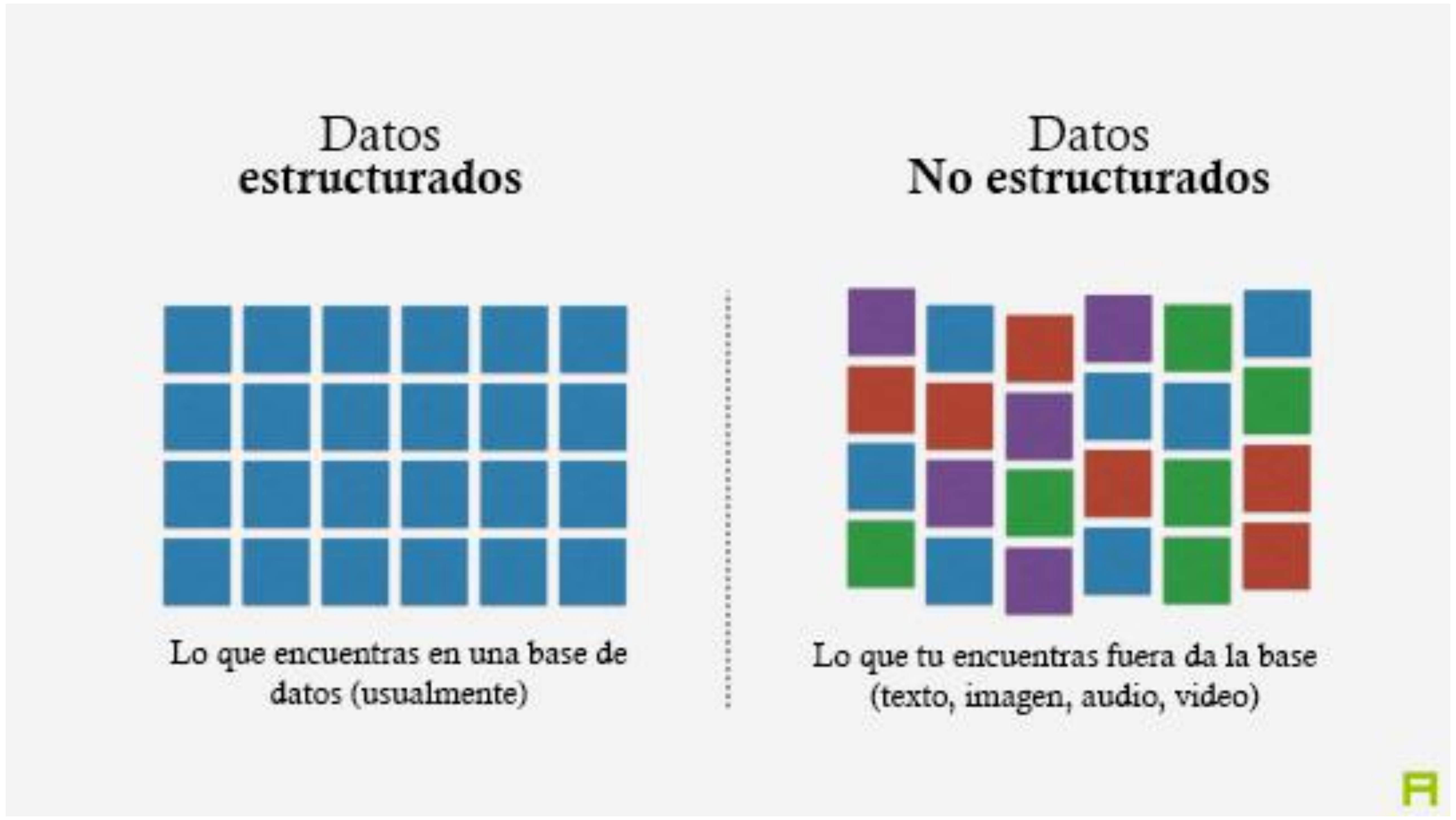
Tecnofor by SNGULAR


example@gmail.com
@([a-zA-Z0–9_.+-]+)\.[a-zA-Z0–9_.+-]

