Python

Basic	Python là gì? Liệt kê một số lợi ích của Python. Python là một ngôn ngữ lập trình bậc cao, thông dịch và đa mục đích. Nó nổi tiếng với cú pháp đơn giản và dễ đọc, giúp việc học và sử dụng trở nên dễ dàng. Một số lợi ích của Python bao gồm: - Dễ học và dễ đọc – cú pháp rõ ràng, gần giống ngôn ngữ tự nhiên. - Đa năng và chạy trên nhiều nền tảng – Windows, macOS, Linux, v.v. - Thư viện chuẩn và hệ sinh thái lớn – nhiều module tích hợp sẵn và gói bên thứ ba. - Hỗ trợ nhiều mô hình lập trình – lập trình thủ tục, hướng đối tượng, và hàm. - Cộng đồng mạnh mẽ – nhiều hướng dẫn, diễn đàn và tài nguyên học tập.	What is Python? Enlist some of its benefits. Python is a high-level, interpreted, and general-purpose programming language. It is known for its simple and readable syntax, which makes it easy to learn and use. Some benefits of Python include: - Easy to learn and read – clean syntax that resembles English. - Versatile and cross-platform – works on Windows, macOS, Linux, etc. - Large standard library and ecosystem – many built-in modules and third-party packages. - Supports multiple programming paradigms – procedural, object-oriented, and functional programming. - Strong community support – many tutorials, forums, and resources available.	
	Bạn có thể cho biết Python thuộc lập trình hướng đối tượng hay lập trình hàm không? Python là một ngôn ngữ lập trình đa mô hình, nghĩa là nó hỗ trợ cả lập trình hướng đối tượng (OOP) và lập trình hàm. - Lập trình hướng đối tượng: Python cho phép bạn định nghĩa class, tạo object, và sử dụng kế thừa, đa hình. - Lập trình hàm: Python hỗ trợ hàm là đối tượng cấp 1, hàm bậc cao, map, filter, reduce, và biểu thức lambda. Vì vậy, Python vừa hướng đối tượng vừa hỗ trợ lập trình hàm, tùy cách bạn viết mã.	Can you tell us if Python is object-oriented or functional programming? Python is a multi-paradigm programming language, which means it supports both object-oriented programming (OOP) and functional programming. - Object-oriented programming: Python allows you to define classes, create objects, and use inheritance and polymorphism. - Functional programming: Python supports functions as first-class objects, higher-order functions, map, filter, reduce, and lambda expressions. So, Python is both object-oriented and functional, depending on how you write your code.	
	Quy tắc nào điều khiển biến cục bộ (local) và biến toàn cục (global) trong Python? Trong Python, phạm vi biến quyết định nơi một biến có thể được truy cập: 1. Biến cục bộ (local) được định nghĩa trong một hàm và chỉ có thể sử dụng bên trong hàm đó. Chúng được tạo khi hàm bắt đầu và bị hủy khi hàm kết thúc. 2. Biến toàn cục (global) được định nghĩa bên ngoài hàm và có thể truy cập ở bất kỳ đâu trong file. 3. Để thay đổi biến toàn cục trong hàm, bạn phải dùng từ khóa global.	What rules govern local and global variables in Python? In Python, variable scope determines where a variable can be accessed: 1. Local variables are defined inside a function and can only be accessed within that function. They are created when the function starts and destroyed when it ends. 2. Global variables are defined outside any function and can be accessed anywhere in the file. 3. To modify a global variable inside a function, you must use the global keyword.	python Copy code `x = 10  # global variable` `def my_function():` `  y = 5  # local variable` `  global x` `  x += 1` `  print("Inside function:", x, y)` `my_function()` `print("Outside function:", x)` **Output:** bash Copy code `Inside function: 11 5` `Outside function: 11`

	Bạn có thể giải thích slicing trong Python là gì không? Slicing trong Python là cách lấy một phần của một chuỗi, list hoặc tuple bằng cách chỉ định vị trí bắt đầu, vị trí kết thúc và bước nhảy (tuỳ chọn).	Can you tell us what is slicing in Python? Slicing in Python is a way to extract a portion of a sequence, like a list, string, or tuple, by specifying a start index, stop index, and optional step.	python Copy code `my_list = [0, 1, 2, 3, 4, 5]` `print(my_list[1:5:2])  # Output: [1, 3]` `print(my_list[:3])  # Output: [0, 1, 2]` `print(my_list[3:])  # Output: [3, 4, 5]`
	Namespace trong Python là gì? Namespace trong Python là một vùng chứa các tên (biến, hàm, class) và ánh xạ chúng tới các đối tượng. Nói cách khác, namespace giúp Python quản lý tất cả các tên biến và đối tượng tương ứng trong bộ nhớ. Các loại namespace: 1. Local namespace – bên trong một hàm. 2. Global namespace – ở mức module. 3. Built-in namespace – các hàm và exception có sẵn của Python.	What is namespace in Python? A namespace in Python is a container that holds names (identifiers) and maps them to objects. In other words, it is a system that keeps track of all the variable names and their corresponding objects in memory. Types of namespaces: 1. Local namespace – inside a function. 2. Global namespace – at the module level. 3. Built-in namespace – Python's built-in functions and exceptions.	python Copy code `x = 10  # global namespace` `def my_func():` `  y = 5  # local namespace` `  print(y)` `my_func()` `print(x)`
	"pass" trong Python dùng để làm gì? Trong Python, pass là một câu lệnh rỗng, không thực hiện bất kỳ hành động nào. Nó được dùng như chỗ giữ chỗ khi cú pháp yêu cầu một câu lệnh nhưng bạn chưa muốn thực hiện gì. "pass" giúp code chạy bình thường mà không báo lỗi, ngay cả khi hàm hoặc lớp chưa có nội dung.	What is pass in Python? In Python, pass is a null statement that does nothing. It is used as a placeholder when a statement is syntactically required but no action is needed. "pass" allows your code to run without errors even if the function or class has no content yet.	python Copy code `def my_function():` `  pass  # To be implemented later` `class MyClass:` `  pass  # Empty class`
	Bạn có thể giải thích unittest trong Python là gì không? "unittest" trong Python là một framework kiểm thử tích hợp sẵn, dùng để viết và chạy unit test — tức là kiểm tra từng phần nhỏ của chương trình (như hàm hoặc class) xem có hoạt động đúng không. Nó cung cấp các tính năng như test case, hàm setup/teardown, và các phương thức assert để kiểm tra kết quả mong đợi. *"unittest là framework có sẵn của Python giúp kiểm tra từng phần nhỏ của chương trình hoạt động đúng hay không."*	Can you explain what is unittest in Python? "unittest" in Python is a built-in testing framework used to write and run unit tests, which help verify that individual parts of your code (like functions or classes) work correctly. It provides features like test cases, setup/teardown methods, and assertions to check expected outcomes. *"unittest is Python's built-in framework for writing and running unit tests to ensure that individual parts of the code work correctly."*	python Copy code `import unittest` `def add(a, b):` `  return a + b` `class TestAdd(unittest.TestCase):` `  def test_addition(self):` `  self.assertEqual(add(2, 3), 5)` `if __name__ == "__main__":` `  unittest.main()`
	Chỉ số âm (negative indexes) trong Python là gì? Chỉ số âm (negative indexes) trong Python được dùng để truy cập các phần tử từ cuối của chuỗi, list hoặc tuple, thay vì từ đầu. Chỉ số -1 là phần tử cuối cùng, Chỉ số -2 là phần tử kế cuối, v.v. Tính năng này giúp truy cập phần tử cuối dễ dàng mà không cần biết độ dài danh sách.	What are negative indexes in Python? Negative indexes in Python are used to access elements from the end of a sequence (like a list or string) instead of the beginning. Index -1 refers to the last element, Index -2 refers to the second last, and so on. This feature makes it easy to work with data from the end of a sequence without knowing its length.	python Copy code `my_list = [10, 20, 30, 40, 50]` `print(my_list[-1])  # Kết quả: 50 (phần tử cuối)` `print(my_list[-2])  # Kết quả: 40 (phần tử kế cuối)`

