Introducción a Python

CodeLab

29/11/2024

FAQ

Frequently Asked Questions

¿DE QUÉ VA A TRATAR?

Programación

Python

¿DE QUÉ NO VA A TRATAR?

Otras librerías*

Uso de frameworks

Inteligencia Artificial

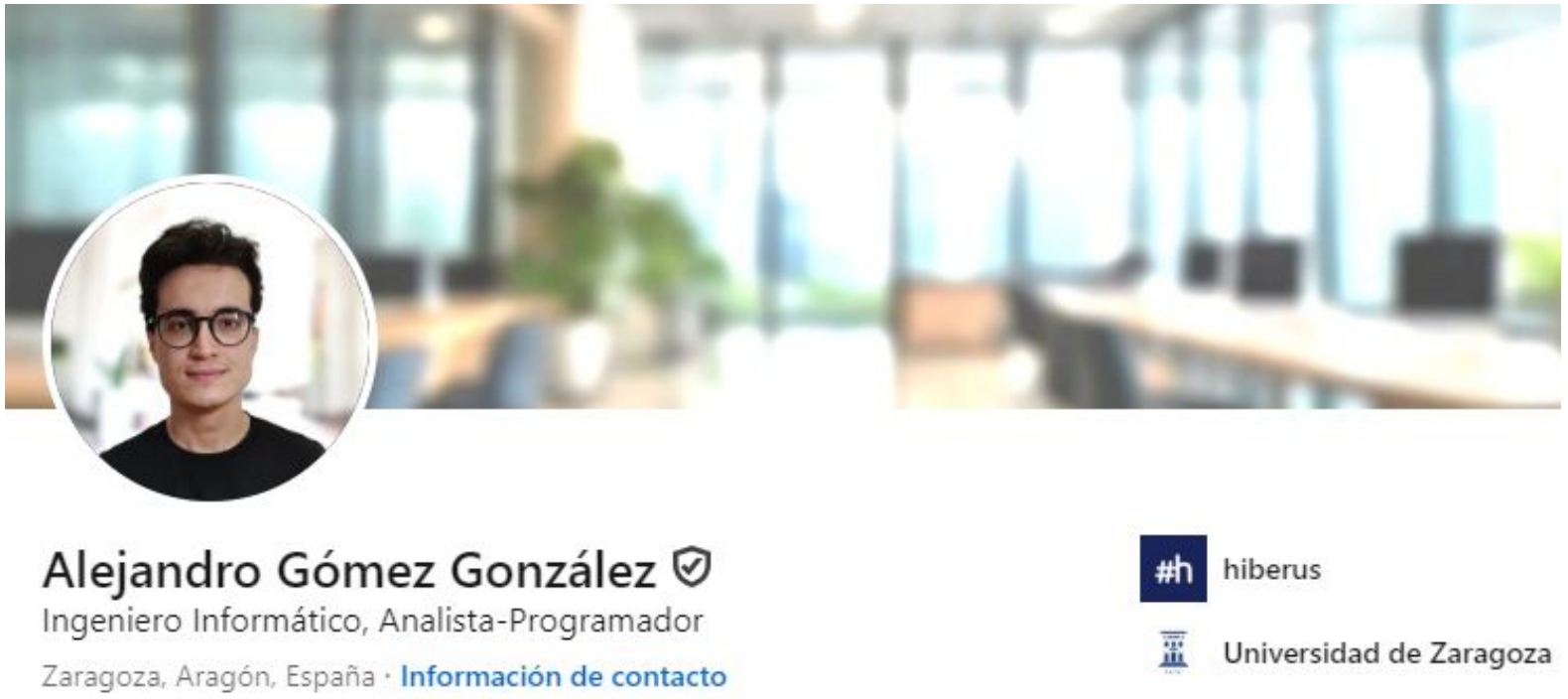
¿QUÉ TENGO QUE SABER?

Fundamentos de la programación (opcional)

* Si que se va a enseñar a importar código y gestionar dependencias

Presentación

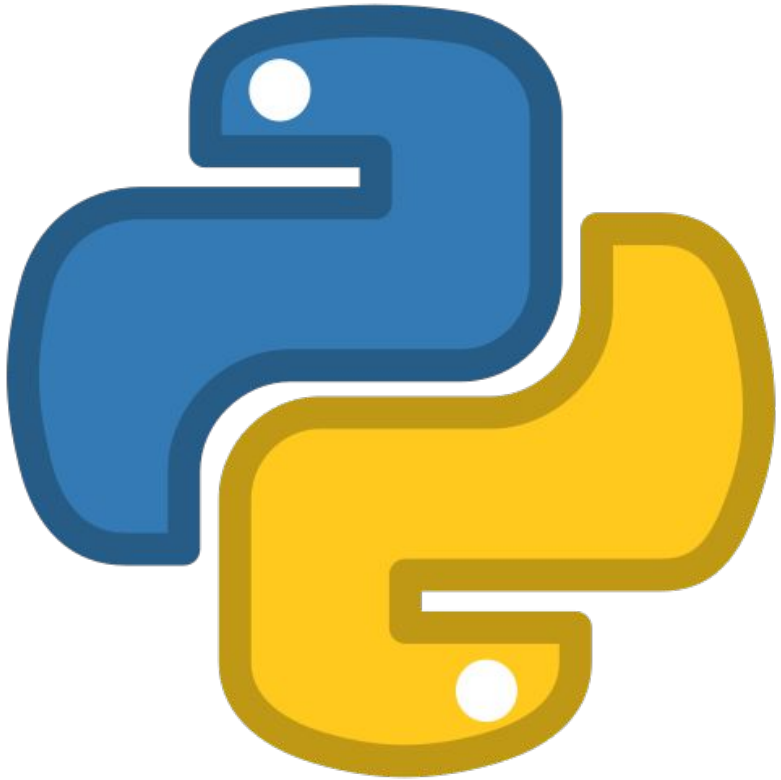
Sígueme en LinkedIn ;)



Índice

Duración aproximada 2-3h

1. Historia y filosofía
2. Sintaxis y ortografía
3. Importación de código
4. Entresijos
5. Anuncio



Entornos de desarrollo

Se vienen cositas

Google Colab

https://colab.research.google.com/#create=true

PyCharm Community Edition

Virtual Studio Code

https://vscode.dev/

Historia y filosofía

CodeLab

29/11/2024

Historia y filosofía

Origen

Van Rossum

En los 80 fue parte del equipo de desarrollando el lenguaje de programación ABC en el CWI

An example function to collect the set of all `words` in a document:

```
HOW TO RETURN words document:
   PUT {} IN collection
   FOR line IN document:
      FOR word IN split line:
         IF word not.in collection:
            INSERT word IN collection
   RETURN collection
```

Historia y filosofía

Origen

Van Rossum

Los caballeros de la mesa cuadrada (1975)

Versión 0.9.0 en 1991

Historia y filosofía

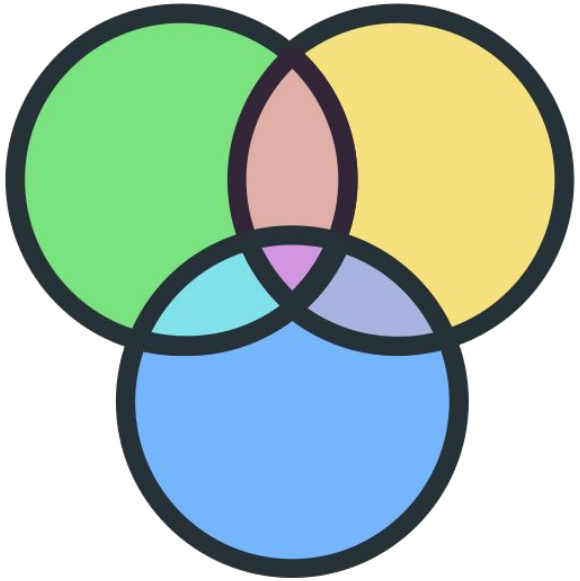
¿Qué tipo de lenguaje de programación es Python?

Según su nivel:
- Bajo nivel
- Medio nivel
- Alto nivel

Según su ejecución:
- Compilado
- Interpretado

Según su tipado:
- Fuertemente tipado
- Débilmente tipado

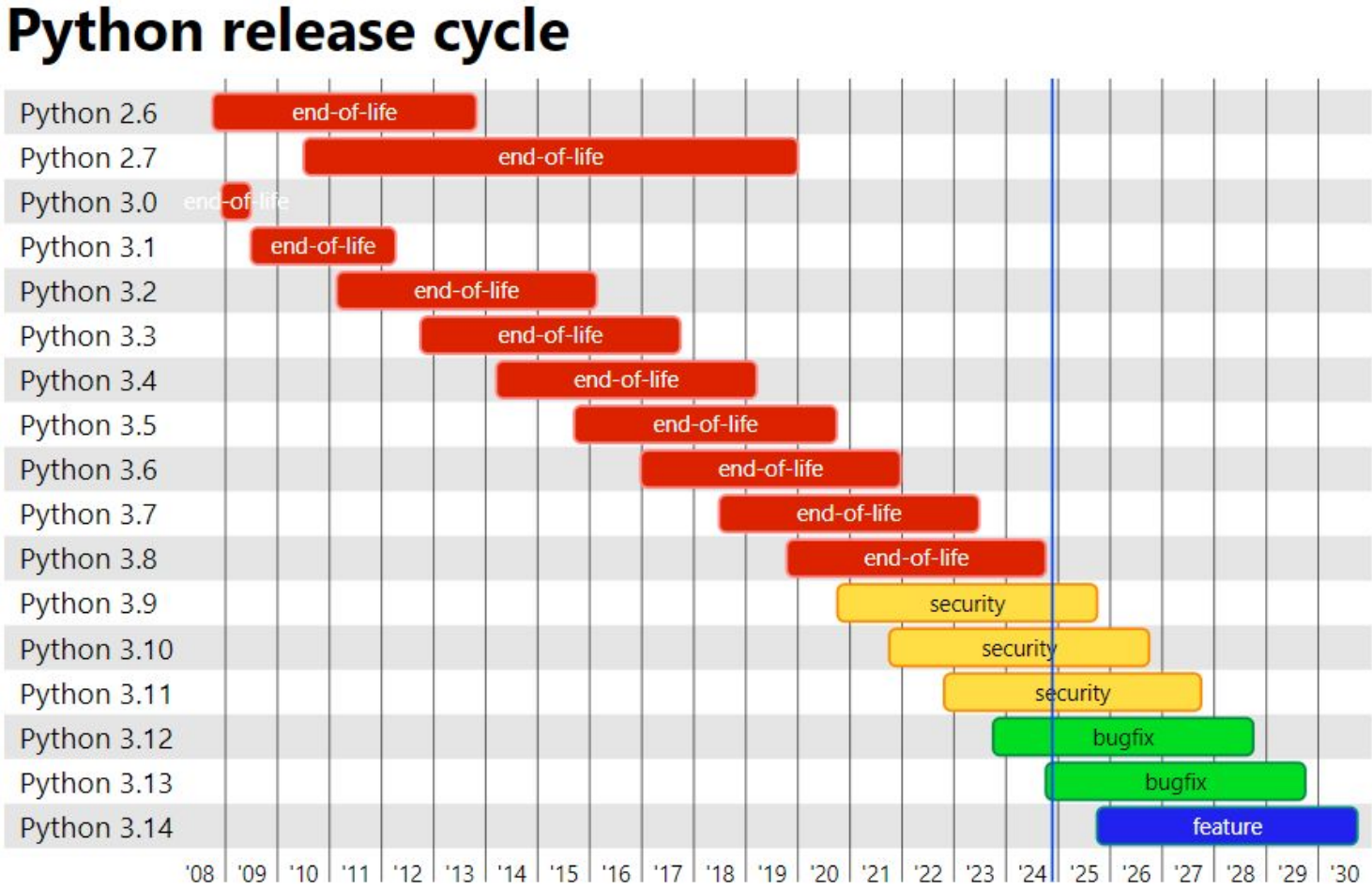
Según su programación:
- Programación orientada a objetos (POO)
- Programación funcional
- Programación imperativa
- Programación declarativa