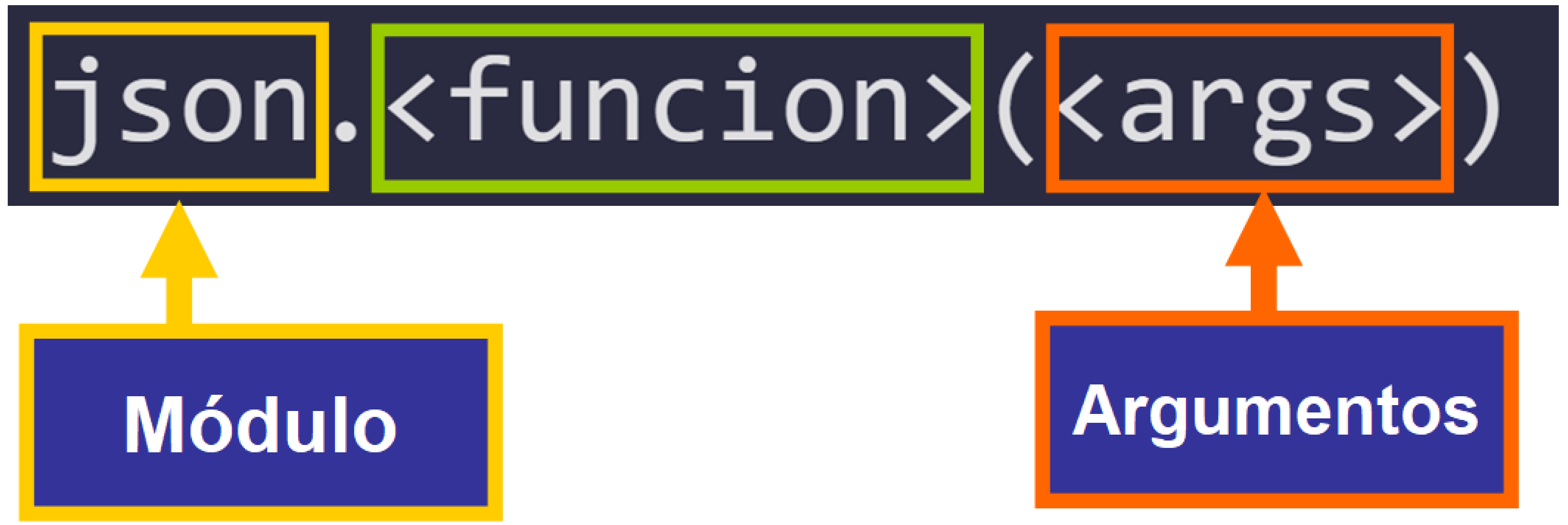
Tecnofor by SNGULAR

Tecnofor

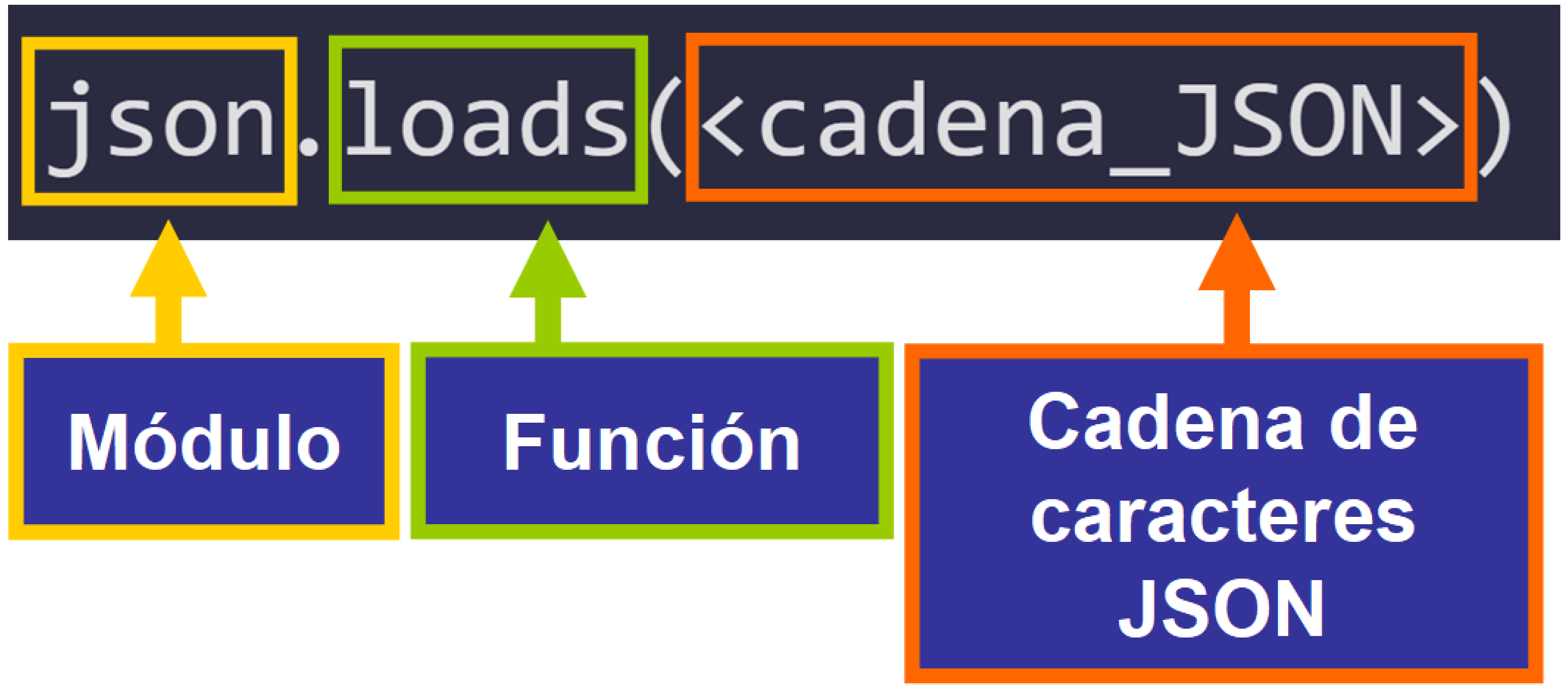
Advanced File Handling


Datos
estructurados
Lo que encuentras en una base de datos (usualmente)
Datos
No estructurados
Lo que tu encuentras fuera da la base (texto, imagen, audio, video)

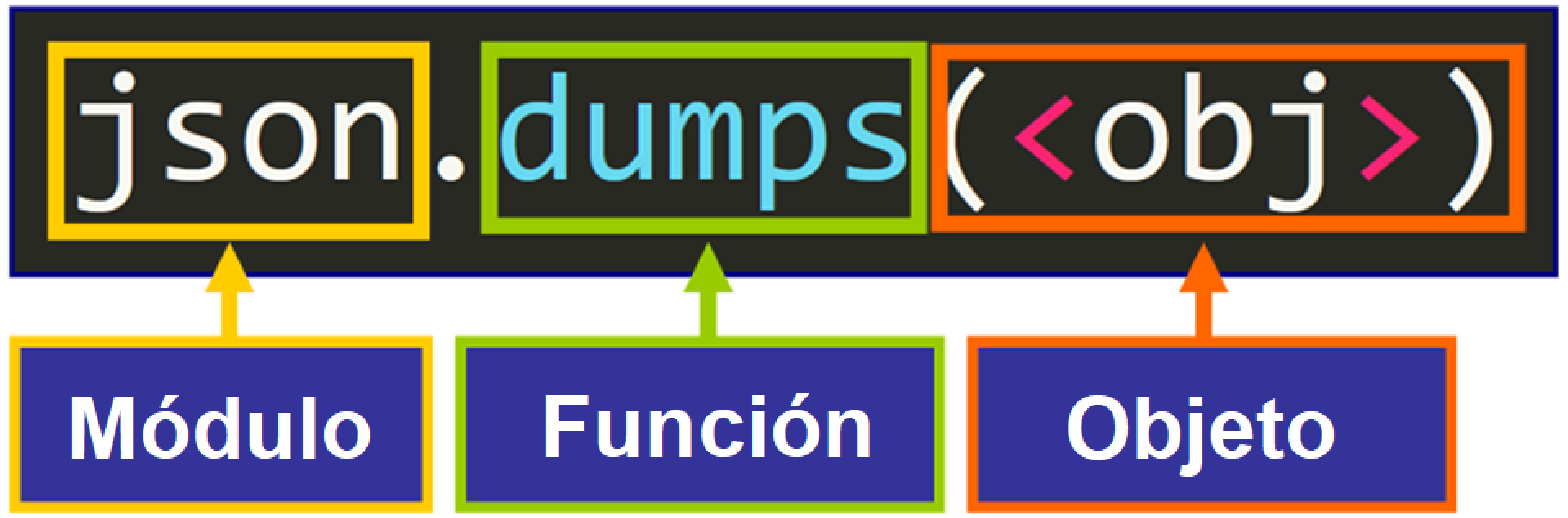
```
{
  “crust”: “original”,
  “toppings”: [“cheese”, “pepperoni”, “garlic”],
  “status”: “cooking”
}
```