	What are ODBC modules in Python? ODBC (Open Database Connectivity) modules in Python are libraries that allow Python programs to connect and interact with databases using the ODBC standard. They provide a way to execute SQL queries, fetch data, and manage transactions from Python. Common Python ODBC modules: 1. pyodbc – widely used for connecting to SQL Server, MySQL, Oracle, etc. 2. pypyodbc – pure Python alternative to pyodbc.	Các module ODBC trong Python là gì? Các module ODBC trong Python là các thư viện cho phép chương trình Python kết nối và tương tác với cơ sở dữ liệu theo chuẩn ODBC. Chúng giúp thực thi truy vấn SQL, lấy dữ liệu và quản lý giao dịch từ Python. Các module ODBC phổ biến: 1. pyodbc – dùng rộng rãi để kết nối SQL Server, MySQL, Oracle, v.v. 2. pypyodbc – phiên bản thuần Python thay thế cho pyodbc.	python Copy code `import pyodbc` `conn = pyodbc.connect(` `"DRIVER={SQL Server};SERVER=localhost;DATABASE=TestDB;UID=user;PWD=passwo` `)` `cursor = conn.cursor()` `cursor.execute("SELECT * FROM Employees")` `for row in cursor.fetchall():` `print(row)` `conn.close()`
	Bạn sẽ gửi email từ một script Python như thế nào? Bạn có thể gửi email từ Python bằng module smtplib, cho phép kết nối tới SMTP server và gửi email. Thường kết hợp với module email để định dạng nội dung.	How will you send an email from a Python Script? You can send an email from a Python script using the smtplib module, which allows you to connect to an SMTP server and send messages. Often, you also use the email module to format the message.	python Copy code `import smtplib` `from email.mime.text import MIMEText` `# Email content` `msg = MIMEText("Hello, this is a test email from Python.")` `msg['Subject'] = "Test Email"` `msg['From'] = "your_email@example.com"` `msg['To'] = "recipient@example.com"` `# Connect to SMTP server and send` `with smtplib.SMTP('smtp.example.com', 587) as server:` `server.starttls()  # Secure the connection` `server.login('your_email@example.com', 'password')` `server.send_message(msg)` `print("Email sent successfully!")`
	PEP 8 là gì và tầm quan trọng của nó là gì? PEP 8 là viết tắt của Python Enhancement Proposal 8. Nó là hướng dẫn chuẩn về phong cách viết code Python, quy định về cách đặt tên, căn lề, thụt lề, và nhiều yếu tố khác. Tầm quan trọng: 1. Giúp code dễ đọc và đồng nhất trên các dự án. 2. Hỗ trợ làm việc nhóm nhờ tuân theo cùng một phong cách. 3. Tăng khả năng bảo trì và giảm lỗi. Việc thụt lề, đặt tên và khoảng trắng đúng chuẩn giúp code dễ đọc hơn.	What is PEP 8 and its importance? PEP 8 stands for Python Enhancement Proposal 8. It is the style guide for writing Python code, providing conventions for code layout, naming, indentation, and more. Importance: 1. Makes code readable and consistent across projects. 2. Helps team collaboration by following a common coding style. 3. Improves maintainability and reduces errors. Using proper indentation, naming, and spacing makes code easier to read.	

	Những đặc điểm nổi bật (key features) của Python là gì? Python có nhiều đặc điểm nổi bật: 1. Dễ học và dễ đọc – cú pháp đơn giản, gần gũi với ngôn ngữ tự nhiên. 2. Ngôn ngữ thông dịch – chạy code từng dòng. 3. Kiểu dữ liệu động – không cần khai báo kiểu biến. 4. Chạy đa nền tảng – Windows, macOS, Linux, v.v. 5. Thư viện chuẩn phong phú – nhiều module hỗ trợ các công việc khác nhau. 6. Hỗ trợ OOP và lập trình hàm – đa mô hình lập trình. 7. Mã nguồn mở – miễn phí, cộng đồng hỗ trợ mạnh.	What are the key features of Python? Python has several key features: 1. Easy to learn and readable – simple syntax, close to English. 2. Interpreted language – runs code line by line. 3. Dynamically typed – no need to declare variable types. 4. Cross-platform – runs on Windows, macOS, Linux, etc. 5. Extensive standard library – many modules for different tasks. 6. Object-oriented and functional – supports multiple programming paradigms. 7. Open-source – free to use with strong community support.	
	Việc Python là ngôn ngữ dynamically typed có nghĩa là gì? Python là dynamically typed có nghĩa là bạn không cần khai báo kiểu dữ liệu của biến. Trình thông dịch sẽ tự xác định kiểu tại thời gian chạy. Bạn cũng có thể thay đổi kiểu dữ liệu của biến trong quá trình thực thi.	What does it mean to be dynamically typed in Python? Being dynamically typed in Python means that you don't need to declare the type of a variable explicitly. The interpreter automatically detects the type at runtime. You can also change the type of a variable during execution.	python Copy code `x = 10 # x is an integer` `x = "Hello" # now x is a string`
	Scope trong Python là gì? Trong Python, scope (phạm vi biến) là khu vực trong code mà một biến có thể được truy cập. Nó quyết định biến nào có thể dùng ở đâu và biến nào được tham chiếu nếu có nhiều biến cùng tên. Các loại scope: 1. Local scope – bên trong hàm. 2. Enclosing scope – trong hàm lồng nhau. 3. Global scope – ở mức module. 4. Built-in scope – các tên và hàm có sẵn của Python.	What is a scope in Python? In Python, scope refers to the region of the code where a variable is accessible. It determines where a variable can be used and which variable a name refers to if multiple variables have the same name. Types of scope: 1. Local scope – inside a function. 2. Enclosing scope – in nested functions. 3. Global scope – at the module level. 4. Built-in scope – Python's built-in names and functions.	python Copy code `x = 10 # global scope` `def outer():` `  y = 5 # enclosing scope` `  def inner():` `  z = 2 # local scope` `  print(x, y, z)` `  inner()` `outer()`
	Làm thế nào để một script Python có thể thực thi (executable) trên hệ thống Unix? Để một script Python có thể thực thi trên Unix: 1. Thêm shebang ở đầu file để chỉ định trình thông dịch Python: #!/usr/bin/env python3 2. Cấp quyền thực thi cho file bằng lệnh chmod: chmod +x script.py 3. Chạy script trực tiếp từ terminal: ./script.py Điều này cho phép script chạy như một chương trình độc lập.	How can Python script be executable on Unix? To make a Python script executable on Unix: 1. Add the shebang line at the top of the script to specify the Python interpreter: #!/usr/bin/env python3 2. Make the script executable using the chmod command: chmod +x script.py 3. Run the script directly from the terminal: ./script.py This allows the script to be executed like a standalone program.	

	Docstring trong Python là gì? Docstring trong Python là một chuỗi đặc biệt dùng để ghi chú (tài liệu) cho module, class hoặc function. Nó được viết bên trong dấu ba nháy (""" hoặc ''') ở đầu khối code và có thể truy cập qua thuộc tính __doc__.	What is Docstring in Python? A docstring in Python is a special string literal used to document a module, class, or function. It is written inside triple quotes (""" or ''') at the beginning of the code block and can be accessed via the __doc__ attribute.	python Copy code `def add(a, b):` `  """This function returns the sum of two numbers."""` `  return a + b` `print(add.__doc__)` **Output:** bash Copy code `This function returns the sum of two numbers.`
	__init__ trong Python là gì? Trong Python, __init__ là phương thức đặc biệt (constructor) của một class, được gọi tự động khi một đối tượng của class được tạo. Nó dùng để khởi tạo các thuộc tính của đối tượng.	What is init in Python? In Python, __init__ is a special method (constructor) of a class that is automatically called when an object of the class is created. It is used to initialize the attributes of the object.	python Copy code `class Person:` `  def __init__(self, name, age):` `  self.name = name` `  self.age = age` `p = Person("Alice", 25)` `print(p.name, p.age)  # Output: Alice 25`
	List và Tuple trong Python là gì? Trong Python: - List (danh sách) là tập hợp có thứ tự và có thể thay đổi. Bạn có thể sửa, thêm hoặc xóa phần tử. List được tạo bằng dấu ngoặc vuông []. - Tuple (bộ giá trị) là tập hợp có thứ tự nhưng không thể thay đổi. Khi tạo xong, các phần tử không thể chỉnh sửa. Tuple được tạo bằng dấu ngoặc tròn ().	What are lists and tuples in Python? In Python: - List is an ordered, mutable collection of items. You can change, add, or remove elements. Lists are defined using square brackets []. - Tuple is an ordered, immutable collection of items. Once created, its elements cannot be changed. Tuples are defined using parentheses ().	python Copy code `# List` `my_list = [1, 2, 3]` `my_list[0] = 10  # Lists are mutable` `my_list.append(4)` `# Tuple` `my_tuple = (1, 2, 3)` `# my_tuple[0] = 10  # This will raise an error, tuples are immutable` `print(my_list)  # Output: [10, 2, 3, 4]` `print(my_tuple)  # Output: (1, 2, 3)`
	Sự khác nhau giữa Array và List trong Python là gì? Trong Python, list có thể chứa các phần tử khác kiểu, rất linh hoạt và có sẵn, trong khi array (từ module array) chỉ chứa các phần tử cùng kiểu, ít linh hoạt hơn nhưng tiết kiệm bộ nhớ và nhanh hơn khi xử lý dữ liệu số lớn.	What is the difference between Arrays and lists in Python? In Python, lists can store elements of different data types, are very flexible, and are built-in, while arrays (from the array module) can store elements of only one data type, are less flexible, but are more memory efficient and faster for large numeric data.	python Copy code `# List` `my_list = [1, "hello", 3.5]` `# Array` `import array` `my_array = array.array('i', [1, 2, 3])  # 'i' means integer type`