Historia y filosofía

Historial de versiones

Versión 0.9.0 en 1991
Versión 1.0.0 en 1994
Versión 2.0.0 en 2000
Versión 3.0.0 en 2008

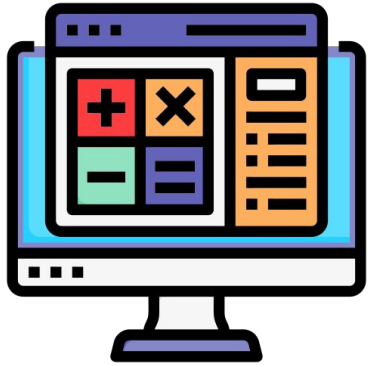


A Noviembre de 2024

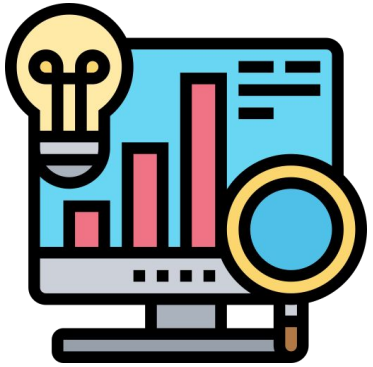
Historia y filosofía

Usos

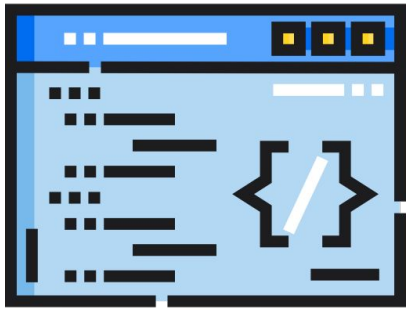
Desarrollo Web



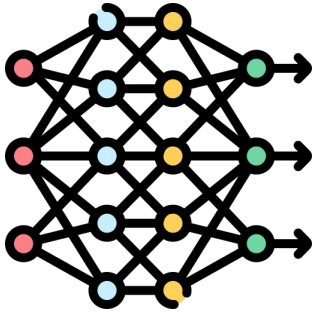
Aplicaciones de Escritorio



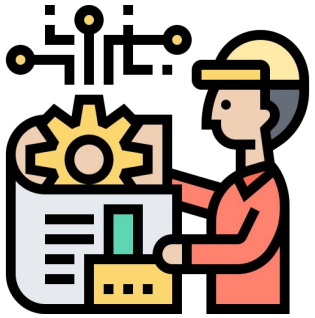
Ciencia de Datos y Análisis



Automatización y Tareas de Scripting



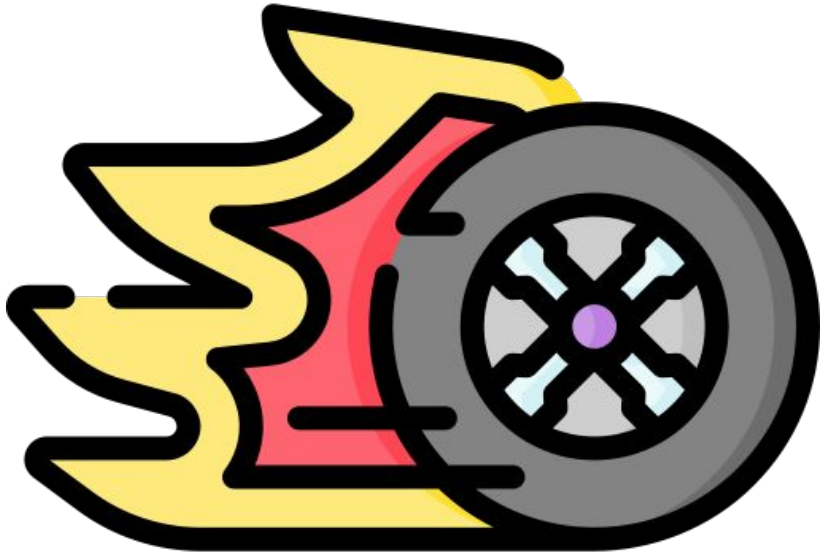
Inteligencia Artificial y Aprendizaje Automático



Aplicaciones Científicas y de Ingeniería

Historia y filosofía

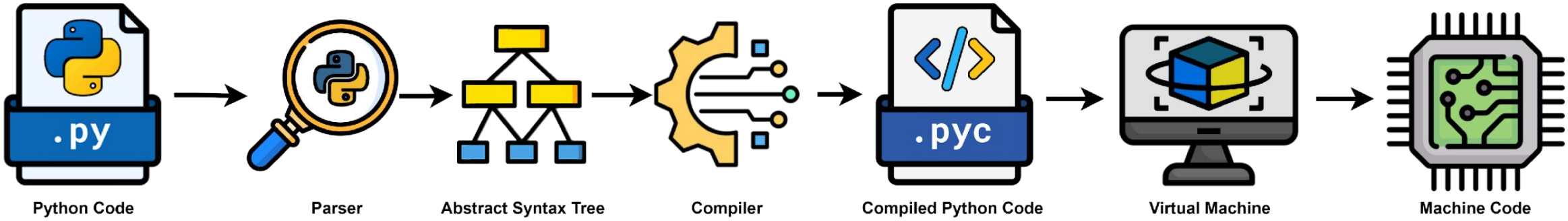
Puntos débiles



Rendimiento extremo

Historia y filosofía

Ejecución de Python



Sintaxis y ortografía:
Lo básico

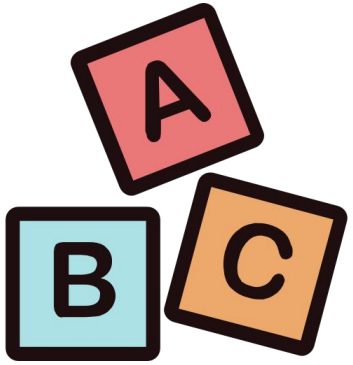
CodeLab

29/11/2024

Sintaxis y ortografía

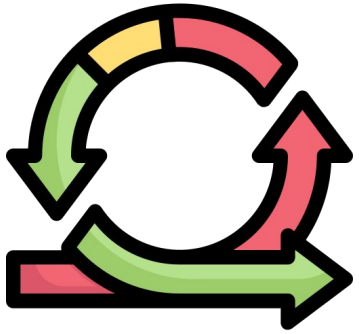
¿Qué hay en el código?

Comentarios



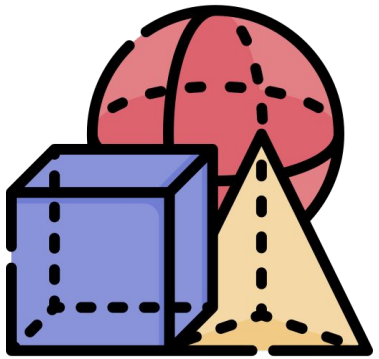
Variables

Operaciones



Estructuras de control

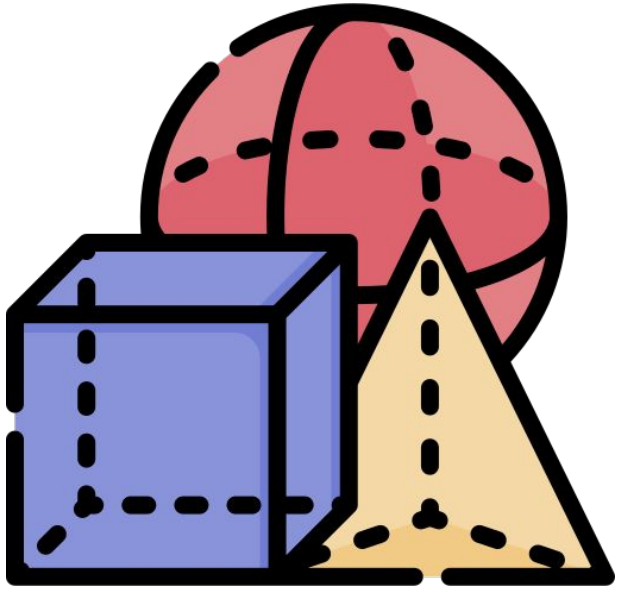
Funciones



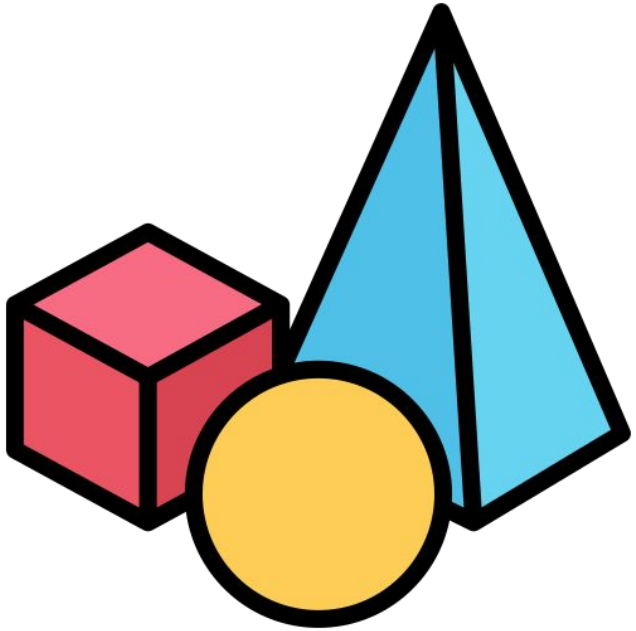
Clases

Sintaxis y ortografía

Las clases se instancian en objetos



Clases



Objetos

Sintaxis y ortografía

Comentarios

example.py

```python
# Comment: This is a single-line comment

"""
Multi-line comment:
This is a comment that spans multiple lines.
"""
```

Sintaxis y ortografía

Declaración de variables

example.py

```python
# Integers (int)
integer_number = 42

# Floats (float)
decimal_number = 3.14

# Strings (str)
name = "John"
message = 'Hello, World!'

# Booleans (bool)
is_true = True
is_false = False

# Multiple variables
x, y, z = 1, 2, 3

# Variables with names containing underscores (snake_case)
user_name = "user123"
```

Sintaxis y ortografía

Sugerencias de tipado (Type Hints)

example.py

```python
# Integers (int)
integer_number: int = 42

# Floats (float)
decimal_number: float = 3.14

# Strings (str)
name: str = "John"
message: str = 'Hello, World!'

# Booleans (bool)
is_true: bool = True
is_false: bool = False

# Multiple variables
# Only single target be annotated!!

# Variables with names containing underscores (snake_case)
user_name = "user123"
```

Sintaxis y ortografía

¿Se puede hacer esto?

example.py

```python
# Variable initialized annotated as an integer
x: int = 42

# Variable reassigned as an string
x: str = "The answer to all things"