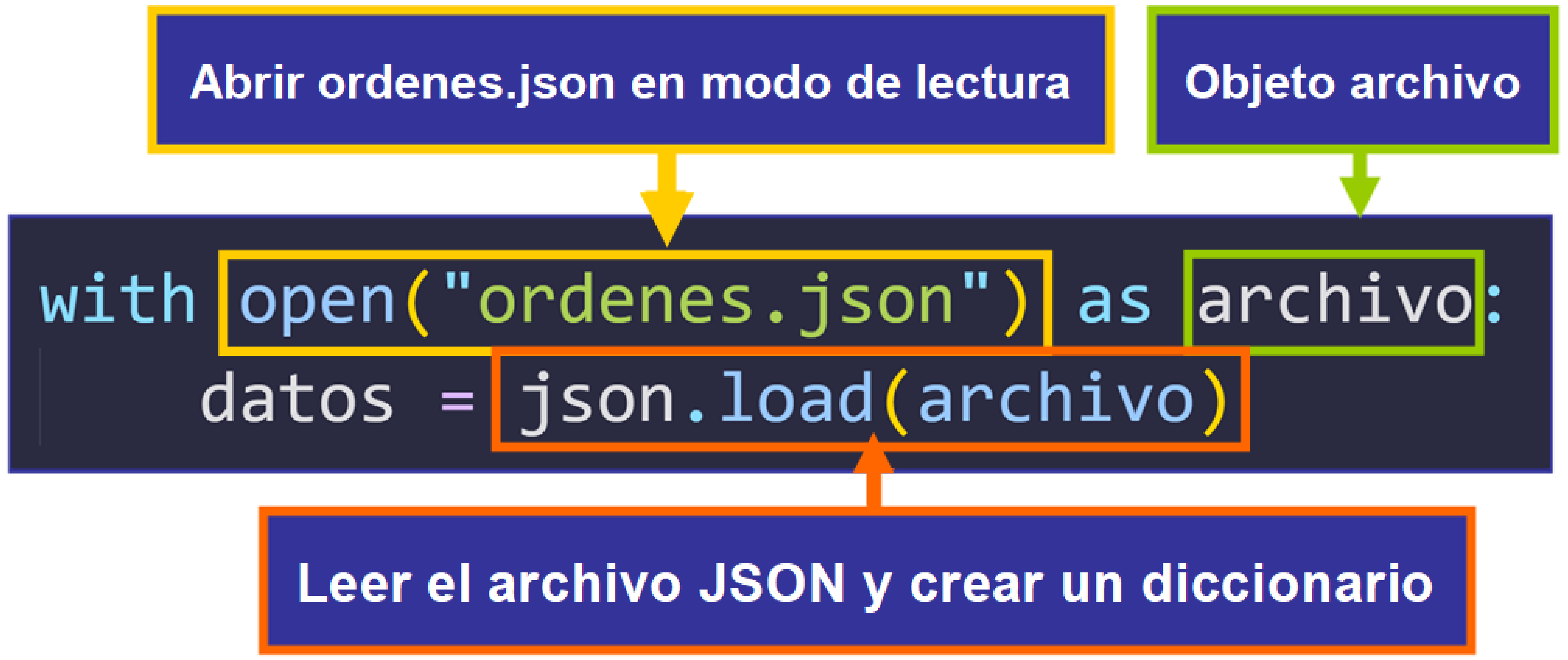
```
json.<funcion>(<args>)
```



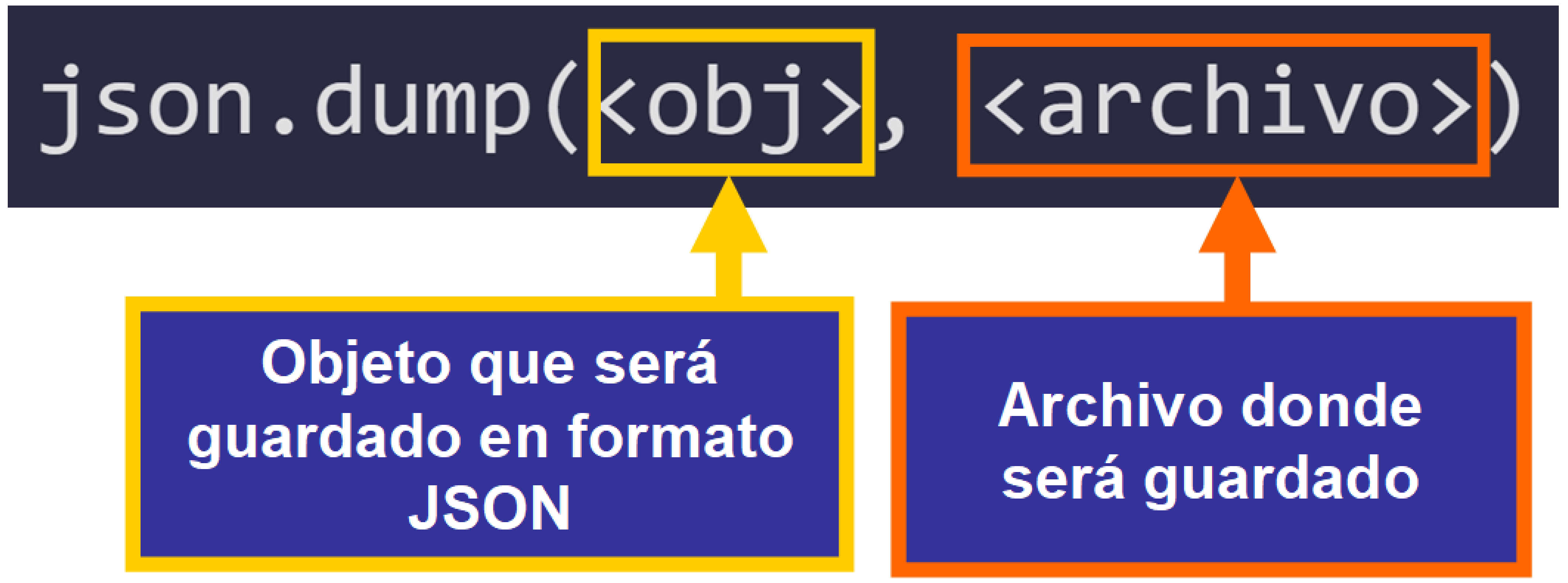
```
json.loads(<cadena_JSON>)
```



```
json.dumps(<obj>)
```


Abrir ordenes.json en modo de lectura
Objeto archivo
Leer el archivo JSON y crear un diccionario

```python
with open("ordenes.json") as archivo:
    datos = json.load(archivo)
```



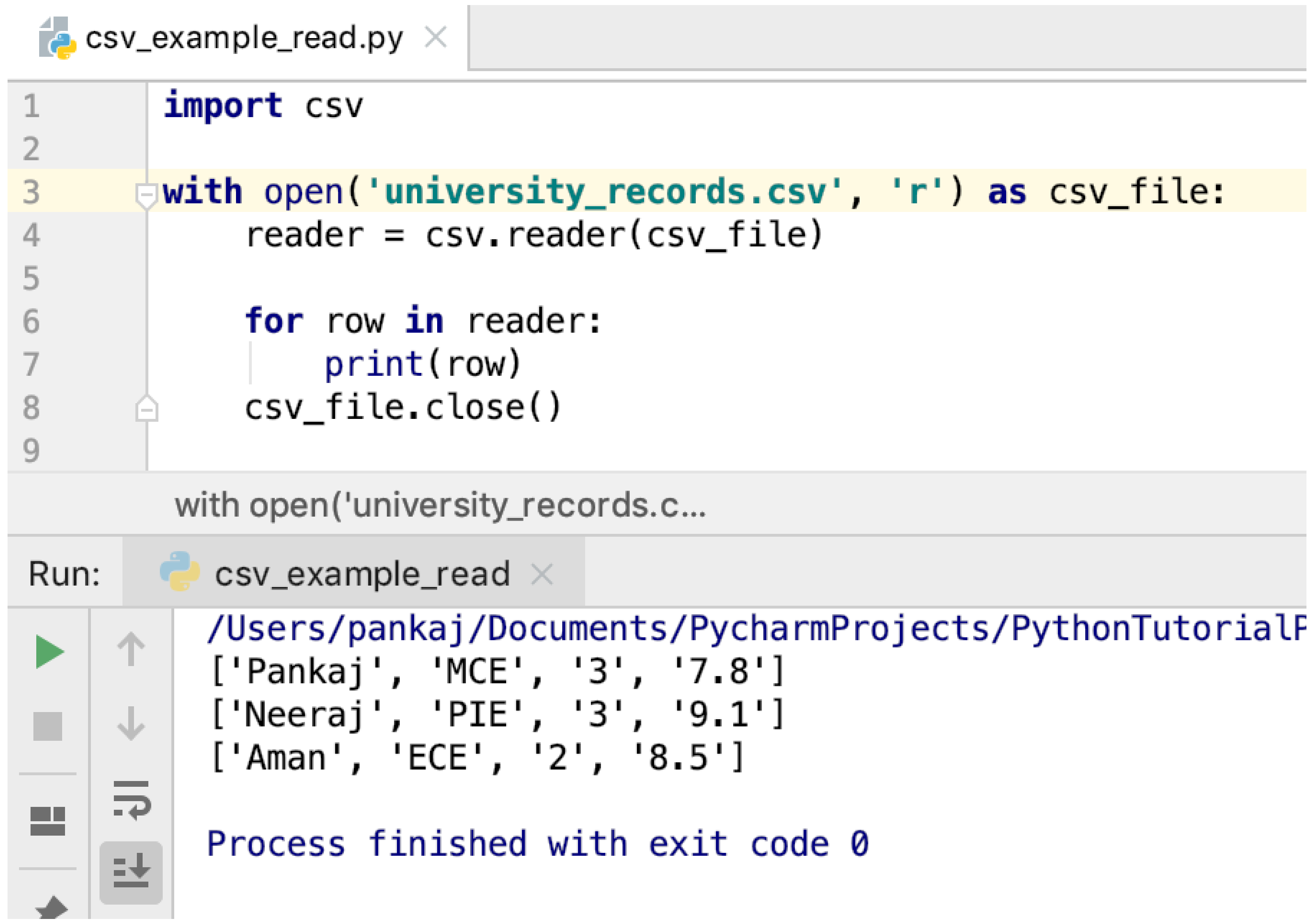
```
json.dump(<obj>, <archivo>)
```

Objeto que será guardado en formato JSON
Archivo donde será guardado

	load()	loads()
Propósito	Crear un objeto de Python a partir de un archivo JSON	Crear un objeto de Python a partir de una cadena de caracteres
Argumento	Archivo JSON	Cadena de caracteres
Valor retornado	Objeto de Python	Objeto de Python

	dump()	dumps()
Propósito	Escribir un objeto en formato JSON a un archivo	Obtener una cadena de caracteres JSON a partir de un objeto
Argumento	Objeto + Archivo	Objeto
Valor retornado	None	Cadena de caracteres