	Từ khóa self trong Python được dùng để làm gì? Trong Python, self là tham chiếu tới chính đối tượng hiện tại của lớp. Nó được dùng để truy cập thuộc tính và phương thức của đối tượng từ bên trong lớp. Mỗi phương thức của đối tượng phải có self làm tham số đầu tiên.	What is Self-used for in Python? In Python, self is a reference to the current instance of a class. It is used to access the attributes and methods of the object from within the class. Every instance method must have self as the first parameter.	python Copy code `class Person:` `  def __init__(self, name):` `  self.name = name` `  def greet(self):` `  print("Hello, my name is", self.name)` `p = Person("Alice")` `p.greet()  # Output: Hello, my name is Alice`
	Hai loại vòng lặp chính (loop statements) trong Python là gì? Hai loại vòng lặp chính trong Python là vòng lặp for và vòng lặp while. - for loop dùng để lặp qua một chuỗi, list, tuple hoặc range. - while loop thực hiện một khối code liên tục khi điều kiện còn đúng.	What are the major two-loop statements in Python? The two major loop statements in Python are for loops and while loops. - for loop is used to iterate over a sequence like a list, tuple, string, or range. - while loop executes a block of code repeatedly as long as a condition is true.	python Copy code `# for loop` `for i in range(3):` `  print(i)` `# while loop` `count = 0` `while count < 3:` `  print(count)` `  count += 1`
	Decorator trong Python là gì? Decorator trong Python là một hàm dùng để thay đổi hoặc mở rộng chức năng của một hàm hoặc phương thức khác mà không cần sửa code gốc. Decorator thường dùng cho logging, kiểm soát truy cập, đo thời gian chạy hoặc caching. Cú pháp áp dụng là @tên_decorator phía trên định nghĩa hàm.	What are Decorators in Python? A decorator in Python is a function that modifies or enhances another function or method without changing its code. Decorators are often used for logging, access control, timing, or caching. They are applied using the @decorator_name syntax above a function definition.	main.py > ... `def my_decorator(func):` `  def wrapper():` `  print("Before function call")` `  func()` `  print("After function call")` `  return wrapper` `@my_decorator` `def say_hello():` `  print("Hello!")` `say_hello()` `# Output` `# Before function call` `# Hello!` `# After function call`
	Những kiểu dữ liệu (built-in types) có sẵn trong Python là gì? Python cung cấp nhiều kiểu dữ liệu có sẵn (built-in types). Các kiểu chính bao gồm: 1. Kiểu số: int, float, complex 2. Kiểu tuần tự (sequence): list, tuple, range 3. Kiểu văn bản: str 4. Kiểu tập hợp (set): set, frozenset 5. Kiểu ánh xạ (mapping): dict 6. Kiểu Boolean: bool 7. Kiểu nhị phân: bytes, bytearray, memoryview Những kiểu này giúp bạn lưu trữ và xử lý dữ liệu theo nhiều dạng khác nhau.	What built-in types are available in Python? Python provides several built-in data types. The main ones include: 1. Numeric types: int, float, complex 2. Sequence types: list, tuple, range 3. Text type: str 4. Set types: set, frozenset 5. Mapping type: dict 6. Boolean type: bool 7. Binary types: bytes, bytearray, memoryview 8. These types allow you to store and manipulate different kinds of data. These types allow you to store and manipulate different kinds of data.	

	Làm thế nào để phân biệt tệp .py và .pyc trong Python? Trong Python: Tệp .py là file mã nguồn, chứa code Python do lập trình viên viết. Tệp .pyc là file bytecode đã biên dịch, được Python tạo tự động khi import tệp .py. Nó chứa phiên bản biên dịch không phụ thuộc nền tảng, giúp chạy chương trình nhanh hơn. "Tệp .py là mã nguồn, còn .pyc là bytecode đã biên dịch do Python tạo ra."	How do you differentiate between .py and .pc files in Python? In Python: A .py file is a source code file containing the Python code written by the programmer. A .pyc file is a compiled bytecode file automatically created by Python when a .py file is imported. It contains a platform-independent, compiled version of the source code, which helps the program run faster. "A .py file is source code, and a .pyc file is the compiled bytecode generated by Python."	python Copy code `# my_module.py  -> mã nguồn của bạn` `import my_module  # Python sẽ tự động tạo my_module.pyc`
	Làm thế nào để tạo một hàm (function) trong Python? Trong Python, bạn tạo một hàm (function) bằng từ khóa def, theo sau là tên hàm, dấu ngoặc cho tham số, và dấu hai chấm. Nội dung hàm được thụt vào. Có thể trả về giá trị bằng câu lệnh return. "Tạo hàm trong Python bằng def, viết code bên trong và có thể dùng return để trả về giá trị."	How do you create a Python function? In Python, you create a function using the def keyword, followed by the function name, parentheses for parameters, and a colon. The function body is indented. You can optionally return a value using the return statement. "You create a function in Python using def, write the code inside it, and optionally use return to return a value."	python Copy code `def add(a, b):` `  return a + b` `result = add(3, 5)` `print(result)  # Kết quả: 8`
	Một hàm trong Python trả về giá trị như thế nào? Trong Python, một hàm trả về giá trị bằng câu lệnh return. Khi hàm gặp return, nó gửi giá trị được chỉ định về cho nơi gọi hàm. Nếu không dùng return, hàm sẽ trả về None mặc định.	How does a function return values in Python? In Python, a function returns a value using the return statement. When the function reaches return, it sends the specified value back to the caller. If no return statement is used, the function returns None by default.	
	Lệnh nào được dùng để xóa (delete) file trong Python? Trong Python, bạn có thể xóa file bằng module os hoặc pathlib. Dùng os.remove(): Dùng pathlib.Path.unlink():	What commands are used to delete Python files? In Python, you can delete files using the os or pathlib modules. Using os.remove(): Using pathlib.Path.unlink():	main.py > ... `# . Using os.remove() :` `import os` `os.remove("example.txt") # deletes the file` `# . Using pathlib.Path.unlink() :` `from pathlib import Path` `file = Path("example.txt")` `file.unlink() # deletes the file`
	Module trong Python là gì? Module trong Python là một file chứa code Python, như hàm, lớp hoặc biến, có thể import và sử dụng trong chương trình khác. Module giúp tổ chức code và tái sử dụng hiệu quả.	What are modules in Python? A module in Python is a file containing Python code, such as functions, classes, or variables, that can be imported and used in other Python programs. Modules help organize code and promote code reusability.	python Copy code `# my_module.py` `def greet(name):` `  print(f"Hello, {name}!")` `# main.py` `import my_module` `my_module.greet("Alice")  # Output: Hello, Alice!`
	Python PATH là gì? Trong Python, Python PATH là danh sách các thư mục mà trình thông dịch sẽ tìm kiếm khi bạn import module hoặc package. Nó được lưu trong biến sys.path. Bạn cũng có thể thêm thư mục của riêng mình vào PATH nếu cần.	What is a Python PATH? In Python, the Python PATH refers to a list of directories that the interpreter searches to find modules and packages when you use import. It is stored in the sys.path variable. You can add your own directories to this path if needed.	python Copy code `import sys` `print(sys.path)  # shows all directories Python searches for modules`

	Package trong Python là gì? Trong Python, package là tập hợp các module được tổ chức trong thư mục. Một package có file đặc biệt __init__.py (có thể rỗng) để Python nhận biết thư mục là package. Package giúp tổ chức các module liên quan thành một cấu trúc duy nhất. "Package là thư mục chứa các module Python và file __init__.py để tổ chức chúng."	What is a Package in Python? In Python, a package is a collection of modules organized in directories. A package contains a special file called __init__.py (which can be empty) to indicate that the directory should be treated as a package. Packages help organize related modules into a single hierarchy. "A package is a directory containing Python modules and an __init__.py file to organize them"	markdown Copy code `my_package/` `__init__.py` `module1.py` `module2.py` Bạn có thể import module từ package: python Copy code `from my_package import module1` `module1.some_function()`
	Làm thế nào để tạo một module trong Python? Trong Python, module đơn giản là một file Python với đuôi .py chứa các hàm, biến hoặc lớp. Bạn có thể import module đó trong file khác để sử dụng. "Module là file Python chứa hàm hoặc lớp có thể import vào file khác."	Create a module in Python. In Python, a module is simply a Python file with a .py extension that contains code such as functions, variables, or classes. You can then import it in another Python file to use its contents. "A module is a Python file with functions or classes that can be imported in another file."	1. Tạo file `my_module.py` với nội dung: python Copy code `# my_module.py` `def greet(name):` `print(f"Xin chào, {name}!")` 2. Sử dụng trong file khác: python Copy code `# main.py` `import my_module` `my_module.greet("Alice")  # Kết quả: Xin chào, Alice!`
	Lambda trong Python là gì? Trong Python, lambda là một hàm vô danh, tức là hàm không có tên. Nó có thể nhận nhiều tham số nhưng chỉ có một biểu thức. Lambda thường dùng cho các phép toán ngắn gọn hoặc khi truyền hàm làm đối số cho các hàm khác như map(), filter() hay sorted(). "Lambda là hàm vô danh trong Python, có một biểu thức và nhận nhiều tham số."	What is lambda in Python? In Python, a lambda is an anonymous function, meaning a function without a name. It can take any number of arguments but can only have one expression. Lambda functions are often used for short, simple operations, usually as arguments to other functions like map(), filter(), or sorted(). "A lambda is an anonymous function in Python that has one expression and can take multiple arguments."	python Copy code `# Lambda function to add two numbers` `add = lambda x, y: x + y` `print(add(3, 5))  # Output: 8` `# Using lambda with map` `numbers = [1, 2, 3]` `squared = list(map(lambda x: x**2, numbers))` `print(squared)  # Output: [1, 4, 9]`