```

Sintaxis y ortografía

Entrada y salida de datos por terminal

example.py

```python
# Get user input from terminal
name = input("Enter your name: ")

# Display content in terminal
print(f"{name} is learning python")

```

Sintaxis y ortografía

Operaciones básicas de strings

example.py

```python
# String Formatting
name = "Alice"
age = 30
formatted_message = f"Hello, my name is {name} and I am {age} years old."
print(formatted_message)
# Hello, my name is Alice and I am 30 years old.

# Concatenation
message = "Python" + " is " + "cool!"
print(message)
# Python is cool!

# String Methods
uppercase_message = message.upper()
print(f"Uppercase: {uppercase_message}")
# Uppercase: PYTHON IS COOL!

lowercase_message = message.lower()
print(f"Lowercase: {lowercase_message}")
# Lowercase: python is cool!

# String Length
message = "Python is amazing"
length = len(message)
print(f"The length of the message is {length} characters")
# The length of the message is 17 characters
```

Sintaxis y ortografía

Operaciones básicas

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

example.py

```python
# Boolean variables
is_true = True
is_false = False

# Logical AND
result_and = is_true and is_false
# result_and = False

# Logical OR
result_or = is_true or is_false
# result_or = True

# Logical NOT
result_not_true = not is_true
# result_not_true = False

```

Sintaxis y ortografía

Operaciones básicas

a es igual que b	a == b
a no es igual que b	a != b
a es mayor que b	a > b
a es menor que b	a < b
a es igual o mayor que b	a >= b
a es igual o menor que b	a <= b

example.py

```python
# Comparisons
a = 5
b = 10

# Equality
equal = a == b
# equal = False

# Inequality
not_equal = a != b
# not_equal = True

# Greater than
greater_than = a > b
# greater_than = False

# Less than
less_than = a < b
# less_than = True

# Greater than or equal to
greater_equal = a >= b
# greater_equal = False

# Less than or equal to
less_equal = a <= b
# less_equal = True

```

Sintaxis y ortografía

Operaciones básicas numéricas

example.py

```python
# Basic Arithmetic Operations in Python (I)

# Addition
num1 = 10
num2 = 3
sum_result = num1 + num2
print("Sum:", sum_result)
# Sum: 13

# Subtraction
difference = num1 - num2
print("Difference:", difference)
# Difference: 7

# Multiplication
product = num1 * num2
print("Product:", product)
# Product: 30

```

example.py

```python
# Basic Arithmetic Operations in Python (II)

# Division
quotient = num1 / num2
print("Quotient:", quotient)
# Quotient: 3.3333333333333335

# Integer Division (Floor Division)
integer_quotient = num1 // num2
print("Integer Quotient:", integer_quotient)
# Integer Quotient: 3

# Modulo (Remainder)
remainder = num1 % num2
print("Remainder:", remainder)
# Remainder: 1

# Exponentiation
power_result = num1 ** num2
print("Power:", power_result)
# Power: 1000

```

Sintaxis y ortografía

Declaración de condicionales

example.py

```python
# Conditional Statements (if, elif, else)
x = 10
y = 5
if x > y:
    print(f"{x} is greater than {y}")
elif x < y:
    print(f"{x} is less than {y}")
else:
    print(f"{x} is equal to {y}")
# 10 is greater than 5

```

Sintaxis y ortografía

Declaración de bucles while

example.py

```python
# While loop
i = 0
while i < 5:
    print(f"Countdown: {i}")
    i += 1
"""
Countdown: 0
Countdown: 1
Countdown: 2
Countdown: 3
Countdown: 4
"""
```

Sintaxis y ortografía

Declaración de bucles for

example.py

```python
for i in [0, 1, 2, 3, 4]:
    print(f"Iteration {i}")

"""
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
"""

```

Sintaxis y ortografía

Declaración de bucles for

example.py

```python
# For loop
for i in range(5):
    print(f"Iteration {i}")
"""
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
"""

```

example.py

```python
# For loop
for i in range(0, 5):
    print(f"Iteration {i}")
"""
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
"""

```

Sintaxis y ortografía

Declaración de bucles for

example.py

```python
# For loop
for i in range(-2, 3):
    print(f"Iteration: {i}")
"""
Iteration: -2
Iteration: -1
Iteration: 0
Iteration: 1
Iteration: 2
"""

```

example.py

```python
# For loop
for i in range(0, 10, 2):
    print(f"Iteration {i}")
"""
Iteration 0
Iteration 2
Iteration 4
Iteration 6
Iteration 8
"""

```

Sintaxis y ortografía

Sentencia pass

pass = no hacer nada

example.py

```
1 if (number > 100):
2     # TODO: Decide how to treat this case
3     pass
4
```

Se suele utilizar para marcar algo que hay que hacer

Sintaxis y ortografía

Sentencia pass

example.py

```python
if evaluate_certain_condition():
  # Nothing to be done here
  pass
else:
  # Need to do something here
  print("Im doing things...")

```

No hagáis estas cosas…

Sintaxis y ortografía

Sentencia pass

example.py

```python
if not evaluate_certain_condition():
  # Need to do something here
  print("Im doing things...")

```

… hacer esto mejor

Sintaxis y ortografía

Sentencia continue

continue = saltar a la siguiente iteración del bucle

example.py

```
1 for i in range(5):
2     if i == 2:
3         continue
4     print(i)
5 """
6 0
7 1
8 3
9 4
10 """
11
```

Cuando llega al continue, salta a la siguiente iteración y no pasa por el

Sintaxis y ortografía

Sentencia break

break = salir del bucle

example.py

```python
for number in [10, 11, 12, 13, 14, 15]:
  if (number % 3) == 0:
    divisible_by_3 = number
# divisble_by_3 = 15

```

Itera sobre toda la lista

example.py

```python
for number in [10, 11, 12, 13, 14, 15]:
  if (number % 3) == 0:
    divisible_by_3 = number
    break
# divisble_by_3 = 12

```

Cuando llega al break, sale del bucle

Sintaxis y ortografía

Declaración de funciones sin argumentos

example.py

```
1 # Function with Arguments and a Return Value
2 def greet():
3     print("Hello world!")
4
5 # Calling the function
6 greet()
7 # Hello world!
8
```

Sintaxis y ortografía

Declaración de funciones con argumentos

example.py

```python
# Function with Arguments and a Return Value
def add(a, b):
    result = a + b
    return result

# Calling the function
sum_result = add(5, 3)
print(f"The sum is: {sum_result}")
# The sum is: 8

```

Sintaxis y ortografía

Declaración de funciones con argumentos opcionales

example.py

```python
# Function with Optional Arguments and a Return Value
def calculate_price(total, discount=0):
    discounted_price = total - (total * discount)
    return discounted_price

# Calling the function with and without a discount
total_price = calculate_price(100)
discounted_price = calculate_price(100, 0.1)

print(f"Total price: {total_price}")
# Total price: 100
print(f"Discounted price: {discounted_price}")
# Discounted price: 90.0

```

Sintaxis y ortografía

Declaración de funciones con argumentos variables

example.py

```python
# Function with Variable-Length Arguments
def average(*args):
    total = sum(args)
    return total / len(args)

# Calling the function with different numbers of arguments
average1 = average(4, 5, 6)
average2 = average(10, 20, 30, 40, 50)

print(f"Average 1: {average1}")
# Average 1: 5.0
print(f"Average 2: {average2}")
# Average 2: 30.0

```

Sintaxis y ortografía

Declaración de funciones con sugerencias de tipado

example.py

```
1 def add_integers(x: int, y: int):
2     return x + y
3
4
5 z = add_integers(3, 2)
6
```

Sintaxis y ortografía

Declaración de funciones con sugerencias de tipado

example.py

```
1 def add_integers(x: int, y: int):
2   return x + y
3
4
5 z = add_integers("eggs", "potatoes")
6
```

Sintaxis y ortografía

Funciones documentadas

example.py

```python
def add_integers(x: int, y: int) -> int:
    """
    This function adds two integers

    :param int x: the first variable of the sum operation
    :param int y: the second variable of the sum operation
    :return int: the sum of variable x and y
    """
    return x + y

```

Sintaxis y ortografía

Funciones como funciones como parámetro

example.py

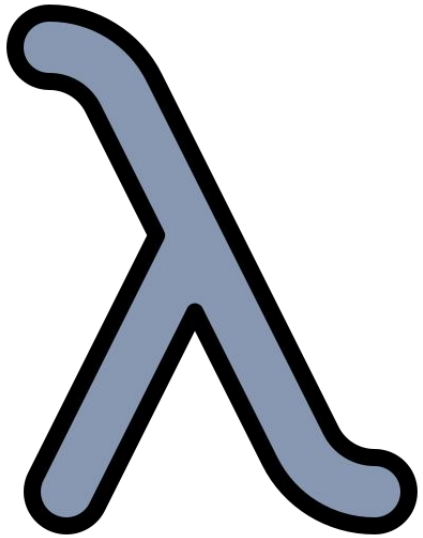
```python
# Function that takes another function as an argument
def apply_function(func, x):
    result = func(x)
    return result

# Example function to be passed as an argument
def square(number):
    return number ** 2

# Another example function to be passed as an argument
def half(number):
    return number / 2

# Calling apply_function with the square function
result = apply_function(square, 5)
# result = 25

# Calling apply_function with the half function
result = apply_function(half, 5)
# result = 2.5

# Calling apply_function with a lambda function
result = apply_function(lambda x: x*2, 5)
# result = 10

```

Sintaxis y ortografía

Funciones lambda

example.py

```python
add_lambda = lambda x, y: x + y

def add_function(x, y):
  return x + y
```



example.py

```python
# Single parameter lambda
duplicate = lambda x: x * 2
print(duplicate(2))
# 4

# Two parameter lambda
add = lambda x, y: x + y
print(add(2, 3))
# 5

# Conditional return
find_max = lambda x, y: x if x > y else y
print(find_max(2, 3))
# 3

# Boolean return
is_even = lambda x: x % 2 == 0
print(is_even(4))
# True

```

Sintaxis y ortografía

Declaración e instanciación de clases

example.py

```python
# Class declaration
class FormulaOneCar:
    def __init__(self, model, driver):
        self.model = model
        self.driver = driver
        self.is_running = False

    def start_engine(self):
        self.is_running = True

    def stop_engine(self):
        self.is_running = False

# Class instantiation
my_ferrari = FormulaOneCar("Ferrari SF23", "Carlos Sainz")

# Accessing class attributes
print(f"My F1 car is a {my_ferrari.model} driven by {my_ferrari.driver}.")
# My F1 car is a Ferrari SF23 driven by Carlos Sainz.

print(f"Is the engine running? {my_ferrari.is_running}")
# Is the engine running? False

# Calling class methods
my_ferrari.start_engine()
print(f"Is the engine running? {my_ferrari.is_running}")
# Is the engine running? True

```

¡Hagamos una calculadora!

CodeLab

29/11/2024

Volvemos a...