TIME FOR PRACTICE


csv_example_read.py

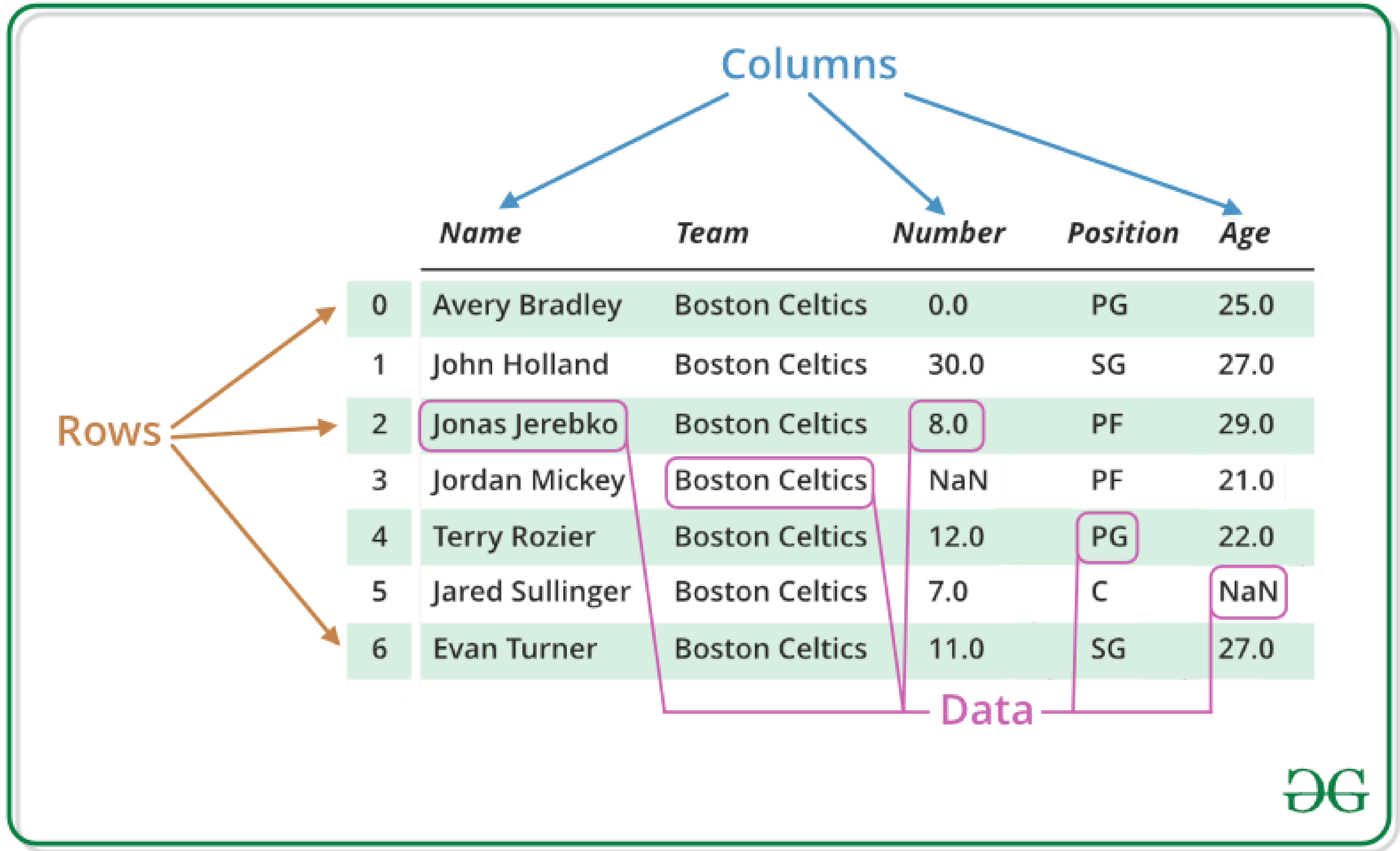
```python
import csv

with open('university_records.csv', 'r') as csv_file:
    reader = csv.reader(csv_file)

    for row in reader:
        print(row)
    csv_file.close()

```

with open('university_records.c...

Run: csv_example_read

```
/Users/pankaj/Documents/PycharmProjects/PythonTutorialP
['Pankaj', 'MCE', '3', '7.8']
['Neeraj', 'PIE', '3', '9.1']
['Aman', 'ECE', '2', '8.5']

Process finished with exit code 0
```