	Làm thế nào để quản lý bộ nhớ (memory management) trong Python? Bộ nhớ trong Python được quản lý tự động thông qua các cơ chế: 1. Đếm tham chiếu (reference counting): Mỗi đối tượng theo dõi số lượng tham chiếu trỏ tới nó. Khi số tham chiếu về 0, bộ nhớ có thể được giải phóng. 2. Garbage collection (thu gom rác): Python có trình thu gom rác để phát hiện và xóa các đối tượng không còn dùng, đặc biệt là các tham chiếu vòng. 3. Cấp phát bộ nhớ động: Các đối tượng được cấp phát từ private heap do trình quản lý bộ nhớ của Python kiểm soát. Trong hầu hết trường hợp, lập trình viên không cần cấp phát hay giải phóng bộ nhớ thủ công.	How memory can be managed in Python? Memory in Python is managed automatically using built-in mechanisms: 1. Reference counting: Each object keeps track of how many references point to it. When the reference count drops to zero, the memory can be freed. 2. Garbage collection: Python has a garbage collector to detect and clean up unused objects, especially those involved in circular references. 3. Dynamic memory allocation: Objects are allocated memory from the private heap managed by Python's memory manager. Developers don't need to manually allocate or free memory in most cases.	
	Keyword trong Python là gì? Trong Python, keyword (từ khóa) là từ được giữ riêng, có ý nghĩa đặc biệt trong ngôn ngữ. Chúng không thể dùng làm tên biến, hàm hay định danh khác. Keywords định nghĩa cú pháp và cấu trúc của Python. Một số keyword: if, else, for, while, def, class, import, return, try, except "Keywords là từ khóa có ý nghĩa đặc biệt trong Python, không dùng làm tên biến hay hàm được."	What are keywords in Python? In Python, keywords are reserved words that have a special meaning in the language. They cannot be used as variable names, function names, or identifiers. Keywords define the syntax and structure of Python. Some Python keywords: if, else, for, while, def, class, import, return, try, except "Keywords are reserved words in Python with special meanings that cannot be used as identifiers."	
	Python có phải là ngôn ngữ phân biệt chữ hoa – chữ thường (case-sensitive) không? Đúng, Python là ngôn ngữ phân biệt chữ hoa – chữ thường (case-sensitive). Điều này có nghĩa là các định danh như tên biến, tên hàm, tên lớp được phân biệt chữ hoa và chữ thường. Ví dụ, myVar và myvar được coi là hai định danh khác nhau. "Đúng, Python phân biệt chữ hoa chữ thường, tên biến và hàm khác nhau nếu khác chữ hoa/chữ thường."	Is Python a case-sensitive language? Yes, Python is a case-sensitive language. This means that identifiers such as variable names, function names, and class names are case-sensitive. For example, myVar and myvar would be considered two different identifiers. "Yes, Python is case-sensitive, so variable and function names are treated differently if their cases differ."	python Copy code `myVar = 10` `myvar = 20` `print(myVar)  # Kết quả: 10` `print(myvar)  # Kết quả: 20`
	Literal trong Python là gì? Trong Python, literal (giá trị hằng) là giá trị cố định được viết trực tiếp trong code. Literal biểu diễn dữ liệu thuộc một kiểu nhất định như số, chuỗi, hoặc giá trị Boolean. Ví dụ về literals: - Số: 10, 3.14 - Chuỗi: "Hello", 'Python' - Boolean: True, False	What are literals in Python? In Python, a literal is a fixed value written directly in the code. Literals represent data of a specific type like numbers, strings, or Boolean values. Examples of literals: - Numeric literals: 10, 3.14 - String literals: "Hello", 'Python' - Boolean literals: True, False	python Copy code `x = 10       # numeric literal` `y = "Hello"  # string literal` `flag = True  # boolean literal`
	Type Conversion trong Python là gì? In Python, type conversion (also called type casting) is the process of converting a value from one data type to another. Python provides built-in functions for type conversion like int(), float(), str(), list(), etc. "Type conversion là chuyển giá trị từ kiểu dữ liệu này sang kiểu khác bằng các hàm như int(), float(), str()."	What is Type Conversion in Python? In Python, type conversion (also called type casting) is the process of converting a value from one data type to another. Python provides built-in functions for type conversion like int(), float(), str(), list(), etc. "Type conversion is changing a value from one data type to another using built-in functions like int(), float(), or str()."	python Copy code `x = 10        # int` `y = float(x)  # convert int to float` `print(y)      # Output: 10.0` `z = str(x)    # convert int to string` `print(z)      # Output: "10"`

	Theo bạn, hàm dir() trong Python được dùng để làm gì? Trong Python, hàm dir() được dùng để liệt kê tất cả thuộc tính và phương thức của một đối tượng, module hoặc lớp. Nó giúp khám phá những gì đối tượng đó chứa. "Hàm dir() hiển thị tất cả thuộc tính và phương thức của một đối tượng, lớp hoặc module."	What do you think is the use of dir () function in Python? In Python, the dir() function is used to list all the attributes and methods of an object, module, or class. It helps to explore what a particular object contains. "The dir() function shows all attributes and methods of an object, class, or module."	python Copy code `x = [1, 2, 3]` `print(dir(x))  # Lists all methods and attributes of a list object`
	Làm thế nào để xóa giá trị khỏi mảng (array) trong Python? Trong Python, nếu dùng module array, bạn có thể xóa giá trị bằng remove() hoặc pop(): - remove(value) xóa giá trị đầu tiên xuất hiện trong mảng. - pop(index) xóa giá trị ở vị trí chỉ định (mặc định là phần tử cuối).	How can you remove values from an array in Python? In Python, if you are using the array module, you can remove values using remove() or pop() methods: - remove(value) removes the first occurrence of the specified value. - pop(index) removes the value at the specified index (default is the last element).	python Copy code `import array` `arr = array.array('i', [1, 2, 3, 2])` `arr.remove(2)     # Xóa giá trị 2 đầu tiên` `print(arr)        # Kết quả: array('i', [1, 3, 2])` `arr.pop(1)        # Xóa phần tử ở chỉ số 1` `print(arr)        # Kết quả: array('i', [1, 2])`
	List trong Python được lưu trữ trong bộ nhớ như thế nào? Trong Python, một list là một danh sách động (dynamic array), có thể chứa các phần tử thuộc bất kỳ kiểu dữ liệu nào. Python không lưu trực tiếp giá trị trong list mà lưu các con trỏ trỏ tới các object trong bộ nhớ. List là một object chứa một mảng các con trỏ, kích thước hiện tại (size) và dung lượng dự trữ (capacity). Khi cập nhật phần tử tại vị trí X, Python chỉ thay đổi con trỏ tại vị trí đó để trỏ tới object mới, các phần tử còn lại không thay đổi. Khi thêm phần tử bằng append, Python sẽ kiểm tra dung lượng còn đủ không: nếu đủ, thêm con trỏ vào mảng; nếu không, Python sẽ cấp phát một mảng mới lớn hơn, copy các con trỏ cũ sang, rồi thêm phần tử mới. Vì vậy, append() trung bình là O(1), nghĩa là thêm phần tử vào cuối list thường mất thời gian hằng số, nhưng thỉnh thoảng phải mở rộng bộ nhớ nên mất O(n) thời gian cho lần append đó.	How is list stored in memory in Python?	
	Tuple trong Python được lưu trữ trong bộ nhớ như thế nào? Tuple trong Python là một danh sách bất biến (immutable sequence), có thể chứa các phần tử thuộc bất kỳ kiểu dữ liệu nào. Python không lưu trực tiếp giá trị trong tuple mà lưu các con trỏ trỏ tới các object trong bộ nhớ. Tuple là một object chứa một mảng các con trỏ, cùng với thông tin về kích thước (size). Vì tuple bất biến, nên sau khi tạo: - Kích thước không thay đổi, và mảng con trỏ cố định. - Khi truy cập phần tử tại vị trí X, Python chỉ đọc con trỏ tại vị trí đó để lấy object tương ứng, các phần tử khác không bị ảnh hưởng. - Không có cơ chế mở rộng như list, nên tuple không thể append, insert hay xóa phần tử. Do cấu trúc cố định và không cần quản lý dung lượng dự trữ, tuple nhẹ hơn list, tiết kiệm bộ nhớ hơn, và truy cập phần tử nhanh hơn. Ngoài ra, nếu tất cả phần tử trong tuple đều immutable, tuple có thể hash được và dùng làm key của dict hoặc element của set.	How is tuple stored in memory in Python?	