Sintaxis y ortografía:
Estructuras de datos

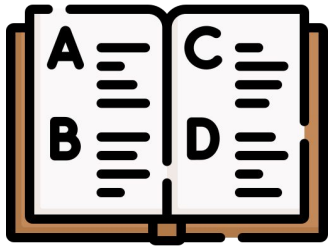
CodeLab

29/11/2024

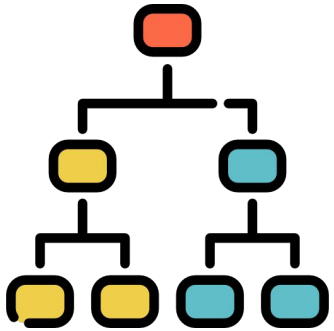
Sintaxis y ortografía

Estructuras de datos

Grafos

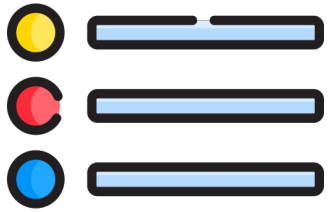


Diccionarios

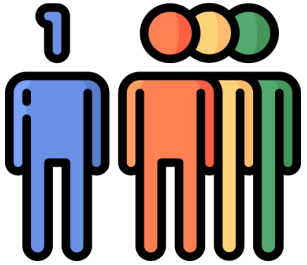


Árboles

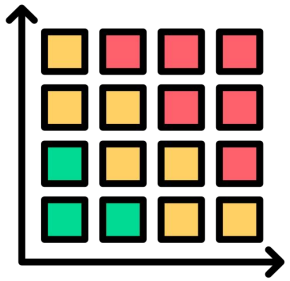
No lineales



Listas



Colas



Matrices

Lineales

Sintaxis y ortografía

Estructuras de datos: lista, tupla, set, diccionario

example.py

```python
# Lists are versatile and commonly used for storing collections of items of varying types.
my_list = [1, 2, 3, 'apple', 'banana']

# Tuples are used to store INMUTABLE sequences of elements, often for data that should not be changed.
my_tuple = (1, 2, 3, 'apple', 'banana')

# Sets are used for storing UNIQUE elements and are useful for tasks like removing duplicates from a list.
my_set = {1, 2, 3, 4, 5}

# Dictionaries are used for mapping keys to values and are helpful for representing structured data.
my_dict = {
  'name': 'Alice',
  'age': 30,
  'city': 'New York'
}

```

Sintaxis y ortografía

Estructuras de datos: diccionarios anidados

example.py

```python
hiberus_ecosystem = {
  'location': {
    'city': 'Zaragoza',
    'postal_code': 50009,
    'address': 'P. de Isabel la Católica, 6'
  }
  'workers': {'Alejandro G', 'Alejandro P', 'Raúl J'}
}

```

Sintaxis y ortografía

Operaciones con listas

example.py

```python
# Creating a list
my_list = [1, 2, 3, 4, 5]

# Accessing elements by index
first_element = my_list[0]
# first_element = 1
last_element = my_list[-1]
# last_element = 5

# Slicing a list
subset = my_list[1:4]
# subset = [2, 3, 4]

# Modifying elements
my_list[2] = 99  # Modifies the third element
# my_list = [1, 2, 99, 4, 5]

# Append element to the end of the list
my_list.append(6)
# my_list = [1, 2, 99, 4, 5, 6]

# Extend list with other list
my_list.extend([7, 8, 9])
# my_list = [1, 2, 99, 4, 5, 6, 7, 8, 9]

# Remove element by value
my_list.remove(99)
# my_list = [1, 2, 4, 5, 6, 7, 8, 9]

# Remove element by list position
popped_element = my_list.pop(2)
# my_list = [1, 2, 5, 6, 7, 8, 9]

```

example.py

```python
my_list = [1, 2, 3, 4, 5]

# Finding elements
element_index = my_list.index(5)

# element_index = 4

# Checking membership
is_3_present = 3 in my_list
# is_3_present = True
is_9_present = 9 in my_list
# is_3_present = False

# Sort list in ascending order
my_list.sort()
# my_list = [1, 2, 3, 4, 5]

# Sort list in reverse order
my_list.reverse()
# my_list = [5, 4, 3, 2, 1]

# Shallow copying a list
new_list = my_list.copy()

# Length of a list
list_length = len(my_list)
# list_length = 5

# Removes all elements
my_list.clear()
# my_list = []

```

Sintaxis y ortografía

Operaciones con tuplas

example.py

```python
# Creating a tuple
fruits = ('apple', 'banana', 'cherry',
'date')

# Accessing elements by index
first_fruit = fruits[0]
# first_fruit = 'apple'
last_fruit = fruits[-1]
# last_fruit = 'date'

# Slicing a tuple
selected_fruits = fruits[1:3]
# selected_fruits = ('banana', 'cherry')

# Checking membership
is_melon_in_fruits = 'melon' in fruits
# is_melon_in_fruits = False

# Counting elements
count_cherry = fruits.count('cherry')
# count_cherry = 1

# Finding the index of an element
index_banana = fruits.index('banana')
# index_banana = 1

# Length of a tuple
num_fruits = len(fruits)
# num_fruits = 4

```

Sintaxis y ortografía

Operaciones con sets

example.py

```python
# Creating a Set
fruits = {"apple", "banana", "cherry"}

# Adding Elements
fruits.add("orange")

# Removing Elements
fruits.remove("banana")

# Checking Membership
is_melon_in_fruits = "melon" in fruits
# is_melon_in_fruits = False

set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Union of sets
union_set = set1.union(set2)
# union_set = {1, 2, 3, 4, 5}

# Intersection of sets
intersection_set = set1.intersection(set2)
# intersection_set = {3}

# Set difference
difference_set = set1.difference(set2)
# difference_set = {1, 2}

# Length of a Set
num_fruits = len(fruits)

```

Sintaxis y ortografía

Operaciones con diccionarios

example.py

```python
# Dictionary Creation
person = {
  "name": "Alex",
  "age": 24,
  "city": "Zaragoza"
}

# Accessing Elements
# Accessing information about Alex
print(f"{person['name']} is {person['age']} years old and lives in {person['city']}")
# Alex is 24 years old and lives in Zaragoza

# Modifying Elements
person["age"] = 25

# Adding new element
person["height"] = 184

# Checking Key Existence
if "height" in person:
    print("Alex's height is:", person["height"])

# Deleting Elements
person.pop("age")

# Dictionary Length (number of elements in the dictionary)
num_elements = len(person)

```

Sintaxis y ortografía

Inicialización de lista, tupla, set y diccionario

example.py

```python
# Empty initialization
empty_list = list()
empty_tuple = tuple()
empty_set = set()
empty_dictionary = dict()

```

example.py

```python
# Regular initialization
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
my_set = {1, 2, 3}
my_dict = {1: False, 2: True, 3: False}

```

Sintaxis y ortografía

Conversiones de lista a otros tipos

example.py

```python
# Transform a tuple into a list
my_list_from_tuple = list(my_tuple)
print(my_list_from_tuple)
# [1, 2, 3]

# Transform a set to list
my_list_from_set = list(my_set)
print(my_list_from_set)
# [1, 2, 3]

# Transform a dictionary into a list of keys and values
keys_list = list(my_dict.keys())
print(keys_list)
# [1, 2, 3]

values_list = list(my_dict.values())
print(values_list)
# [False, True, False]

```

Sintaxis y ortografía

Conversiones de tupla a otros tipos

example.py

```python
# Transform a list into a tuple
my_tuple_from_list = tuple(my_list)
print(my_tuple_from_list)
# (1, 2, 3)

# Transform a set into a tuple
my_tuple_from_set = tuple(my_set)
print(my_tuple_from_set)
# (1, 2, 3)

# Transform a dictionary into a tuple of keys and values
keys_tuple = tuple(my_dict.keys())
print(keys_tuple)
# (1, 2, 3)

values_tuple = tuple(my_dict.values())
print(values_tuple)
#(False, True, False)

```

Sintaxis y ortografía

Conversiones de set a otros tipos

example.py

```
# Transform a list into a set
my_set_from_list = set(my_list)
print(my_set_from_list)
# {1, 2, 3}

# Transform a tuple into a set
my_set_from_set = set(my_tuple)
print(my_set_from_set)
# {1, 2, 3}

# Transform a dictionary into a set of keys and values
keys_set = set(my_dict.keys())
print(keys_set)
# {1, 2, 3}

values_tuple = set(my_dict.values())
print(values_tuple)
# {False, False}

```

Sintaxis y ortografía

Conversiones de diccionario a otros tipos

example.py

```python
person = dict(name = "Alex", age = 24, city = "Zaragoza")
print(person)
# {'name': 'Alex', 'age': 24, 'city': 'Zaragoza'}

# Transform a list of lists into a dictionary
list_of_lists = [['name', 'Alex'], ['age', 24], ['city', 'Zaragoza']]
dict_from_list_of_lists = dict(list_of_lists)
print(dict_from_list_of_lists)
# {'name': 'Alex', 'age': 24, 'city': 'Zaragoza'}

# Transform a list of tuples into a dictionary
list_of_tuples = [['name', 'Alex'], ['age', 24], ['city', 'Zaragoza']]
dict_from_list_of_tuples = dict(list_of_tuples)
print(dict_from_list_of_tuples)
# {'name': 'Alex', 'age': 24, 'city': 'Zaragoza'}

# Transform a tuple of lists into a dictionary
tuple_of_lists = (['name', 'Alex'], ['age', 24], ['city', 'Zaragoza'])
dict_from_tuple_of_lists = dict(tuple_of_lists)
print(dict_from_tuple_of_lists)
# {'name': 'Alex', 'age': 24, 'city': 'Zaragoza'}

# Transform a tuple of tuples into a dictionary
tuple_of_tuples = (('name', 'Alex'), ('age', 24), ('city', 'Zaragoza'))
dict_from_tuple_of_tuples = dict(tuple_of_tuples)
print(dict_from_tuple_of_tuples)
# {'name': 'Alex', 'age': 24, 'city': 'Zaragoza'}

```

¡Añadamos un histórico a la calculadora!

CodeLab

29/11/2024

Sintaxis y ortografía:
Comprensión de listas, tuplas, dict

CodeLab

29/11/2024

Sintaxis y ortografía

Sintaxis de comprensiones

example.py

```
1 list = [value for element in iterable]
2 set = {value for element in iterable}
3 dictionary = {key: value for element in iterable}
4
```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
# Traditionally
result = []
for character in 'text':
    reusult.append(character)
print(result)
# ['t', 'e', 'x', 't']

# With list comprehension
result = [character for character in 'text']
print(result)
# ['t', 'e', 'x', 't']

```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
# Traditionally
squared_numbers = []
for number in range(0,11):
    squared_numbers.append(number**2)
print(squared_numbers)
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

# With list comprehension
squared_numbers = [number**2 for number in range(0,11)]
print(squared_numbers)
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
# Traditionally
even_numbers = []
for number in range(0,11):
    if number % 2 == 0:
        even_numbers.append(number)
print(even_numbers)
# [0, 2, 4, 6, 8, 10]

# With list comprehension
even_numbers = [number for number in range(0,11) if number % 2 == 0 ]
print(even_numbers)
# [0, 2, 4, 6, 8, 10]

```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
# Traditionally
squared_numbers = []
for number in range(0,11):
    squared_numbers.append(number**2)

squared_even_numbers = []
for squared_number in squared_numbers:
    if squared_number % 2 == 0:
        squared_even_numbers.append(squared_number)
print(squared_even_numbers)
# [0, 4, 16, 36, 64, 100]

# With list comprehension
squared_numbers = [number**2 for number in range(0, 11)]
squared_even_numbers = [squared_number for squared_number in squared_numbers if squared_number % 2 == 0]
print(squared_even_numbers)
# [0, 4, 16, 36, 64, 100]

```

Sintaxis y ortografía

Comprensión de listas

example.py