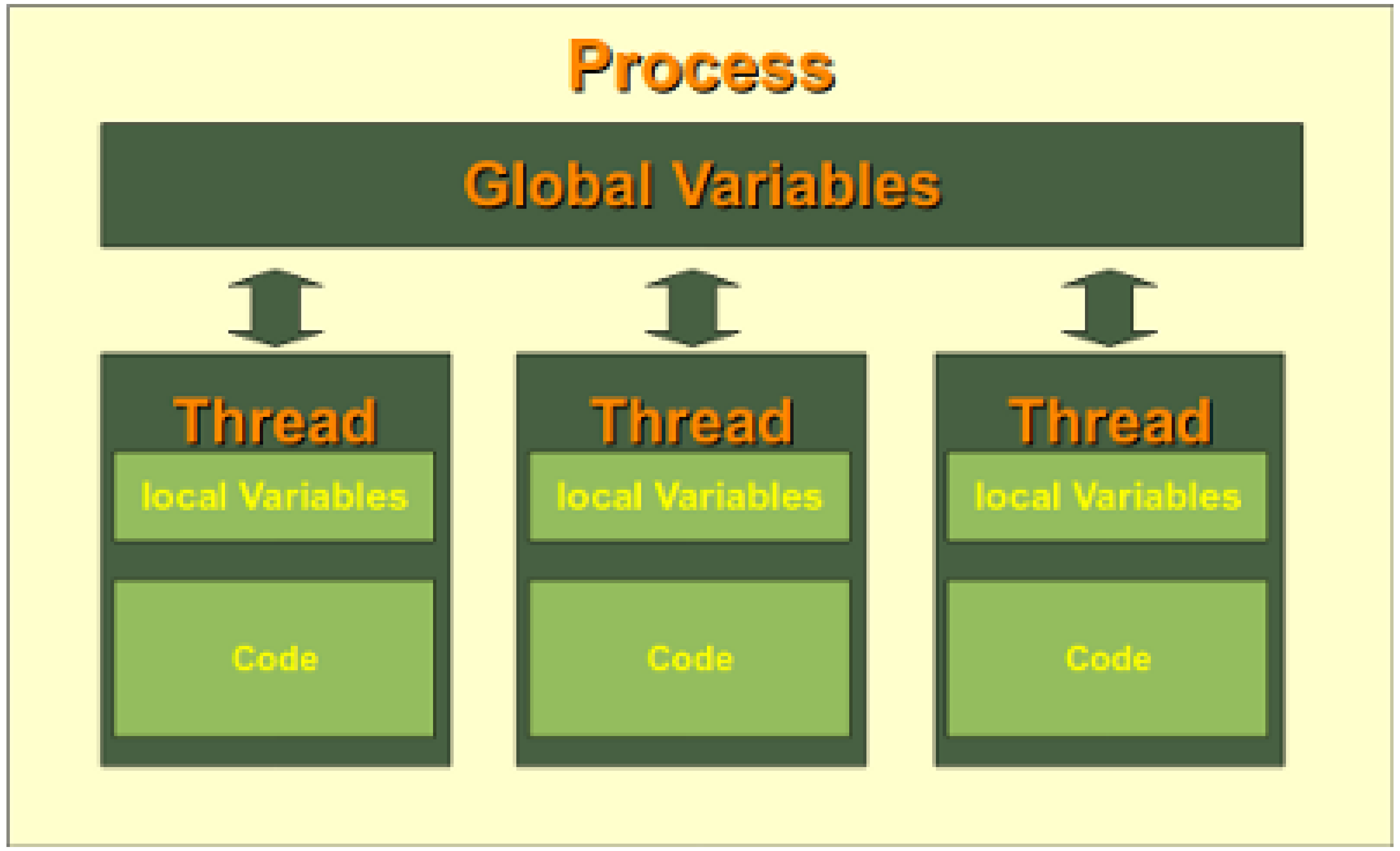
Tecnofor by SNGULAR



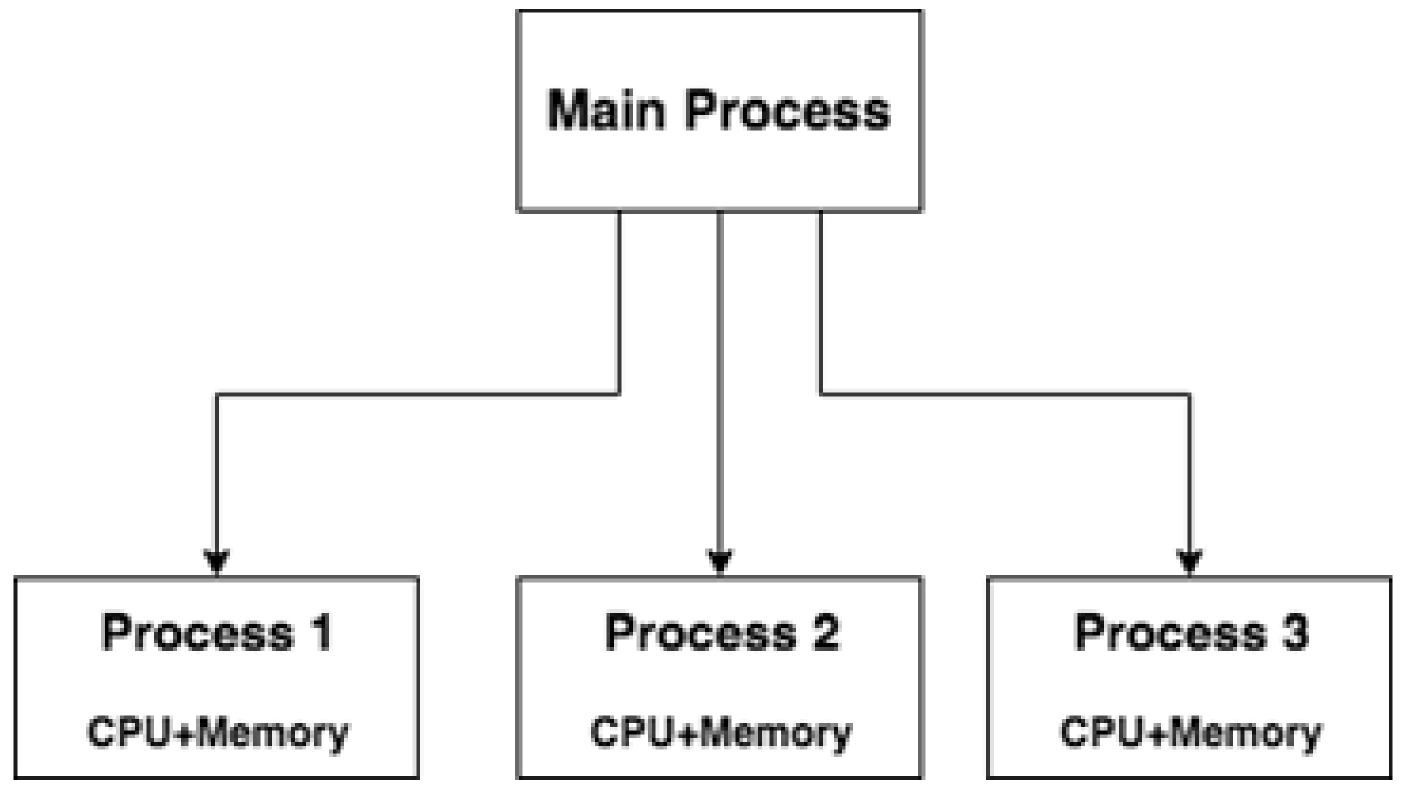
	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

TIME FOR PRACTICE

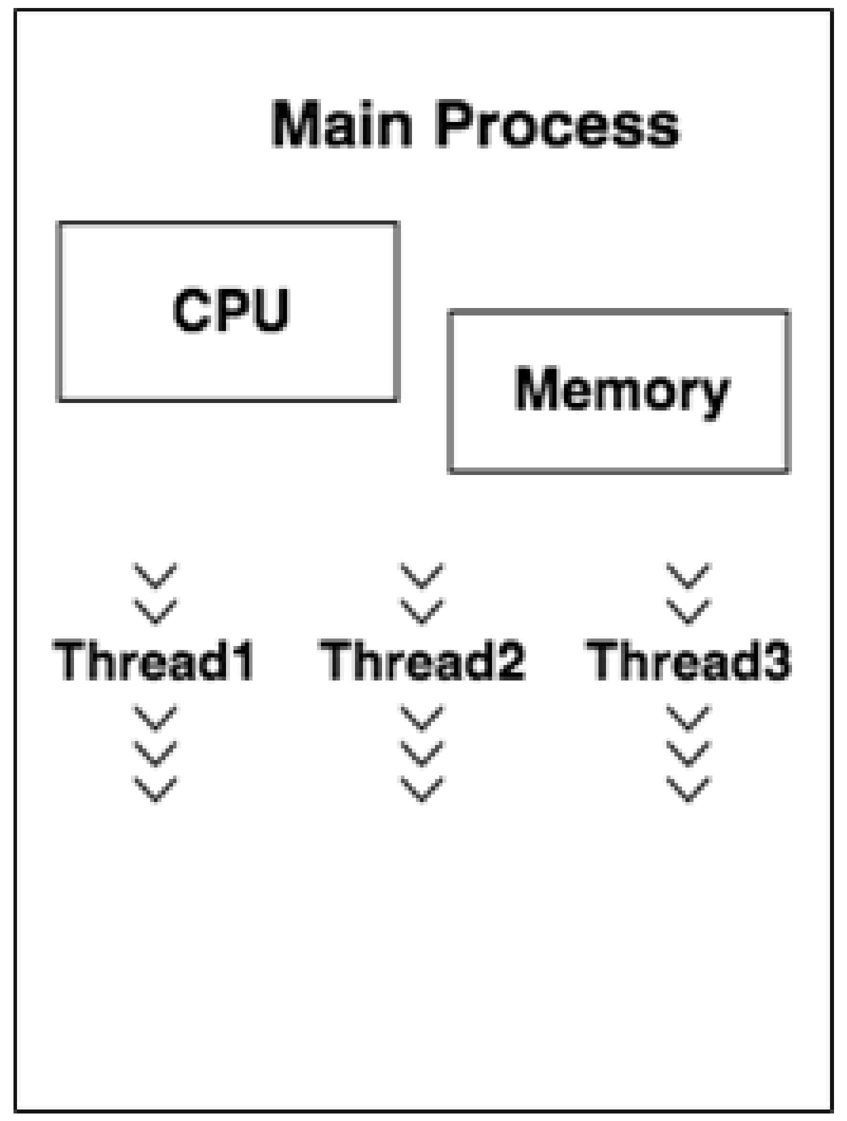
Concurrency and Parallelism

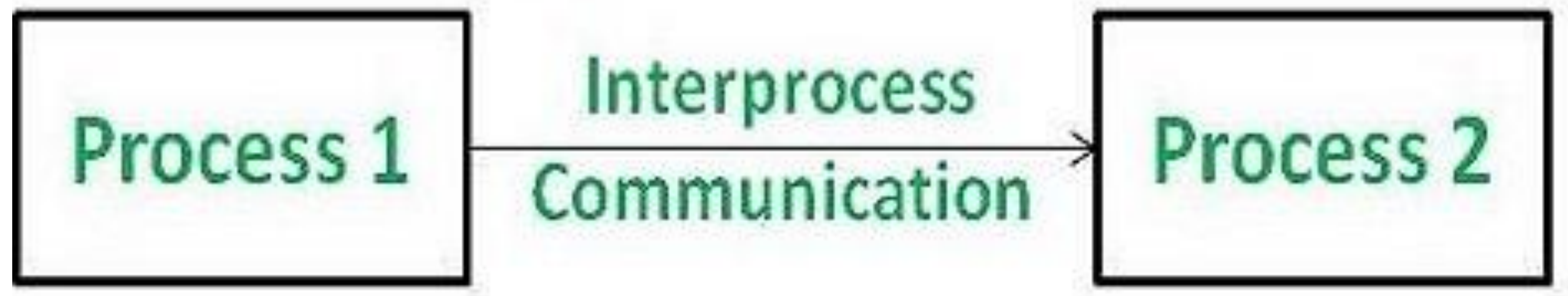

Process
Global Variables
Thread
local Variables
Code
Thread
local Variables
Code
Thread
local Variables
Code