	How are list, tuple, set, array, and dictionary stored in memory in Python? List, tuple, set, array và dictionary trong Python được lưu trữ trong bộ nhớ như thế nào?	How are list, tuple, set, array, and dictionary stored in memory in Python? List, tuple, set, array và dictionary trong Python được lưu trữ trong bộ nhớ như thế nào?	
Intermediate	Một hàm trong Python có thể không có câu lệnh return không? Và điều đó có hợp lệ không Đúng, một hàm trong Python có thể không có câu lệnh return và điều này vẫn hợp lệ. Nếu không có return, hàm sẽ tự động trả về None theo mặc định.	Is it possible for a function not to have a return statement and is it valid? Yes, it is possible and valid for a Python function not to have a return statement. If a function does not explicitly return a value, it automatically returns None by default.	python Copy code `def greet(name):` `  print(f"Hello, {name}!")` `result = greet("Alice")` `print(result)  # Output: None`
	Khi nào nên dùng dấu ngoặc kép ba (triple quotes) làm ký tự phân tách trong Python? Trong Python, dấu ngoặc kép ba (''' hoặc """) được dùng khi bạn muốn: 1. Tạo chuỗi nhiều dòng trải dài trên nhiều dòng. 2. Viết docstring để ghi chú cho module, class hoặc hàm.	When should Python use triple quotes as a delimiter? In Python, triple quotes (''' or """) are used as delimiters when you want to: 1. Create multi-line strings that span several lines. 2. Write docstrings to document modules, classes, or functions.	python Copy code `# Multi-line string` `text = """This is` `a string that spans` `multiple lines."""` `# Docstring` `def greet(name):` `  """This function greets the user by name."""` `  print(f"Hello, {name}!")`
	Vai trò chính của phương thức __init__ là gì? Hãy đưa ra một ví dụ minh họa bằng code. Vai trò chính của phương thức __init__ trong Python là khởi tạo các thuộc tính của lớp khi một đối tượng được tạo ra. Nó được gọi tự động khi một instance mới của lớp được khởi tạo.	What is the main role of the init method? Give a code block example. The main role of the __init__ method in Python is to initialize the attributes of a class when an object is created. It is automatically called when a new instance of the class is instantiated.	python Copy code `class Person:` `  def __init__(self, name, age):` `  self.name = name` `  self.age = age` `p = Person("Alice", 25)` `print(p.name)  # Output: Alice` `print(p.age)  # Output: 25`
	Làm thế nào để chuyển một chuỗi (string) sang chữ thường (lowercase) trong Python? Trong Python, bạn có thể chuyển một chuỗi sang chữ thường bằng phương thức lower() của chuỗi.	How do you convert string to lowercase in Python? In Python, you can convert a string to lowercase using the lower() method of the string.	python Copy code `text = "Hello, World!"` `lower_text = text.lower()` `print(lower_text)  # Output: hello, world!`

	How do you use the split method in Python? In Python, the split() method is used to divide a string into a list of substrings based on a separator. By default, it splits by whitespace.	Làm thế nào để sử dụng phương thức split() trong Python? Trong Python, phương thức split() dùng để chia một chuỗi thành danh sách các chuỗi con dựa trên kí tự phân tách. Mặc định, nó chia theo khoảng trắng.	python Copy code `text = "Python is fun"` `words = text.split()  # Split by whitespace` `print(words)  # Output: ['Python', 'is', 'fun']` `csv = "a,b,c"` `items = csv.split(",")  # Split by comma` `print(items)  # Output: ['a', 'b', 'c']`
	Khối Try (Try Block) trong Python là gì? Trong Python, khối try dùng để bao quanh đoạn code có thể phát sinh lỗi (exception). Nếu lỗi xảy ra trong try, nó có thể bắt và xử lý trong khối except. Try giúp chương trình không bị dừng khi có lỗi.	What is a Try Block? In Python, a try block is used to wrap code that might raise an exception. If an exception occurs inside the try block, it can be caught and handled in the corresponding except block. The try block helps prevent the program from crashing due to errors.	main.py > ... `try:` `    x = 10 / 0` `except ZeroDivisionError:` `    print("Cannot divide by zero!")` `# Output` `# Cannot divide by zero!`
	Generator trong Python là gì? Trong Python, generator là hàm trả về một iterator và phát sinh (yield) từng giá trị một bằng từ khóa yield thay vì return. Generator giúp xử lý dữ liệu lớn hiệu quả hơn vì không lưu toàn bộ giá trị trong bộ nhớ, mà tạo ra từng giá trị khi cần. "Generator là hàm dùng yield để tạo giá trị từng cái một, giúp tiết kiệm bộ nhớ."	What are generators in Python? In Python, a generator is a function that returns an iterator and yields values one at a time using the yield keyword instead of return. Generators are used to handle large data efficiently because they don't store all values in memory — they generate them on the fly. "A generator is a function that yields values one by one using yield, allowing efficient memory use."	main.py > count_up_to `def count_up_to(n):` `    i = 1` `    while i <= n:` `        yield 1` `        i +=1` `for num in count_up_to(3):` `    print(num)` `# Output:` `# 1` `# 2` `# 3`
	Làm thế nào để truy cập một module viết bằng Python từ ngôn ngữ C? Một module Python có thể được truy cập từ ngôn ngữ C bằng cách nhúng trình thông dịch Python (Python interpreter) vào chương trình C. Việc này thực hiện thông qua Python/C API, có trong file tiêu đề Python.h. Bạn khởi tạo trình thông dịch, import module Python, rồi gọi hàm của nó qua API. "Có thể truy cập module Python từ C bằng cách nhúng trình thông dịch Python và dùng Python/C API."	How can a module written in Python be accessed from C? A Python module can be accessed from C by embedding the Python interpreter into a C program. This is done using the Python/C API provided by the Python.h header file. You initialize the interpreter, import the module, and then call its functions using the API. "You can access a Python module from C by embedding the Python interpreter and using the Python/C API."	main.cpp `#include <Python.h>` `int main() {` `    // Start the Python interpreter` `    Py_Initialize();` `    // Import Python module` `    PyObject *module = PyImport_ImportModule("mymodule");` `    // Run Python code` `    PyRun_SimpleString("print('Accessed from C!')");` `    // End the interpreter` `    Py_Finalize();` `    return 0;` `}`

	Làm thế nào để đảo ngược (reverse) một list trong Python? Trong Python, có thể đảo ngược list bằng vài cách: 1. Dùng phương thức reverse() — đảo trực tiếp trên list gốc. 2. Dùng cú pháp cắt (slicing) — list[::-1] tạo một bản sao đảo ngược.	How a list can be reversed in Python? In Python, a list can be reversed in several ways: 1. Using the reverse() method — it reverses the list in place (modifies the original list). 2. Using slicing — list[::-1] creates a reversed copy of the list.	python Copy code ```python # Method 1: Using reverse() numbers = [1, 2, 3, 4] numbers.reverse() print(numbers) # Output: [4, 3, 2, 1] # Method 2: Using slicing nums = [1, 2, 3, 4] reversed_nums = nums[::-1] print(reversed_nums) # Output: [4, 3, 2, 1] ```
	Có những cách nào để kết hợp (combine) các DataFrame trong Python? Trong Python (dùng thư viện pandas), có nhiều cách để kết hợp các DataFrame tùy vào mục đích: 1. concat() – Ghép các DataFrame theo hàng hoặc theo cột. 2. merge() – Kết hợp dựa trên các cột hoặc khóa chung, tương tự lệnh JOIN trong SQL. 3. join() – Kết hợp dựa trên chỉ số (index). "Có thể kết hợp DataFrame bằng concat() để ghép, merge() để nối theo cột chung, và join() để nối theo index."	What are ways to combine dataframes in Python? In Python (using pandas), there are several ways to combine DataFrames depending on how you want to merge the data: 1. concat() – Combines DataFrames along rows or columns. 2. merge() – Combines DataFrames based on common columns or keys (like SQL joins). 3. join() – Combines DataFrames based on their indexes. "We can combine DataFrames using concat() for stacking, merge() for SQL-style joins, and join() for index-based merging."	python Copy code ```python import pandas as pd df1 = pd.DataFrame({'A': [1, 2], 'B': [3, 4]}) df2 = pd.DataFrame({'A': [5, 6], 'B': [7, 8]}) # 1. concat() combined = pd.concat([df1, df2]) print(combined) # 2. merge() df3 = pd.DataFrame({'A': [1, 2], 'C': ['x', 'y']}) merged = pd.merge(df1, df3, on='A') print(merged) # 3. join() joined = df1.join(df3.set_index('A'), on='A') print(joined) ```
	PIP là gì trong Python? PIP là trình cài đặt package cho Python. Nó cho phép bạn cài đặt, nâng cấp và quản lý các package Python từ Python Package Index (PyPI) hoặc các kho khác. Đây là công cụ chuẩn để thêm thư viện bên thứ ba vào môi trường Python. "PIP là trình cài đặt package cho Python, dùng để cài đặt và quản lý thư viện bên ngoài."	What is a PIP? PIP is the package installer for Python. It allows you to install, upgrade, and manage Python packages from the Python Package Index (PyPI) or other repositories. It's the standard tool to add third-party libraries to your Python environment. "PIP is the package installer for Python, used to install and manage third-party libraries."	bash Copy code ```bash # Cài đặt package pip install requests # Nâng cấp package pip install --upgrade requests # Liệt kê các package đã cài pip list ```