```python
# With list comprehension
squared_even_numbers = [number**2 for number in range(0, 11) if number**2 % 2 == 0]
print(squared_even_numbers)
# [0, 4, 16, 36, 64, 100]

```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
module_of_7 = set()
for number in [7, 10, 11, 14]:
    module_of_7.add(number % 7)
print(module_of_7)
# {0, 3, 4}

# Set comprehension
module_of_7 = {number % 7 for number in [7, 10, 11, 14]}
print(module_of_7)
# {0, 3, 4}

```

Sintaxis y ortografía

Bucle vs Comprensión

example.py

```python
# Traditionally
is_even_by_number = {}
for number in range(0,6):
    is_even_by_number[number] = number % 2 == 0
print(is_even_by_number)
# {0: True, 1: False, 2: True, 3: False, 4: True, 5: False}

# With list comprehension
is_even_by_number = {number: number % 2 == 0 for number in range(0,6) }
print(is_even_by_number)
# {0: True, 1: False, 2: True, 3: False, 4: True, 5: False
```

Sintaxis y ortografía

Bucle vs Comprensión

Sintaxis y ortografía:
Map y Filter

CodeLab

29/11/2024

Sintaxis y ortografía

Map

Sintaxis y ortografía

Map

example.py

```python
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, numbers))
# squared = [1, 4, 9, 16, 25]
```

Sintaxis y ortografía

Map

example.py

```python
numbers_1 = [1, 7, 5, 9, 6]
numbers_2 = [2, 3, 8, 4, 10]

# Lambda function that takes two arguments
find_max = lambda x, y: x if x > y else y

# map() with a 2 paramater lambda function process 2 lists
max_result = list(map(find_max, numbers_1, numbers_2))
print(max_result)
# [2, 7, 8, 9, 10]

```

Sintaxis y ortografía

Map

example.py

```python
numbers = [7, 10, 11, 14]
module_of_7 = set(map(lambda x: x % 7, numbers))
# module_of_7 = {0, 3, 4}

```

Sintaxis y ortografía

Filter

Sintaxis y ortografía

Filter

example.py

```python
numbers = [-3, -1, 0, 2, 5, -8, 9]
positive_numbers = list(filter(lambda x: x > 0, numbers))
# positive_numbers = [2, 5, 9]

```

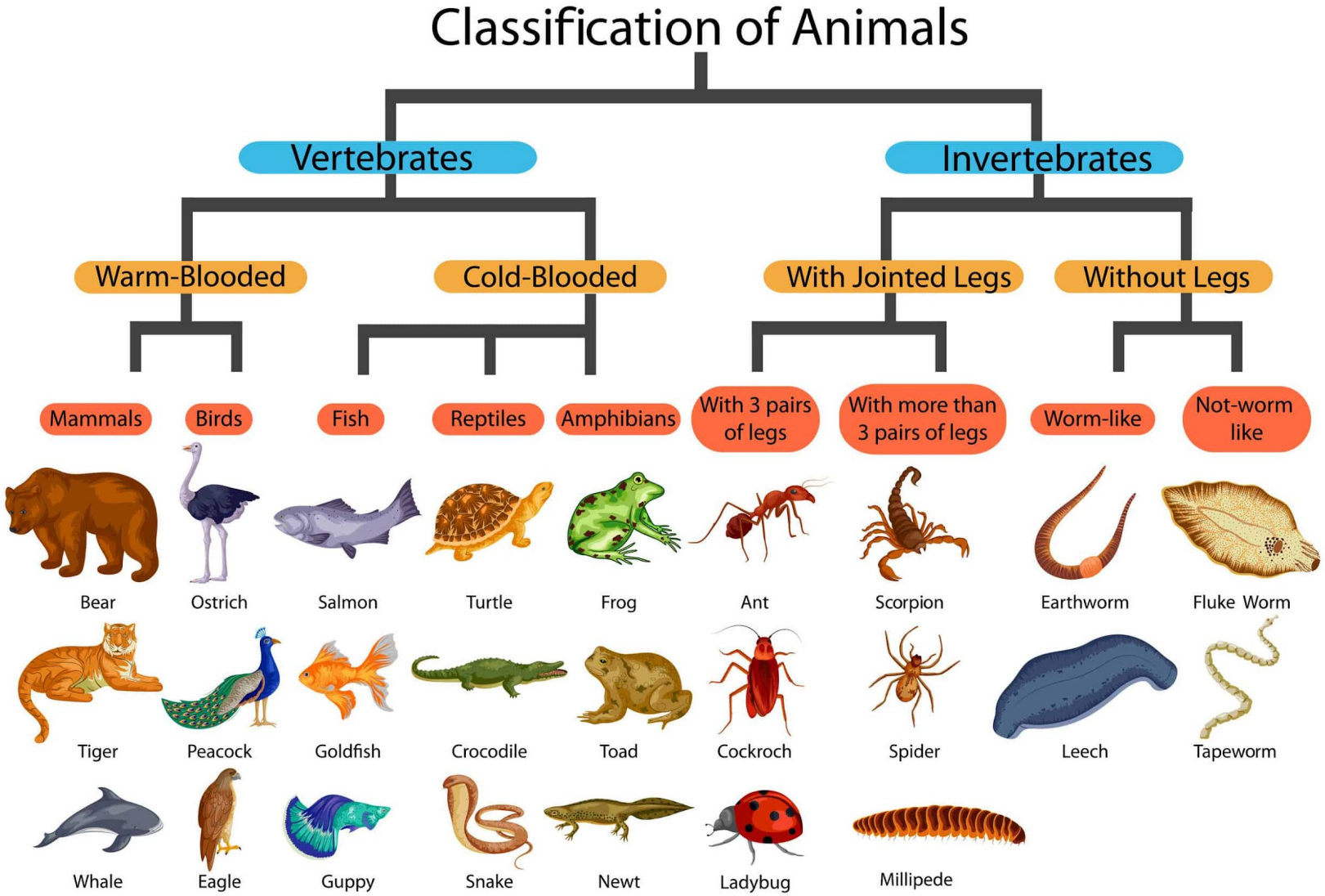
Sintaxis y ortografía:
Herencia de clases

CodeLab

29/11/2024

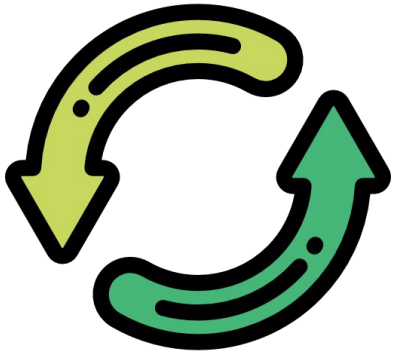
Sintaxis y ortografía

Definición herencia de clases

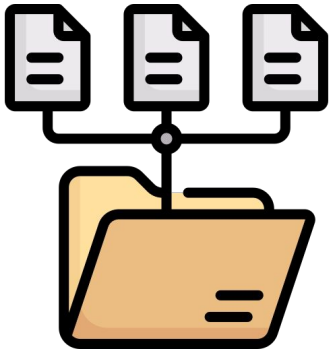


Sintaxis y ortografía

Utilidad herencia de clases

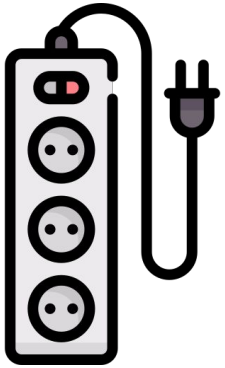


Reutilizar código



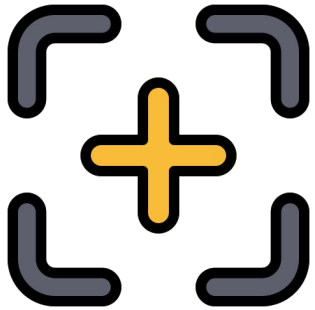
Estructuración

Polimorfismo



Extensibilidad

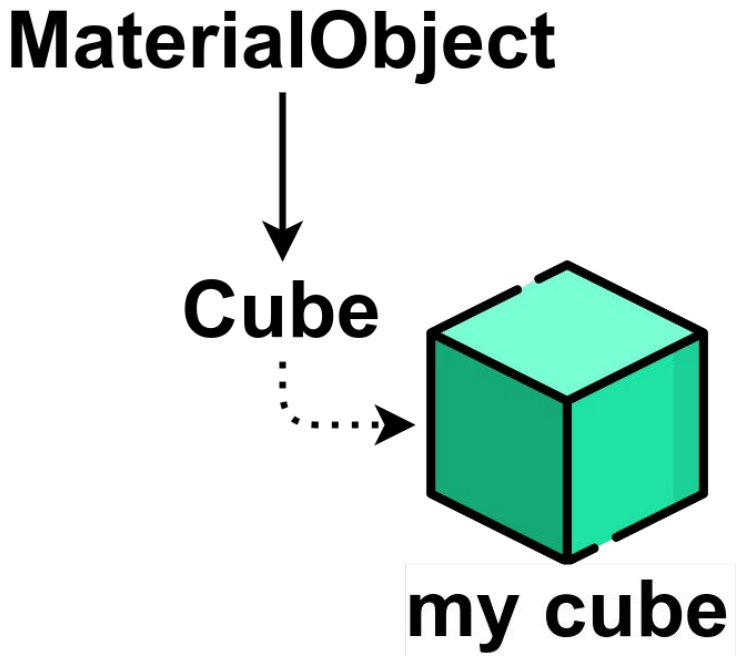
Abstracción



Especialización

Sintaxis y ortografía

Herencia simple

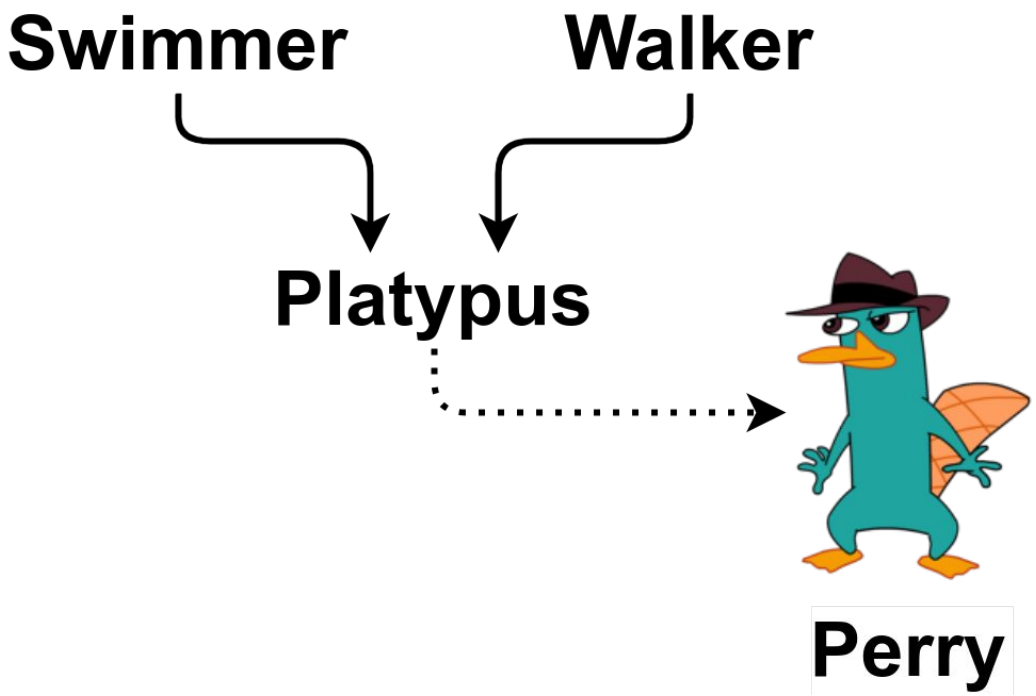


example.py

```python
class MaterialObject:
  def __init__(self, volume, mass):
    self.volume = volume
    self.mass = mass

  def calculate_volume(self):
    return self.mass / self.volume

class Cube(MaterialObject):

  def __init__(self, side, mass):
    self.side = side
    return super().__init__(side**3, mass)

  def calculate_area(self):
    return self.side**2 * 6

my_cube = Cube(2, 10)
print(f"My cube volume is: {my_cube.calculate_volume()}")
# My cube volume is: 1.25
print(f"My cube area is: {my_cube.calculate_area()}")
# My cube area is: 24