Multiprocessing



Multithreading




Process 1
Interprocess
Communication
Process 2

Tecnofor

Web Development with Python

Flask
web development,
one drop at a time

vs

django

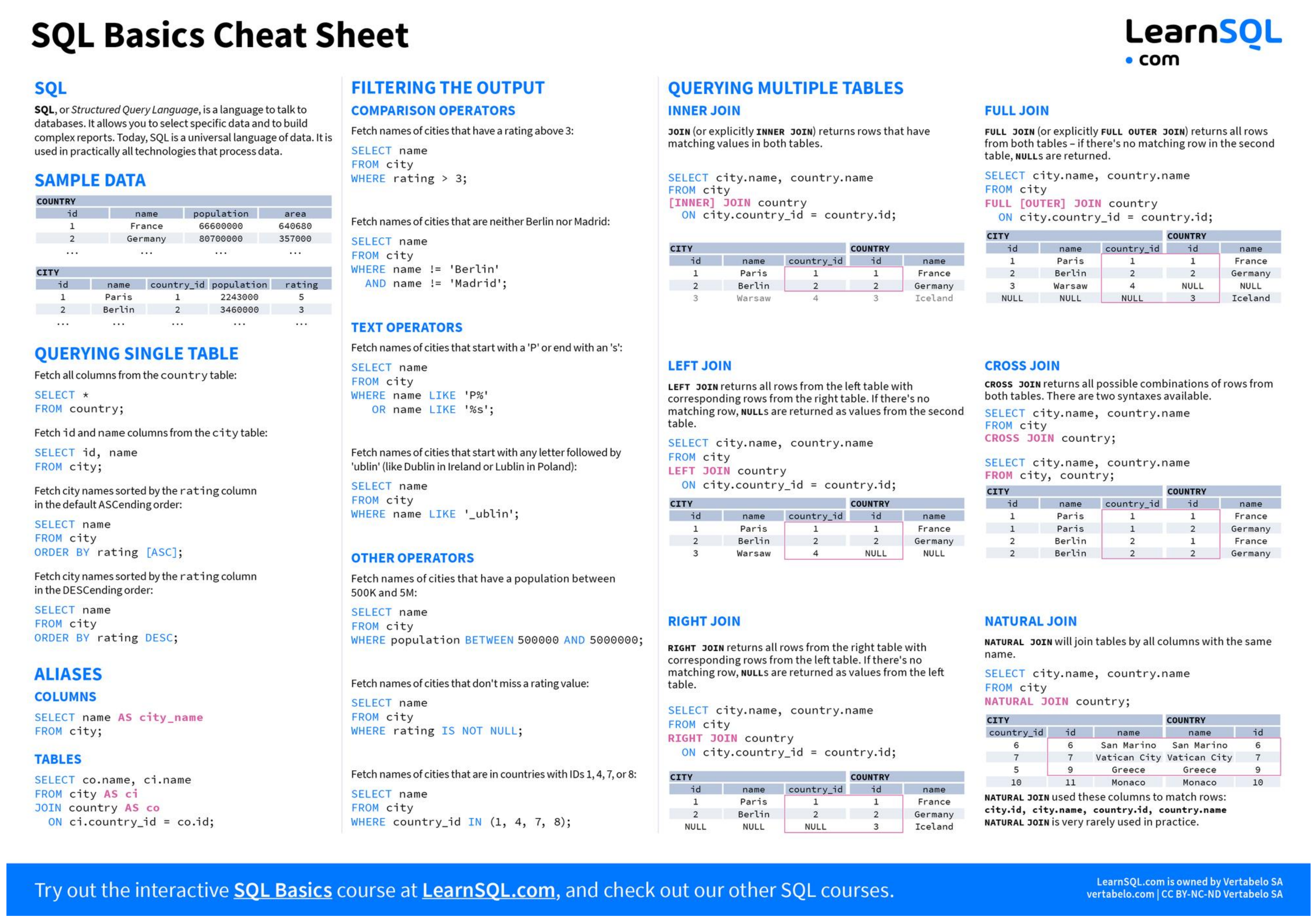
Tecnofor
by SNGULAR

Tecnofor
by SNGULAR

Tecnofor

Introduction to Databases with Python

SQLite



SQL Basics Cheat Sheet

LearnSQL • com

SQL

SQL, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

QUERYING SINGLE TABLE

Fetch all columns from the `country` table:

```
SELECT *
FROM country;
```

Fetch `id` and `name` columns from the `city` table:

```
SELECT id, name
FROM city;
```

Fetch city names sorted by the `rating` column in the default ASCending order:

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

Fetch city names sorted by the `rating` column in the DESCending order:

```
SELECT name
FROM city
ORDER BY rating DESC;
```

ALIASES

COLUMNS

```
SELECT name AS city_name
FROM city;
```

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

FILTERING THE OUTPUT

COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
  AND name != 'Madrid';
```

TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%'
   OR name LIKE '%s';
```

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

Fetch names of cities that don't miss a rating value:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

Fetch names of cities that are in countries with IDs 1, 4, 7, or 8:

```
SELECT name
FROM city
WHERE country_id IN (1, 4, 7, 8);
```

QUERYING MULTIPLE TABLES

INNER JOIN

JOIN (or explicitly **INNER JOIN**) returns rows that have matching values in both tables.

```
SELECT city.name, country.name
FROM city
[INNER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	3	Iceland

LEFT JOIN

LEFT JOIN returns all rows from the left table with corresponding rows from the right table. If there's no matching row, **NULL**s are returned as values from the second table.

```
SELECT city.name, country.name
FROM city
LEFT JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL

RIGHT JOIN

RIGHT JOIN returns all rows from the right table with corresponding rows from the left table. If there's no matching row, **NULL**s are returned as values from the left table.

```
SELECT city.name, country.name
FROM city
RIGHT JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
NULL	NULL	NULL	3	Iceland

FULL JOIN

FULL JOIN (or explicitly **FULL OUTER JOIN**) returns all rows from both tables – if there's no matching row in the second table, **NULL**s are returned.

```
SELECT city.name, country.name
FROM city
FULL [OUTER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL
NULL	NULL	NULL	3	Iceland

CROSS JOIN

CROSS JOIN returns all possible combinations of rows from both tables. There are two syntaxes available.

```
SELECT city.name, country.name
FROM city
CROSS JOIN country;
```

```
SELECT city.name, country.name
FROM city, country;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
1	Paris	1	2	Germany
2	Berlin	2	1	France
2	Berlin	2	2	Germany

NATURAL JOIN

NATURAL JOIN will join tables by all columns with the same name.