	Tại sao lại sử dụng finalize trong Python? Trong Python, finalize (từ module weakref) được dùng để đăng ký một hàm callback sẽ được gọi khi một đối tượng sắp bị garbage collection. Nó hữu ích để dọn dẹp tài nguyên như đóng file hoặc kết nối mạng khi đối tượng không còn cần nữa, mà không phải dựa vào __del__(). "finalize dùng để đăng ký callback dọn dẹp khi một đối tượng sắp bị garbage collection."	Why is finalize used in Python? In Python, finalize (from the weakref module) is used to register a callback function that will be called when an object is about to be garbage collected. It is helpful for cleaning up resources like closing files or network connections when an object is no longer needed, without relying on __del__(). "finalize is used to register a callback for cleanup when an object is about to be garbage collected."	main.py > ... 1 import weakref 2 3 class MyClass: 4 def _init_(self, name): 5 self.name = name 6 7 def goodbye(obj_name): 8 print(f"{obj_name} is being finalized") 9 10 obj = MyClass("TestObject") 11 finalizer = weakref.finalize(obj, goodbye, obj.name) 12 13 del obj # Triggers the finalize callback 14 15 # Output: 16 # TestObject is being finalized 17
	Phân biệt giữa override và new modifiers. Sự khác biệt giữa override và new (thường trong C#; Python không có từ khóa này nhưng ý tưởng tương tự): 1. Override: - Khi lớp con muốn cung cấp triển khai mới cho một phương thức đã định nghĩa trong lớp cha. - Đảm bảo hành vi đa hình, phương thức lớp con được gọi ngay cả khi tham chiếu là lớp cha. 2. New: - Khi lớp con định nghĩa phương thức cùng tên với lớp cha nhưng không override. - Ẩn phương thức lớp cha, phương thức lớp cha được gọi nếu tham chiếu là kiểu lớp cha.	Differentiate between override and new modifiers. The difference between override and new modifiers (common in languages like C#; Python doesn't have explicit modifiers but conceptually similar behavior can exist) is: 1. Override: - Used when a derived class wants to provide a new implementation for a method already defined in the base class. - Ensures polymorphic behavior, so the derived class method is called even when using a base class reference. 2. New: - Used when a derived class defines a method with the same name as in the base class but does not override it. - It hides the base class method; the base class version is called when the object is referenced as the base type.	
	Trong Python, làm thế nào để tạo một class rỗng (empty class)? Trong Python, để tạo một class rỗng, bạn định nghĩa class bằng từ khóa class và dùng pass để chỉ rằng class không có nội dung.	In Python, what would you do to create an empty class? In Python, to create an empty class, you define the class with the class keyword and use pass inside to indicate no content.	python Copy code class EmptyClass: pass obj = EmptyClass() print(obj) # Output: <__main__.EmptyClass object at 0x...>
	Bạn có nghĩ rằng có thể gọi lớp cha (parent class) mà không cần tạo instance của nó không? Trong Python, bạn có thể gọi phương thức hoặc truy cập thuộc tính của lớp cha mà không cần tạo instance nếu phương thức đó là class method (@classmethod) hoặc static method (@staticmethod). Với phương thức bình thường, bạn cần tạo instance của lớp cha. "Bạn có thể gọi lớp cha mà không tạo instance nếu phương thức là static hoặc class method."	Do you think you can call the parent class without its instance creation? In Python, you can call methods or access attributes of a parent class without creating an instance of it if the method is a class method (@classmethod) or a static method (@staticmethod). For regular instance methods, you normally need an instance of the parent class. "You can call a parent class without creating an instance if the method is static or a class method."	python Copy code class Parent: @staticmethod def greet(): print("Hello from Parent") # Call static method without creating instance Parent.greet() # Output: Hello from Parent

	Có những cách nào để truy cập các thành phần (members) của lớp cha (parent) trong lớp con (child)? Trong Python, các thành phần của lớp cha (thuộc tính hoặc phương thức) có thể được truy cập trong lớp con bằng nhiều cách: 1. Dùng super() – Gọi phương thức hoặc truy cập thuộc tính của lớp cha một cách an toàn và linh hoạt. 2. Dùng trực tiếp tên lớp cha – Gọi phương thức hoặc truy cập thuộc tính một cách rõ ràng. 3. Kế thừa trực tiếp – Nếu thành phần là public, lớp con có thể dùng trực tiếp.	In what ways can parent members in a child class be accessed? In Python, parent class members (attributes or methods) can be accessed in a child class in several ways: 1. Using super() – Calls parent methods or accesses parent attributes in a safe and flexible way. 2. Directly using the parent class name – Calls parent methods or accesses attributes explicitly. 3. Inherited automatically – If the member is public, the child class can use it directly.	main.py > ... 1 class Parent: 2 def _init_(self): 3 self.value - 10 4 5 def show(self): 6 print("Parent value:", self.value) 7 8 class Child(Parent): 9 def display(self): 10 # Using super() 11 super().show() 12 # Using parent class name directly 13 Parent.show(self) 14 # Direct access to inherited attribute 15 print("Access value directly:", self.value) 16 17 c = Child() 18 c.display() 19 20 # Parent value: 10 21 # Parent value: 10 22 # Access value directly: 10
	Pandas trong Python là gì? Pandas là thư viện Python dùng để xử lý và phân tích dữ liệu. Nó cung cấp các cấu trúc dữ liệu như DataFrame và Series, giúp xử lý dữ liệu dạng bảng hoặc dữ liệu theo thời gian, thực hiện lọc, tổng hợp và các phép toán thống kê một cách hiệu quả. "Pandas là thư viện Python để xử lý và phân tích dữ liệu, dùng các cấu trúc DataFrame và Series"	What are Pandas in Python? Pandas is a Python library used for data manipulation and analysis. It provides data structures like DataFrame and Series that make it easy to handle tabular or time-series data, perform filtering, aggregation, and statistical operations efficiently. "Pandas is a Python library for data manipulation and analysis, using DataFrame and Series structures."	python Copy code import pandas as pd data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]} df = pd.DataFrame(data) print(df) markdown Copy code Name Age 0 Alice 25 1 Bob 30
	NumPy là gì? NumPy (Numerical Python) là thư viện Python cho tính toán số học. Nó cung cấp mảng đa chiều hiệu suất cao (ndarray) và các hàm để thực hiện các phép toán toán học, logic và thống kê một cách hiệu quả. NumPy được dùng nhiều trong tính toán khoa học, phân tích dữ liệu và học máy. "NumPy là thư viện Python cho tính toán số học, với mảng hiệu suất cao và các hàm toán học."	What is a NumPy? NumPy (Numerical Python) is a Python library for numerical computing. It provides a high-performance multidimensional array object (ndarray) and functions to perform mathematical, logical, and statistical operations efficiently. It is widely used in scientific computing, data analysis, and machine learning. "NumPy is a Python library for numerical computing with high-performance arrays and mathematical functions."	python Copy code import numpy as np arr = np.array([1, 2, 3, 4]) print(arr * 2) # Output: [2 4 6 8]

	Tại sao NumPy lại được ưu tiên sử dụng hơn Python lists? NumPy được ưu tiên hơn Python list vì: 1. Hiệu năng: Mảng NumPy được cài đặt bằng C, nhanh hơn nhiều khi tính toán số học trên dữ liệu lớn. 2. Tiết kiệm bộ nhớ: NumPy dùng khối bộ nhớ liên tục và kiểu dữ liệu đồng nhất, tốn ít bộ nhớ hơn list. 3. Phép toán vector hóa: Cho phép thao tác từng phần tử mà không cần dùng vòng lặp, code gọn và nhanh hơn. 4. Hàm toán học phong phú: Cung cấp nhiều hàm toán, thống kê và đại số tuyến tính sẵn có. "NumPy nhanh hơn, tiết kiệm bộ nhớ, hỗ trợ phép toán vector hóa và nhiều hàm toán học hơn Python list."	Why is NumPy preferred over Python lists? NumPy is preferred over Python lists for several reasons: 1. Performance: NumPy arrays are implemented in C and are much faster than Python lists for large-scale numerical computations. 2. Memory efficiency: NumPy arrays use less memory because they store elements in a homogeneous and contiguous block. 3. Vectorized operations: NumPy allows element-wise operations without explicit loops, making code cleaner and faster. 4. Rich mathematical functions: NumPy provides a wide range of built-in mathematical, statistical, and linear algebra functions. "NumPy is faster, more memory-efficient, supports vectorized operations, and has rich mathematical functions compared to Python lists."	python Copy code `import numpy as np` `arr = np.array([1, 2, 3, 4])` `print(arr * 2)  # Output: [2 4 6 8]  (vectorized operation)`
	Làm thế nào để tải dữ liệu từ tệp văn bản (text file) một cách hiệu quả trong Python? Trong Python, dữ liệu từ tệp văn bản có thể được tải hiệu quả tùy thuộc vào kích thước và định dạng: 1. Dùng with open() – Đọc tệp an toàn và tự động đóng tệp. 2. Dùng readlines() hoặc lặp từng dòng – Hiệu quả cho tệp lớn, tránh tải toàn bộ vào bộ nhớ. 3. Dùng pandas.read_csv() – Với dữ liệu có cấu trúc (CSV hoặc tab), nhanh và tiện lợi. 4. Dùng numpy.loadtxt() – Với dữ liệu số, hiệu quả và trả về mảng NumPy.	How can we efficiently load data from a text file? In Python, data from a text file can be efficiently loaded depending on the file size and format: 1. Using with open() – Reads the file safely and automatically closes it. 2. Using readlines() or iterating line by line – Efficient for large files to avoid loading everything into memory. 3. Using pandas.read_csv() – For structured data (like CSV or tab-delimited), this is fast and convenient. 4. Using numpy.loadtxt() – For numerical data, it's efficient and returns NumPy arrays.	python Copy code `# Method 1: Using open() and iteration` `with open("data.txt") as f:` `  for line in f:` `  print(line.strip())` `# Method 2: Using pandas` `import pandas as pd` `df = pd.read_csv("data.txt", delimiter="\t")` `print(df)` `# Method 3: Using NumPy` `import numpy as np` `data = np.loadtxt("data.txt")` `print(data)`
	Reindexing trong Pandas là gì? Trong Pandas, reindexing là thay đổi nhãn hàng hoặc cột của một DataFrame hoặc Series sang tập nhãn mới. Điều này có thể dùng để thêm, xóa hoặc sắp xếp lại thứ tự các hàng/cột. Khi nhãn mới được thêm, giá trị thiếu sẽ được điền NaN theo mặc định hoặc giá trị bạn chỉ định. "Reindexing trong Pandas thay đổi nhãn hàng/cột, thêm hoặc sắp xếp lại và điền giá trị cho các mục thiếu."	What is reindexing in Pandas? In Pandas, reindexing means changing the row or column labels of a DataFrame or Series to a new set of labels. This can be used to add new rows/columns, remove some, or rearrange the order. When new labels are introduced, the missing values are filled with NaN by default or a specified value. "Reindexing in Pandas changes the row or column labels, adding or rearranging them and filling missing values."	python Copy code `import pandas as pd` `df = pd.DataFrame({'A': [1, 2, 3]}, index=[0, 1, 2])` `new_index = [0, 1, 2, 3]  # Adding a new row` `df_reindexed = df.reindex(new_index, fill_value=0)` `print(df_reindexed)` css Copy code `  A` `0 1` `1 2` `2 3` `3 0`