```

Sintaxis y ortografía

Herencia múltiple

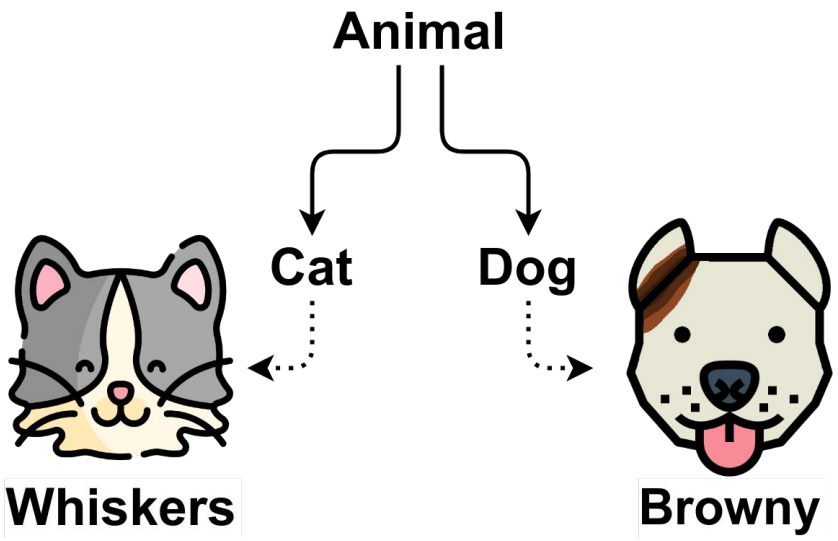


example.py

```python
# Base class 1
class Swimmer:
    def swim(self):
        return "Swimming gracefully."

# Base class 2
class Walker:
    def walk(self):
        return "Walking on land."

# Derived class inheriting from Swimmer and Walker
class Platypus(Swimmer, Walker):
    def __init__(self, name):
        self.name = name

    def description(self):
        return f"A platypus? {self.name} the platypus!"

# Create an instance of the Platypus
perry = Platypus("Perry")

# Use methods from both base classes
print(perry.swim())
# Swimming gracefully.
print(perry.walk())
# Walking on land.

# Use the method of the derived class
print(perry.description())
# A platypus? Perry the platypus!

```

Sintaxis y ortografía

Sobreescritura de métodos mediante heren

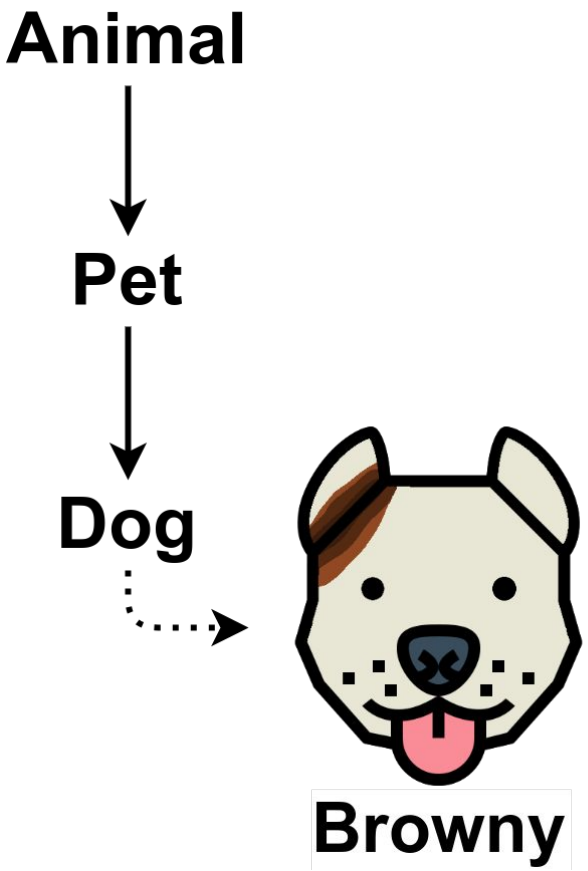


example.py

```python
# Base class
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return f"{self.name} can speak!"

# Derived classes with method overriding
class Dog(Animal):
    def speak(self):
        return f"{self.name} the dog says Woof!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} the cat says Meow!"


# Creating instances of derived classes
dog = Dog("Browny")
cat = Cat("Whiskers")

# Using method overridden in derived classes
print(dog.speak())
# Browny the dog says Woof!
print(cat.speak())
# Whiskers the cat says Meow!

```

Sintaxis y ortografía

Ejemplo herencia



example.py

```python
# Base class
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return f"{self.name} can speak!"

# Intermediate class with shared behavior
class Pet(Animal):
    def __init__(self, name, owner):
        super().__init__(name)
        self.owner = owner

    def introduce(self):
        return f"{self.name} is owned by {self.owner}."

# Derived classes with method overriding and new attributes
class Dog(Pet):
    def __init__(self, name, owner, breed):
        super().__init__(name, owner)
        self.breed = breed

    def speak(self):
        return f"{self.name} the dog ({self.breed}) says Woof!"

# Creating instances of derived classes with new attributes
dog = Dog("Browny", "Carlos", "Pitbull")

# Using methods and accessing new attributes
print(dog.introduce())
# Browny is owned by Carlos
print(dog.speak())
# Browny the dog (Pitbull) says Woof!

```

Sintaxis y ortografía

Clases abstractas

example.py

```python
from abc import ABC, abstractmethod

# Abstract class
class Animal(ABC):
    def __init__(self, name, species):
        self.name = name
        self.species = species

    @abstractmethod
    def speak(self):
        pass

class Dog(Animal):

    def __init__(self, name):
        super().__init__(name, "Dog")

    def speak(self):
        return "Woof!"

# The next line will give a TypeError (Can't instantiate abstract class)
abstract_animal = Animal("Name", "Species")
# Creating instance of the Dog class
browny_boy = Dog("Browny")
```

Sintaxis y ortografía:
Decoradores

CodeLab

29/11/2024

Sintaxis y ortografía

Decoradores

decorators.py

```python
def my_decorator(func):
  def wrapper_my_decorator(*args, **kwargs):
    # Here goes wathever you want
    print("You are using a decorator")
    return func(*args, **kwargs)
  return wrapper_my_decorator

@my_decorator
def hello_world():
  print("Hello world!")

hello_world()
# You are using a decorator
# Hello world!
```

Sintaxis y ortografía

Crisis de identidad con decoradores

decorators.py

```
1 name = hello_world.__name__
2 # name = wrapper_my_decorator
3
4 help(hello_world)
5 # Help on function wrapper_my_decorator in module decorators:
6 # wrapper_my_decorator(*args, **kwargs)
7
```

Sintaxis y ortografía

Resolver crisis de identidad con decoradores

decorators.py

```python
import functools


def my_decorator(func):
    @functools.wraps(func)
    def wrapper_my_decorator(*args, **kwargs):
        # Here goes wathever you want
        print("You are using a decorator")
        return func(*args, **kwargs)

    return wrapper_my_decorator


@my_decorator
def hello_world():
    print("Hello world!")


hello_world()
# You are using a decorator
# Hello world!

name = hello_world.__name__
# name = hello_world

help(hello_world)
# Help on function hello_world in module decorators:
# hello_world()

```

Sintaxis y ortografía

Múltiples decoradores

decorators.py

```python
import functools


def decorator_a(func):
    @functools.wraps(func)
    def inner_a():
        print("Decorator a called")
        return "Decorator a: " + func()
    return inner_a

def decorator_b(func):
    @functools.wraps(func)
    def inner_b():
        print("Decorator b called")
        return "Decorator b: " + func()
    return inner_b

@decorator_a
@decorator_b
def hello_world_with_decorators():
    return "Hello world!"


result = hello_world_with_decorators()
## Decorator a called
## Decorator b called
# result = Decorator a called: Decorator b called: Hello world!
= "Decorator a: " + ("Decorator b: " + ("Hello world!"))

```

Sintaxis y ortografía

Decoradores con parámetros

decorators.py

```python
import functools

def multiply_by(x):
  def inner_multiply_by(func):
    @functools.wraps(func)
    def wrapper_multiply_by(*args, **kwargs):
      return x * func(*args, **kwargs)
    return wrapper_multiply_by
  return inner_multiply_by

@multiply_by(3)
def give_me_a_number():
  return 2

result = give_me_a_number()
# result = 6

```

CodeLab

Sintaxis y ortografía

Ejemplo práctico con decoradores

decorators.py

```python
import functools
import time


# Decorator to measure execution time
def calculate_execution_time(func):
  @functools.wraps(func)
  def wrapper(*args, **kwargs):
    # Record the start time
    start_time = time.time()
    # Execute the original function
    result = func(*args, **kwargs)
    # Record the end time
    end_time = time.time()
    # Calculate the execution time
    execution_time = end_time - start_time
    print(f"The function '{func.__name__}' took {execution_time:.4f} seconds to execute.")
    return result

  return wrapper

```

main.py

```python
import math

from decorators import calculate_execution_time

# Gets n first prime numbers
@calculate_execution_time
def get_prime_numbers(n: int):
  prime_numbers = [2, 3]

  def evaluate_number(number):
    for prime_number in prime_numbers:
      if prime_number * 3 > number:
        prime_numbers.append(number)
        break
      if number % prime_number == 0:
        break

  def uneven_numbers(min_uneven=0, max_uneven=math.inf):
    if min_uneven % 2 == 0:
      min_uneven += 1

    i = min_uneven
    while i < max_uneven:
      yield i
      i += 2

  for i in uneven_numbers(5):
    evaluate_number(i)
    if len(prime_numbers) >= n:
      break
  return prime_numbers


get_prime_numbers(10000)
# The function 'get_prime_numbers' took 0.8821 seconds to execute.