```
SELECT city.name, country.name
FROM city
NATURAL JOIN country;
```

CITY			COUNTRY	
country_id	id	name	name	id
6	6	San Marino	San Marino	6
7	7	Vatican City	Vatican City	7
5	9	Greece	Greece	9
10	11	Monaco	Monaco	10

NATURAL JOIN used these columns to match rows:
city.id, city.name, country.id, country.name
NATURAL JOIN is very rarely used in practice.

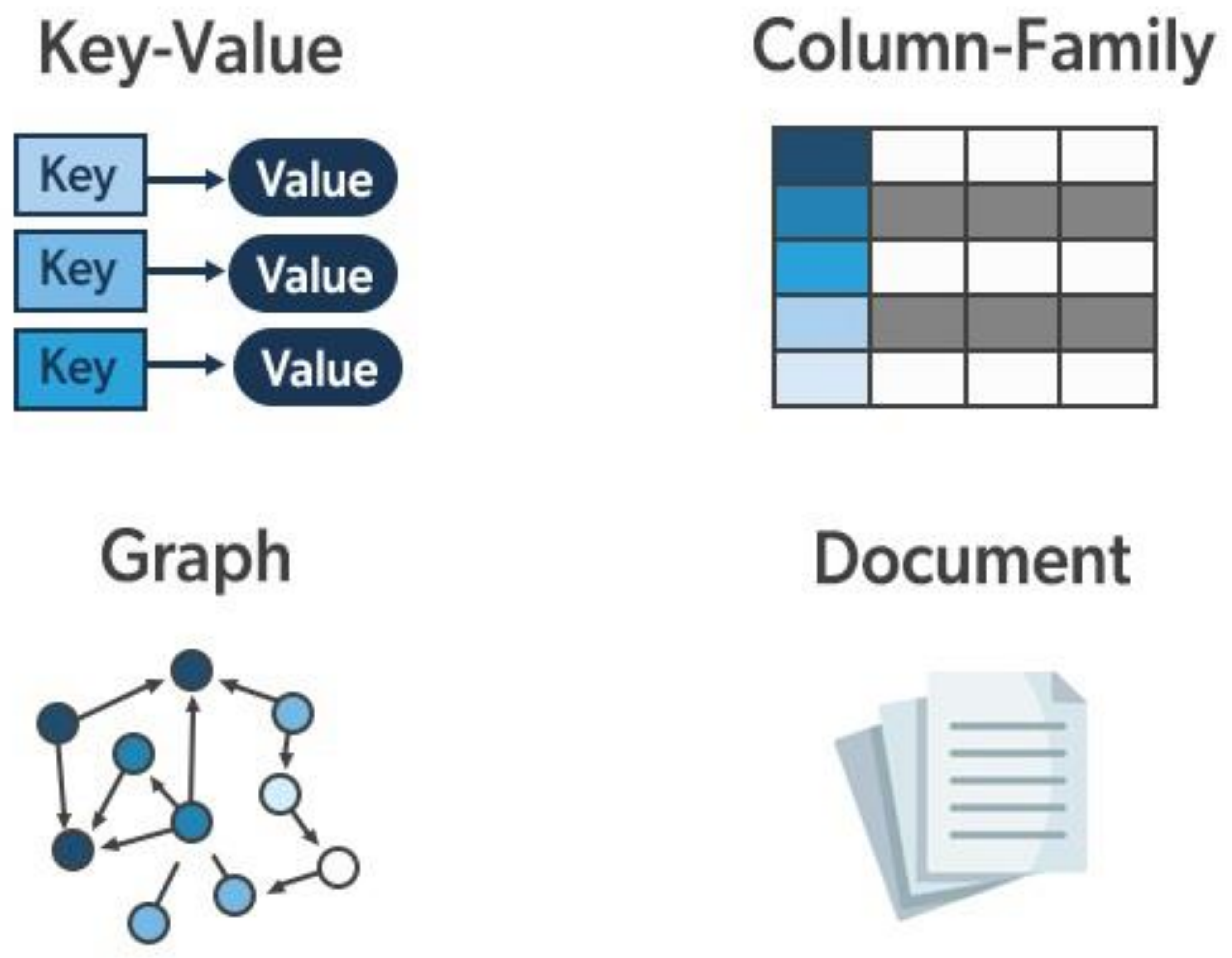
Try out the interactive **SQL Basics** course at **LearnSQL.com**, and check out our other SQL courses.

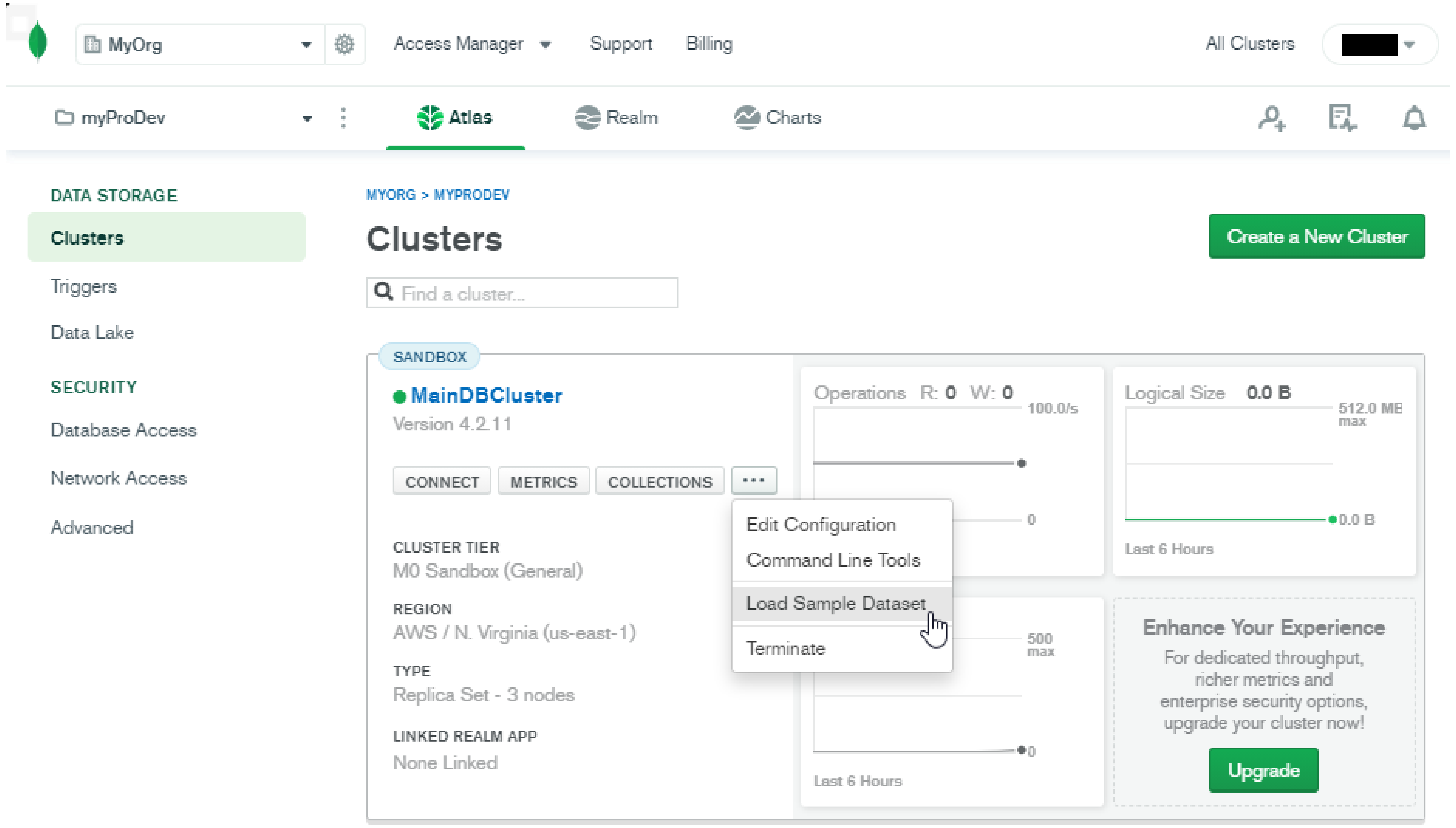
Tecnofor by SNGULAR

TIME FOR PRACTICE

mongoDB®

NoSQL


Key-Value
Key
Value
Key
Value
Key
Value
Column-Family
Graph
Document

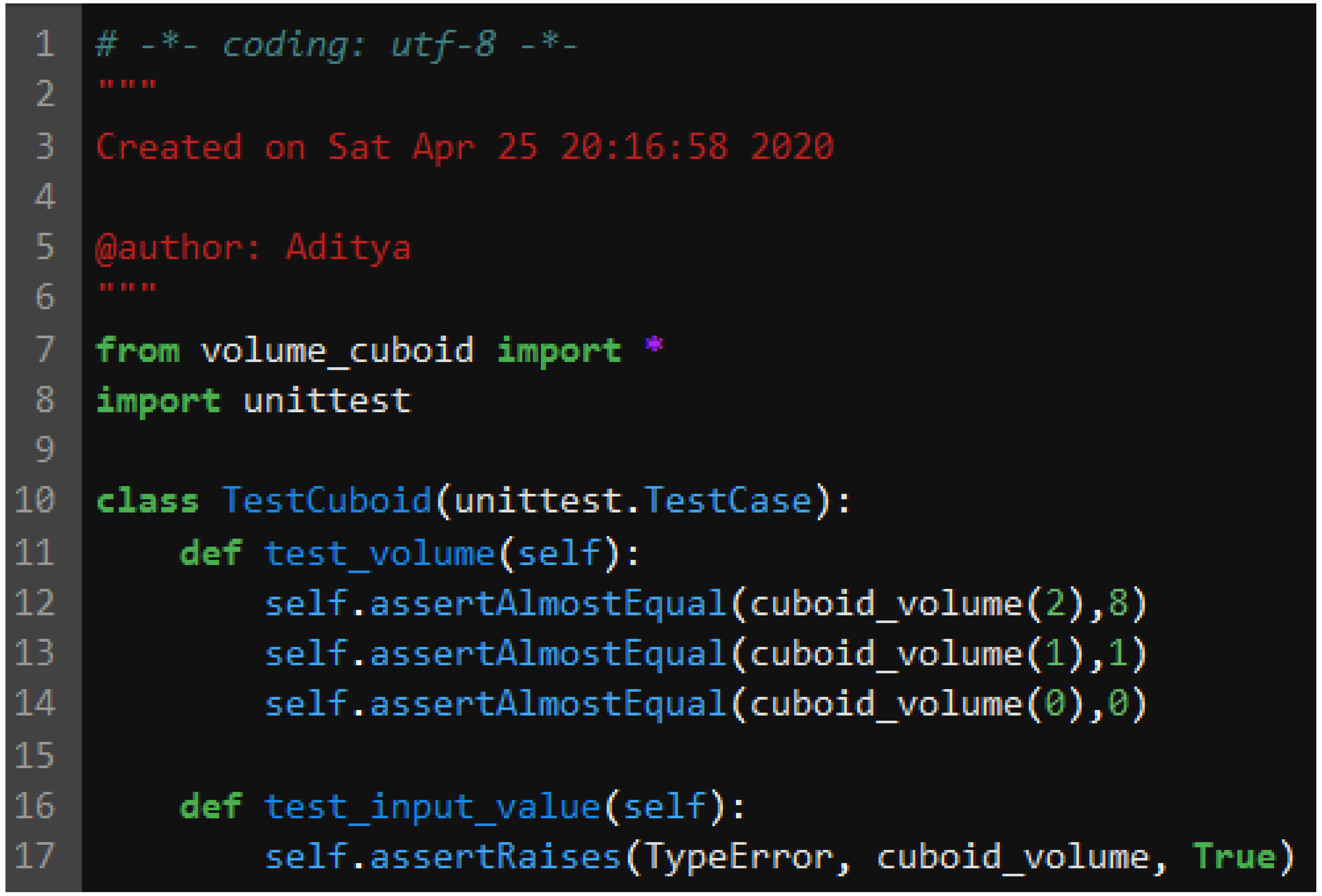


Tecnofor by SNGULAR