	Làm thế nào để sao chép (copy) một đối tượng (object) trong Python? Trong Python, bạn có thể sao chép một đối tượng bằng hai cách chính: 1. Shallow copy – Tạo đối tượng mới, nhưng các đối tượng lồng bên trong vẫn được chia sẻ. Dùng copy.copy(). 2. Deep copy – Tạo đối tượng mới với tất cả các đối tượng lồng bên trong được sao chép đệ quy. Dùng copy.deepcopy(). "Dùng copy.copy() cho shallow copy và copy.deepcopy() cho deep copy."	How can you copy an object in Python? In Python, you can copy an object in two main ways: 1. Shallow copy – Creates a new object, but nested objects are shared. Use copy.copy(). 2. Deep copy – Creates a new object with all nested objects copied recursively. Use copy.deepcopy(). "Use copy.copy() for shallow copy and copy.deepcopy() for deep copy of objects."	python Copy code `import copy` `original = [[1, 2], [3, 4]]` `# Shallow copy` `shallow = copy.copy(original)` `shallow[0][0] = 100` `print(original)  # Output: [[100, 2], [3, 4]]` `# Deep copy` `deep = copy.deepcopy(original)` `deep[0][0] = 999` `print(original)  # Output: [[100, 2], [3, 4]]  (unchanged)`
	Shallow copy và deep copy trong Python khác nhau như thế nào? Sự khác nhau giữa shallow copy và deep copy trong Python: 1. Shallow copy: - Tạo đối tượng mới, nhưng các đối tượng lồng bên trong vẫn được chia sẻ với bản gốc. - Thay đổi các đối tượng lồng sẽ ảnh hưởng cả bản gốc lẫn bản sao. - Dùng copy.copy(). 2. Deep copy: - Tạo một bản sao hoàn toàn độc lập, bao gồm tất cả các đối tượng lồng bên trong đệ quy. - Thay đổi ở bản sao không ảnh hưởng đến đối tượng gốc. - Dùng copy.deepcopy(). "Shallow copy chỉ sao chép đối tượng cấp trên, chia sẻ các đối tượng lồng, còn deep copy sao chép toàn bộ đệ quy, tạo đối tượng độc lập."	What is shallow and deep copying in Python? The difference between shallow copy and deep copy in Python is: 1. Shallow copy: - Creates a new object, but the nested objects inside are shared with the original. - Changes to nested objects affect both the original and the copied object. - Use copy.copy(). 2. Deep copy: - Creates a completely independent copy, including all nested objects recursively. - Changes in the copied object do not affect the original object. - Use copy.deepcopy(). "Shallow copy copies only the top-level object, sharing nested objects, while deep copy copies everything recursively, making an independent object."	python Copy code `import copy` `original = [[1, 2], [3, 4]]` `shallow = copy.copy(original)` `shallow[0][0] = 100` `print(original)  # Output: [[100, 2], [3, 4]] (nested changed)` `deep = copy.deepcopy(original)` `deep[0][0] = 999` `print(original)  # Output: [[100, 2], [3, 4]] (original unchanged)`
	Pickling là gì? Trong Python, pickling là quá trình chuyển một đối tượng Python thành luồng byte để có thể lưu vào file hoặc gửi qua mạng. Quá trình ngược lại, chuyển byte stream thành đối tượng Python, gọi là unpickling. Pickling thường dùng để lưu trữ đối tượng Python. "Pickling là chuyển đối tượng Python thành byte stream để lưu trữ hoặc gửi đi; unpickling phục hồi lại đối tượng.	What are Pickling? In Python, pickling is the process of converting a Python object into a byte stream so that it can be saved to a file or sent over a network. The reverse process, converting the byte stream back to a Python object, is called unpickling. Pickling is commonly used for persisting Python objects. "Pickling is serializing a Python object to a byte stream for storage or transmission; unpickling restores it."	main.py > ... `import pickle` `data = {'name': 'Alice', 'age': 25}` `# Pickling (serialize)` `with open('data.pkl', 'wb') as f:` `    pickle.dump(data, f)` `# Unpickling (deserialize)` `with open('data.pkl', 'rb') as f:` `    loaded_data - pickle.load(f)` `print(loaded_data)` `# output` `# {'name': "Alice', 'age' : 25}`

	Bạn định nghĩa Unpickling trong Python như thế nào? Trong Python, unpickling là quá trình chuyển một byte stream đã được pickling trở lại thành đối tượng Python ban đầu. Đây là quá trình ngược với pickling và dùng để tải lại đối tượng đã lưu hoặc đã truyền đi trước đó. "Unpickling chuyển byte stream đã pickling thành đối tượng Python ban đầu."	How can you define Unpickling in Python? In Python, unpickling is the process of converting a pickled (serialized) byte stream back into the original Python object. It is the reverse of pickling and is used to load objects that were previously saved or transmitted. "Unpickling converts a pickled byte stream back into the original Python object."	python Copy code `import pickle` `# Assume 'data.pkl' was created by pickling` `with open('data.pkl', 'rb') as f:` `  loaded_data = pickle.load(f)` `print(loaded_data)` bash Copy code `{'name': 'Alice', 'age': 25}`
	Hàm nào được sử dụng để thực hiện Pickling và Unpickling? Trong Python, module pickle cung cấp các hàm để pickling và unpickling: 1. Pickling (chuẩn hóa): Dùng pickle.dump() để ghi đối tượng vào file hoặc pickle.dumps() để tạo byte stream. 2. Unpickling (phục hồi): Dùng pickle.load() để đọc đối tượng từ file hoặc pickle.loads() từ byte stream.	What function is used for Pickling and Unpickling? In Python, the pickle module provides functions for pickling and unpickling: 1. Pickling (serialization): Use pickle.dump() to write an object to a file or pickle.dumps() to get a byte stream. 2. Unpickling (deserialization): Use pickle.load() to read an object from a file or pickle.loads() to load from a byte stream.	main.py > ... `import pickle` `data = {'name': 'Alice', 'age' : 25}` `# Pickling to a file` `with open('data.pkl', 'wb') as f:` `  pickle.dump(data, f)` `# Unpickling from a file` `with open('data.pkl', 'rb') as f:` `  loaded_data = pickle.load(f)` `print(loaded_data)` `# output` `# {'name': 'Alice', 'age': 25}`
	Kể tên một số kiểu chuyển đổi dữ liệu (Type Conversion) trong Python. Trong Python, Type Conversion là chuyển đổi biến từ kiểu dữ liệu này sang kiểu dữ liệu khác. Có hai loại chính: 1. Chuyển đổi ngầm định (Implicit Type Conversion): Python tự động chuyển đổi kiểu dữ liệu, ví dụ int sang float. 2. Chuyển đổi tường minh (Explicit Type Conversion): Bạn chuyển đổi thủ công bằng các hàm sẵn có: - int() – chuyển sang số nguyên - float() – chuyển sang số thực - str() – chuyển sang chuỗi - list() – chuyển sang list - tuple() – chuyển sang tuple - dict() – chuyển sang dictionary	What are some types of Type Conversion in Python? In Python, type conversion is changing a variable from one data type to another. There are two main types: 1. Implicit Type Conversion (Type Casting): Python automatically converts one type to another. Example: converting int to float. 2. Explicit Type Conversion: You manually convert types using built-in functions like: - int() – converts to integer - float() – converts to float - str() – converts to string - list() – converts to list - tuple() – converts to tuple - dict() – converts to dictionary	python Copy code `x = 10  # int` `y = float(x)  # explicit conversion to float` `print(y)  # Output: 10.0`
	Hãy đưa ra một ví dụ về hàm lambda. Hàm lambda trong Python là hàm ẩn danh, viết gọn trong một dòng và dùng từ khóa lambda. "Hàm lambda là hàm ẩn danh, viết gọn trong một dòng, dùng từ khóa lambda."	Give an example of a lambda function. A lambda function in Python is an anonymous, single-line function defined using the lambda keyword. "A lambda function is an anonymous single-line function defined with lambda."	python Copy code `# Lambda function to add two numbers` `add = lambda x, y: x + y` `print(add(5, 3))  # Output: 8`