```

Sintaxis y ortografía:
Gestores de contexto

CodeLab

29/11/2024

Sintaxis y ortografía

Gestores de contexto

example.py

```
1 file = open('some_file', 'r')
2 text_in_file = file.read()
3 print(text_in_file)
4 file.close()
5
```

example.py

```
1 with open('some_file', 'r') as file:
2   text_in_file = file.read()
3   print(text_in_file)
4
```

Sintaxis y ortografía

Modos de abrir un fichero

Modo	Descripción
'r'	Modo de lectura (read): Abre el archivo para lectura. Si el archivo no existe, se genera un error.
'w'	Modo de escritura (write): Abre el archivo para escritura. Si el archivo ya existe, trunca su contenido; si no existe, crea uno nuevo.
'a'	Modo de agregado (append): Abre el archivo para escritura, pero agrega datos al final sin truncar. Si el archivo no existe, se crea uno nuevo.
'r+'	Modo de lectura y escritura (read-write): Abre el archivo para lectura y escritura. Si el archivo no existe, se genera un error.
'w+'	Modo de escritura y lectura (write-read): Abre el archivo para escritura y lectura. Si el archivo ya existe, trunca su contenido. Si no existe, crea uno nuevo.
'a+'	Modo de agregado y lectura (append-read): Abre el archivo para lectura y escritura. Si el archivo no existe, se crea uno nuevo. Permite agregar datos al final.
'x'	Modo de creación exclusiva (exclusive creation): Abre el archivo para escritura, generando un error si ya existe. Útil para crear nuevos archivos de forma segura.
'rb'	Modo binario de lectura (binary read): Abre el archivo en modo binario para lectura. Utilizado para leer archivos binarios como imágenes.
'wb'	Modo binario de escritura (binary write): Abre el archivo en modo binario para escritura. Utilizado para escribir archivos binarios.

Sintaxis y ortografía

Concurrencia con ThreadPool

example.py

```python
from concurrent.futures import ThreadPoolExecutor, as_completed

CONCURRENCY = 4


def calculate_square(number):
    return number ** 2


tasks = []
with ThreadPoolExecutor(CONCURRENCY) as executor:
    for number in range(10):
        tasks.append(executor.submit(calculate_square, number))
    for task in as_completed(tasks):
        result = task.result()
        print(result)
```

Sintaxis y ortografía

Gestores de contexto

example.py

```python
# Class definition:
class DatabaseConnection(object):
    def __init__(self, db_name):
        self.db_name = db_name

    def __enter__(self):
        self.db_resource = open_db_resource(self.db_name)

        return self.db_resource

    def __exit__(self, *args):
        self.db_resource.close()



# Use example:
with DatabaseConnection('my_database') as db_connection:
    db_connection.save_entity(my_entity)

```

Sintaxis y ortografía:
Control de excepciones

CodeLab

29/10/2024

Sintaxis y ortografía

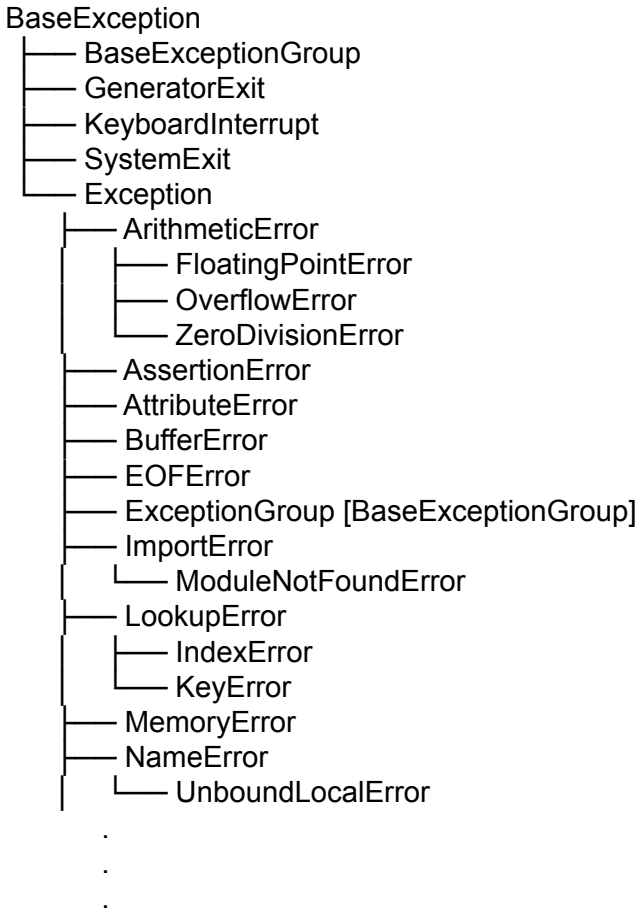
Sintaxis control de excepciones

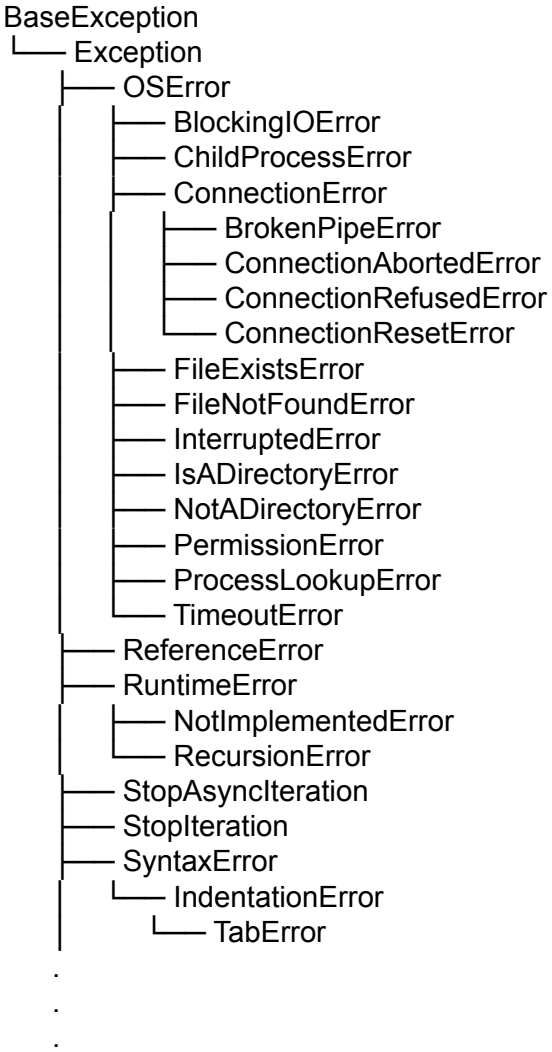
example.py

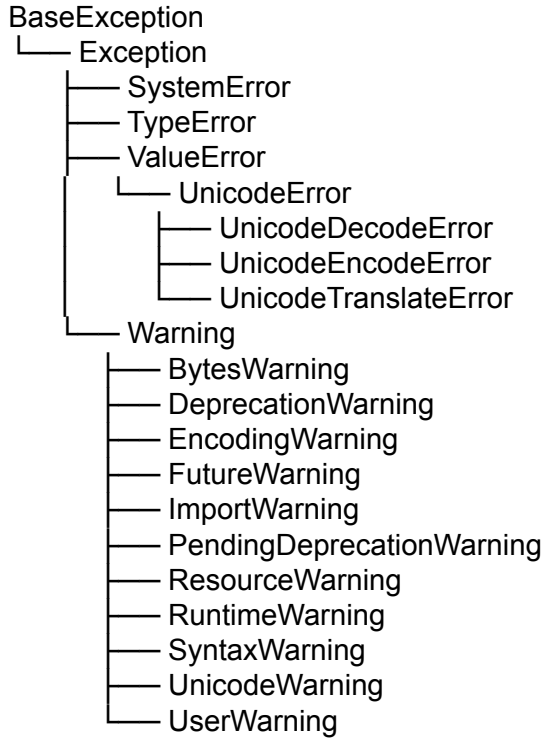
```python
try:
    # Try to execute code that may raise an exception
    number = int(input("Enter a number: "))
    result = 10 / number
    print("The result is:", result)

except ZeroDivisionError:
    # Catch the ZeroDivisionError exception if it occurs
    print("Error: You can't divide by zero.")

except ValueError as e:
    # Catch the ValueError exception if it occurs, and display the error message
    print(f"Error: {e}")

except Exception as e:
    # Catch any other unhandled exceptions
    print(f"Unexpected error: {e}")

else:
    # This block executes if no exception is raised
    print("Operation successful.")

finally:
    # This block always executes, regardless of whether there was an exception
    print("End of the program.")

```

Sintaxis y ortografía

Tipos de excepciones







Sintaxis y ortografía

Ejemplo de control de excepciones

example.py

```python
# Custom exception class for Shrek-related errors
class ShrekException(Exception):
    def __init__(self, message="Oops! Something Shrektastic happened."):
        super().__init__(message)

# Function that may raise the custom ShrekException
def is_it_swampy():
    swamp_decision = input("Is it swampy? (yes/no): ").lower().strip()
    if swamp_decision != "yes":
        raise ShrekException("This is not a swamp! Shrek would not approve.")

def shrek_color():
    what_colour = input("What's your favourite colour?: ").lower().strip()
    if what_colour != "green":
        raise ShrekException("Why not green! Shrek would not approve.")

try:
    is_it_swampy()
    shrek_color()
    print("If you can see this, shrek is happy and no exception was raised")
except ShrekException as e:
    print(f"Shrek Exception: {e}")

```

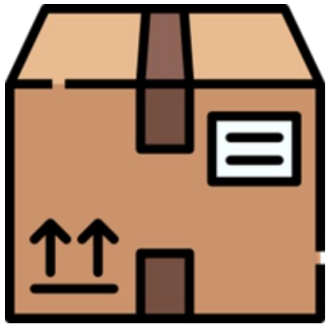
Importación de código:
Módulos, paquetes e importaciones

CodeLab

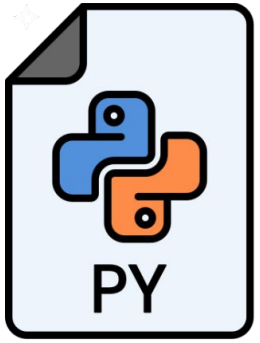
29/11/2024

Importación de código

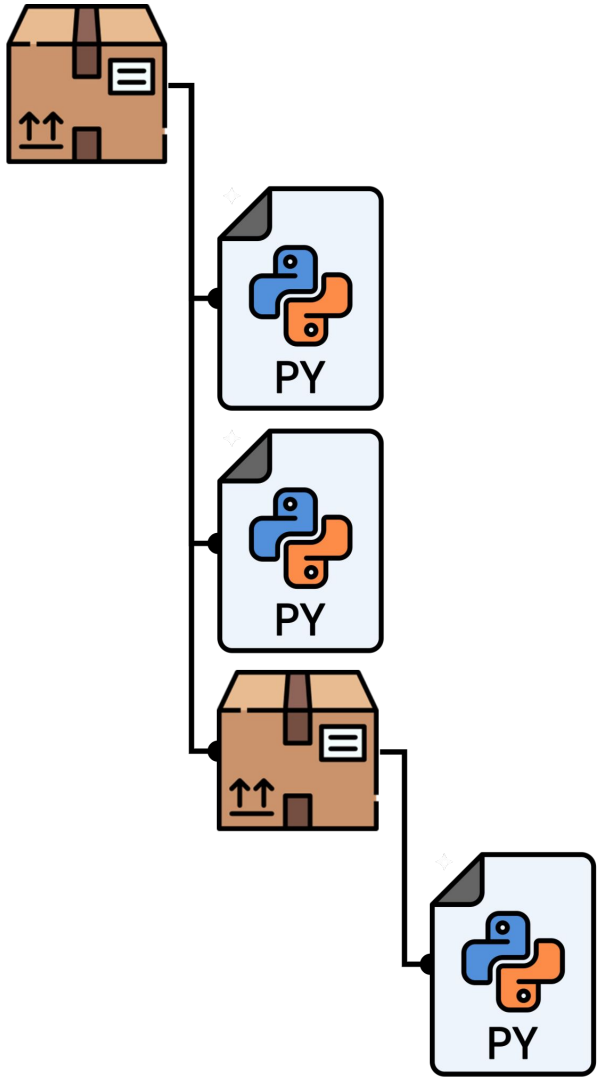
Módulos y paquetes



package



module.py



Organización de paquetes y modulos

Importación de código

Como importar código

example.py

```python
import module
import package.module
import package.subpackage.module
import module_1, module_2
import module as custom_name

from module import object
from package.module import object
from module import object_1, object_2