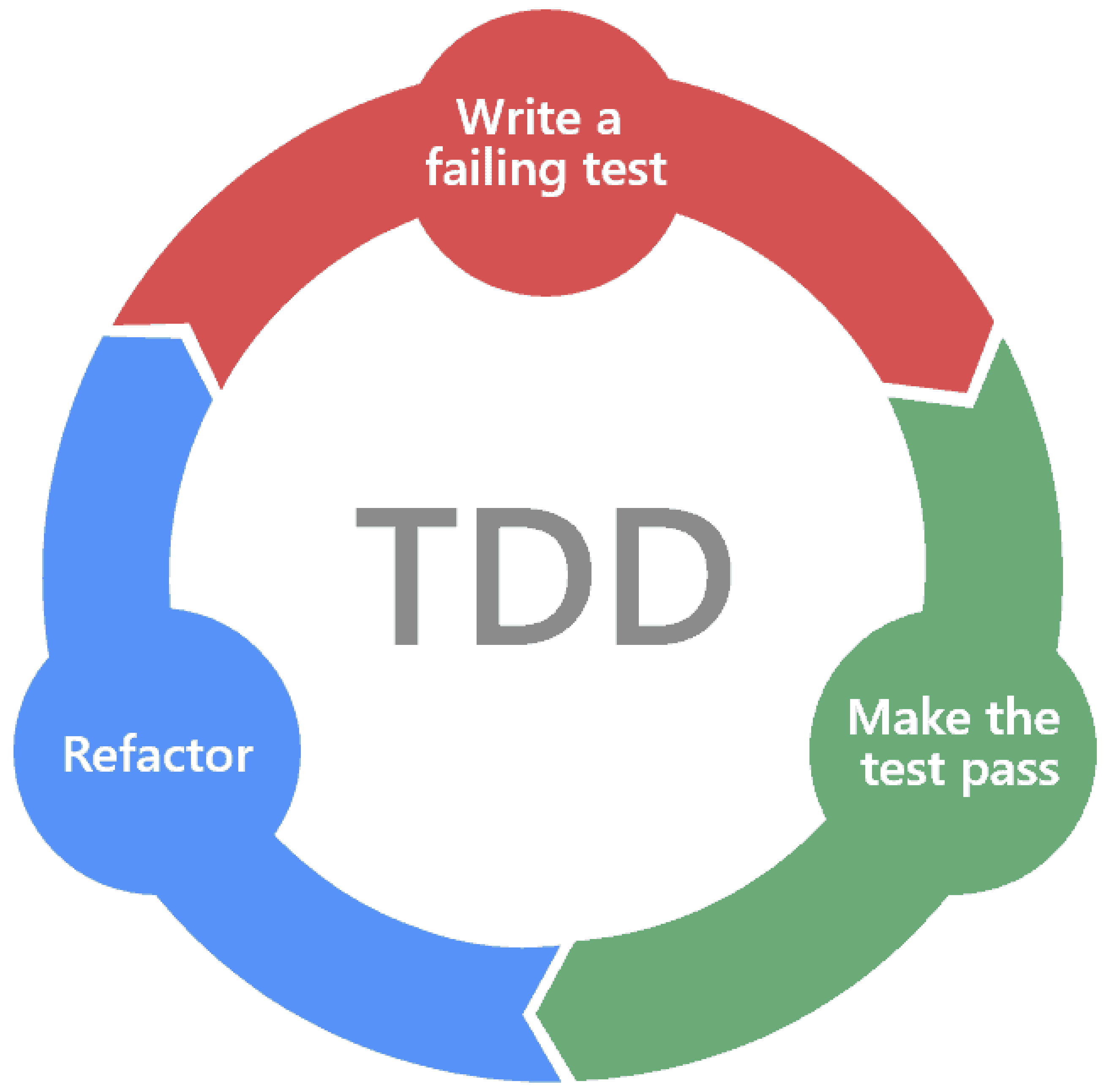
Tecnofor
by SNGULAR

Tecnofor

Unit Testing and TDD (Test Driven Development)



```python
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 25 20:16:58 2020

@author: Aditya
"""
from volume_cuboid import *
import unittest

class TestCuboid(unittest.TestCase):
    def test_volume(self):
        self.assertAlmostEqual(cuboid_volume(2),8)
        self.assertAlmostEqual(cuboid_volume(1),1)
        self.assertAlmostEqual(cuboid_volume(0),0)

    def test_input_value(self):
        self.assertRaises(TypeError, cuboid_volume, True)
```


Write a failing test
TDD
Make the test pass
Refactor

Tecnofor
by SNGULAR

¡Gracias!

Plaza de la Independencia, 8
28001-Madrid

www.tecnofor.es