```

Importación de código

Módulos y paquetes

```
animals/
    • animal.py
    • dog.py
    • pet.py
main.py
```

animal.py

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

```

Importación de código

Módulos y paquetes

```
animals/
    • animal.py
    • dog.py
    • pet.py
main.py
```

pet.py

```python
from .animal import Animal


class Pet(Animal):
    def __init__(self, name, owner):
        super().__init__(name)
        self.owner = owner

    def introduce(self):
        return f"{self.name} is owned by {self.owner}."

```

Importación de código

Módulos y paquetes

```
animals/
    • animal.py
    • dog.py
    • pet.py
main.py
```

dog.py

```python
from .pet import Pet


class Dog(Pet):
    def __init__(self, name, owner, breed):
        super().__init__(name, owner)
        self.breed = breed

    def speak(self):
        return f"{self.name} the dog ({self.breed}) says Woof!"

```

Importación de código

Módulos y paquetes

```
animals/
    • animal.py
    • dog.py
    • pet.py
main.py
```

main.py

```python
from animals.dog import Dog


# Creating instances of derived classes with new attributes
dog = Dog("Browny", "Carlos", "Pitbull")

# Using methods and accessing new attributes
print(dog.introduce())
# Browny is owned by Carlos.
print(dog.speak())
# Browny the dog (Pitbull) says Woof!

```

Importación de código:
Librerías externas

CodeLab

29/10/2024

Importación de código

Python Package Index (PyPI)

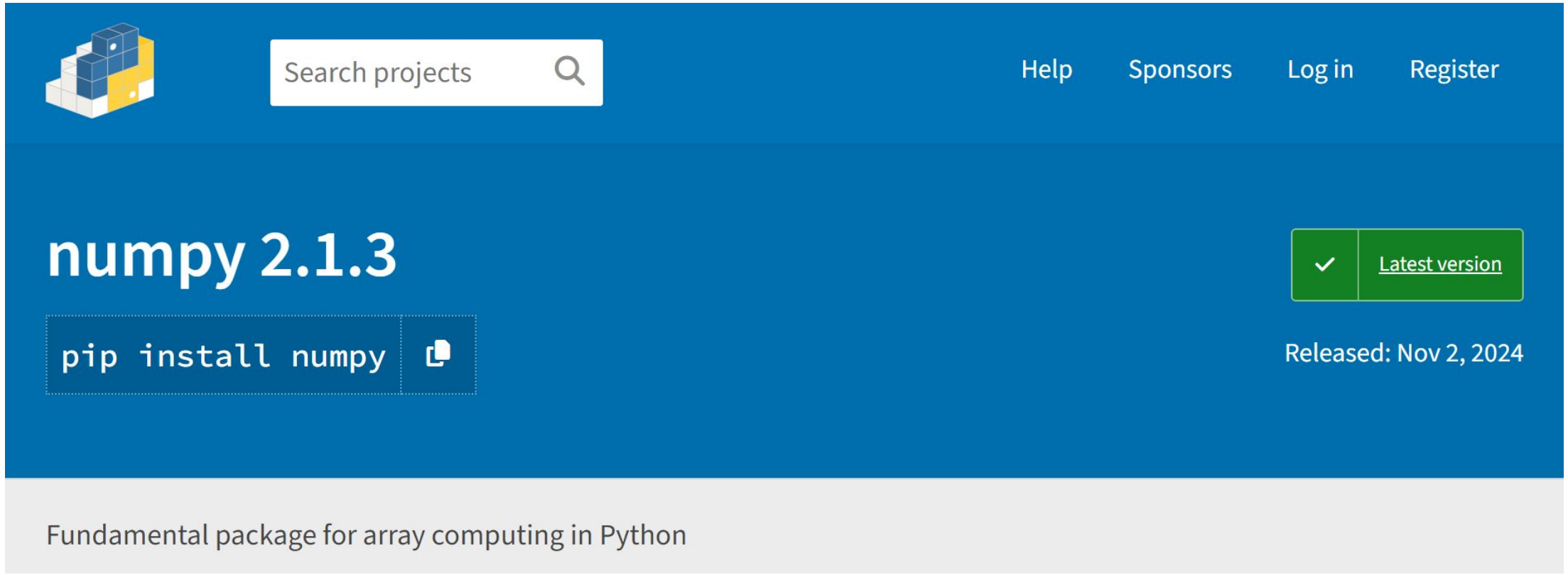
Python Package Index (PyPI)

Importación de código

Instalación de dependencias

Importación de código

Instalación de dependencias



Importación de código

Instalación de dependencias

terminal

```
pip install numpy
```

example.py

```
1 import numpy as np
2
3 a = np.array([[1, 2, 3], [4, 5, 6]])
```

Librerías externas

Gestionar dependencias – requirements.txt

requirements.txt

```
# Framework
fastapi==0.115.0
uvicorn==0.30.6

# Database connection
pymongo==4.9.1

# Operation
numpy==2.1.1

# Testing
pytest==8.3.3

```

Para instalar las dependencias:

terminal

```
pip install -r requirements.txt
```

Importación de código

Gestionar dependencias – pipenv

Para instalar las dependencias:

terminal

```
pipenv install fastapi==0.115.0
```

Pipfile

```
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
fastapi = "==0.115.0"
uvicorn = "==0.30.6"
pymongo = "==4.9.1"
numpy = "==2.1.1"

[dev-packages]
pytest = "==8.3.3"

[requires]
python_version = "3.13"
python_full_version = "3.13.0"

```

Entresijos:
Referencia o valor

CodeLab

29/10/2024

Sintaxis y ortografía

Los argumentos por referencia o por valor

Sintaxis y ortografía

Los argumentos por referencia o por valor

example.py

```python
def modify_number(num):
    num += 10
    return num

x = 5
result = modify_number(x)
# x = 5 (unchanged)
# result = 15 (modified and returned inside the function)

```

Sintaxis y ortografía

Los argumentos por referencia o por valor

example.py

```python
def modify_list(my_list):
    my_list.append(4)
    return my_list

my_list = [1, 2, 3]
result = modify_list(my_list)
# my_list = [1, 2, 3, 4] (modified inside the function)
# result = [1, 2, 3, 4] (same list)

```

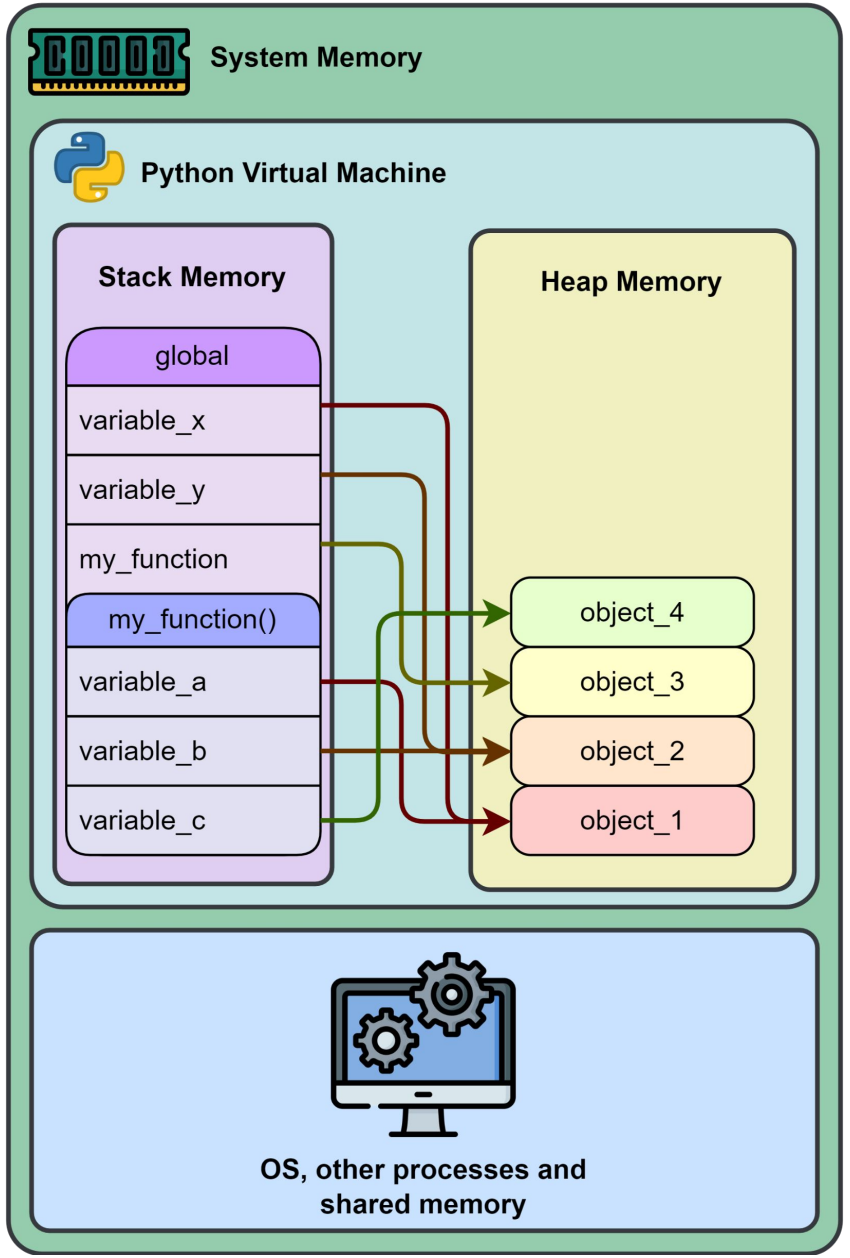
Sintaxis y ortografía

Los argumentos por referencia o por valor

No es lo mismo reasignar que modificar…

example.py

```
1 # Reassign
2 variable_a = "red"
3
4 variable_a = "blue"
5
6
7 # Modify
8 variable_b = ["eggs"]
9
10 variable_b.append("potatoes")
```



Entresijos:
Funcionamiento de los import

CodeLab

29/10/2024

Entresijos

Funcionamiento de import

module_a.py

```python
import module_b


def function_a():
    # Do some stuff...
    print("function_a called")
    module_b.function_b()


function_a()

```

module_b.py

```python
def function_b():
    # Do some stuff...
    print("function_b called")


function_b()

```

terminal

```
$ python module_a.py
function_b called
function_a called
function_b called
```

Entresijos

Funcionamiento de import

module_a.py

```python
from module_b import function_b


def function_a():
    # Do some stuff...
    print("function_a called")
    function_b()


function_a()

```

module_b.py

```python
def function_b():
    # Do some stuff...
    print("function_b called")


function_b()

```

terminal

```
$ python module_a.py
function_b called
function_a called
function_b called
```

Entresijos

if __name__ == '__main__':

module_a.py

```python
import module_b


def function_a():
    # Do some stuff
    print("function_a called")
    module_b.function_b()


if __name__ == '__main__':
    function_a()

```

module_b.py

```python
def function_b():
    # Do some stuff
    print("function_b called")


if __name__ == '__main__':
    function_b()

```

terminal

```
$ python module_a.py
function_a called
function_b called
```

CodeLab

¡Muchas gracias!

CodeLab

29/10/2024