	Trong Python, Polymorphism (đa hình) là gì? Trong Python, polymorphism (đa hình) là khả năng của các đối tượng khác nhau được truy cập qua cùng một giao diện, cho phép chúng thực hiện hành động khác nhau tùy vào loại của chúng. Thường thấy trong method overriding, operator overloading và duck typing. "Đa hình cho phép các đối tượng khác nhau dùng cùng một giao diện, nhưng thực hiện hành động theo cách riêng."	In Python, what is Polymorphism? In Python, polymorphism is the ability of different objects to be accessed through the same interface, allowing them to perform actions in different ways depending on their type. It is commonly seen in method overriding, operator overloading, and duck typing. "Polymorphism allows different objects to be accessed through the same interface, performing actions in their own way."	main.py > ... 1 class Cat: 2 def speak(self): 3 print("Meow") 4 5 class Dog: 6 def speak(self): 7 print("Woof") 8 9 def make_animal_speak(animal): 10 animal. speak() 11 12 cat = Cat() 13 dog = Dog() 14 15 make_animal_speak(cat) # Output: Meow 16 make_animal_speak(dog) # Output: Woof 17
	Bạn sẽ định nghĩa hàm swapcase() trong Python như thế nào? Trong Python, swapcase() là một phương thức của chuỗi dùng để chuyển tất cả chữ hoa thành chữ thường và chữ thường thành chữ hoa trong chuỗi. Các ký tự không phải chữ cái không bị thay đổi. "swapcase() chuyển chữ hoa thành chữ thường và chữ thường thành chữ hoa trong chuỗi."	How would you define the Swapcase () in Python? In Python, the swapcase() method is a string method that converts all uppercase letters to lowercase and all lowercase letters to uppercase in a string. It does not change non-alphabetic characters. "swapcase() converts uppercase letters to lowercase and lowercase letters to uppercase in a string."	python Copy code text = "Hello World" swapped = text.swapcase() print(swapped) # Output: hELLO wORLD
Advanced	Trong Python, cú pháp [::-1] có tác dụng gì? Hãy đưa ra một ví dụ minh họa. Trong Python, [::-1] là cú pháp dùng để đảo ngược một chuỗi, list hoặc tuple. Nó có nghĩa là: bắt đầu từ cuối về đầu với bước nhảy là -1. "[::-1] đảo ngược một chuỗi hoặc list trong Python."	What does [::-1] do in Python and give an example? In Python, the [::-1] slice notation is used to reverse a sequence such as a string, list, or tuple. It means: start from the end towards the first element with a step of -1. "[::-1] reverses a sequence in Python, like a string or list."	python Copy code text = "Hello" reversed_text = text[::-1] print(reversed_text) # Output: olleH numbers = [1, 2, 3, 4] print(numbers[::-1]) # Output: [4, 3, 2, 1]
	Giải thích cách kết nối cơ sở dữ liệu (database connection) trong Flask (Python framework). Trong Flask, kết nối cơ sở dữ liệu thực hiện bằng cách dùng driver cơ sở dữ liệu hoặc ORM. Các cách phổ biến: 1. Dùng Flask extensions như Flask-SQLAlchemy: - Hỗ trợ ORM, đơn giản hóa thao tác với cơ sở dữ liệu. - Bạn định nghĩa URI của database, tạo đối tượng SQLAlchemy, và thao tác dữ liệu qua các model. 2. Dùng driver trực tiếp (ví dụ sqlite3 hoặc pymysql): - Mở kết nối, tạo cursor, thực thi truy vấn, rồi đóng kết nối.	Explain database connection in Python Flask. In Flask, connecting to a database involves using a database driver or ORM (Object Relational Mapper). Common approaches include: 1. Using Flask extensions like Flask-SQLAlchemy: - Provides ORM support and simplifies database operations. - You define the database URI, create a SQLAlchemy object, and interact with the database through models. 2. Using direct database drivers (like sqlite3 or pymysql): - You manually open a connection, create a cursor, execute queries, and close the connection.	python Copy code from flask import Flask from flask_sqlalchemy import SQLAlchemy app = Flask(__name__) app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db' db = SQLAlchemy(app) class User(db.Model): id = db.Column(db.Integer, primary_key=True) name = db.Column(db.String(50)) # Create tables with app.app_context(): db.create_all()

	Hiệu ứng Dogpile (Dogpile effect) là gì? Hiệu ứng Dogpile xảy ra khi nhiều tiến trình hoặc luồng cố gắng tạo lại cùng một dữ liệu trong cache cùng lúc sau khi dữ liệu hết hạn. Điều này có thể gây quá tải cho cơ sở dữ liệu hoặc hệ thống backend, vì tất cả các yêu cầu đều kích hoạt tính toán nặng cùng lúc. Cách phòng tránh: - Dùng khóa cache (cache lock) hoặc stale-while-revalidate để chỉ một tiến trình tạo lại cache, các tiến trình khác chờ. Ví dụ (khái niệm): Giả sử cache của giá sản phẩm phổ biến hết hạn. Khi đó hàng trăm người dùng truy cập cùng lúc, khiến nhiều truy vấn cơ sở dữ liệu cùng lúc — đây chính là hiệu ứng Dogpile. *"Hiệu ứng Dogpile xảy ra khi nhiều yêu cầu cùng lúc tạo lại cache hết hạn, gây quá tải."*	What is the Dogpile effect? The Dogpile effect occurs when multiple processes or threads try to regenerate the same cached data simultaneously after it expires. This can overload the database or backend system because all requests trigger the same expensive computation at once. Prevention: - Use cache locks or stale-while-revalidate techniques to ensure only one process regenerates the cache while others wait. Example (conceptual): Imagine a web cache for a popular product price. When the cache expires, hundreds of users request the price at the same time, causing many queries to the database simultaneously — this is the Dogpile effect. *"Dogpile effect happens when multiple requests try to regenerate expired cache simultaneously, causing overload."*	
	Python có hỗ trợ kế thừa đa lớp (multiple inheritance) không? Đúng, Python hỗ trợ kế thừa đa lớp, nghĩa là một lớp có thể kế thừa thuộc tính và phương thức từ nhiều lớp cha. Python sử dụng Method Resolution Order (MRO) để xác định thứ tự tìm kiếm phương thức khi gọi. "Python có hỗ trợ kế thừa đa lớp và dùng MRO để xác định thứ tự tìm phương thức."	Are multiple inheritances supported in Python? Yes, Python supports multiple inheritance, which means a class can inherit attributes and methods from more than one parent class. Python uses the Method Resolution Order (MRO) to determine the order in which base classes are searched when calling a method. "Yes, Python supports multiple inheritance and uses MRO to determine method lookup order."	python Copy code `class Parent1:` `  def greet(self):` `  print("Hello from Parent1")` `class Parent2:` `  def greet(self):` `  print("Hello from Parent2")` `class Child(Parent1, Parent2):` `  pass` `c = Child()` `c.greet()  # Output: Hello from Parent1 (MRO decides Parent1 first)`
	Python có sử dụng các phạm vi truy cập (access specifiers) như public, private, protected không? Python không có các phạm vi truy cập nghiêm ngặt như public, private, hay protected như trong Java hoặc C++. Thay vào đó, Python dựa trên quy ước đặt tên: 1. Public: Thuộc tính hoặc phương thức không có dấu gạch dưới, truy cập được ở mọi nơi. 2. Protected: Tiền tố _ (ví dụ _var) để chỉ rằng nó nên được coi là protected, nhưng chỉ là quy ước. 3. Private: Tiền tố __ (ví dụ __var) sẽ thực hiện name mangling, làm cho việc truy cập từ bên ngoài khó hơn (nhưng không tuyệt đối). "Python dùng quy ước đặt tên thay vì access specifiers nghiêm ngặt: public, protected (_), private (__ với name mangling)."	Does Python make use of access specifiers? Python does not have strict access specifiers like public, private, or protected found in languages such as Java or C++. Instead, it relies on naming conventions: 1. Public: Any attribute or method without underscores is public and accessible anywhere. 2. Protected: Prefix a single underscore _ (e.g., _var) to indicate it should be treated as protected, but it's only a convention. 3. Private: Prefix a double underscore __ (e.g., __var) triggers name mangling, making it harder (but not impossible) to access from outside the class. "Python uses naming conventions instead of strict access specifiers: public, protected (_), and private (__ with name mangling)."	main.py > ... `class MyClass:` `  public_var = 1` `  _protected_var = 2` `  private_var = 3` `obj = MyClass()` `# Accessible` `print(obj.public_var)` `# Accessible, but should be treated as protected` `print(obj.protected_var)` `# Raises AttributeError` `# print(obj.__private_var)` `# Access private using name mangling` `print(obj._MyClass_private_var)`

	Làm thế nào để tạo constructor (hàm khởi tạo) trong Python? Trong Python, constructor là phương thức đặc biệt __init__() được gọi tự động khi một đối tượng của lớp được tạo. Nó dùng để khởi tạo các thuộc tính của đối tượng. "Constructor trong Python là phương thức __init__() dùng để khởi tạo các thuộc tính của đối tượng."	How do you create a constructor in Python? In Python, a constructor is a special method called __init__() that is automatically invoked when an object of a class is created. It is used to initialize the attributes of the object. "A constructor in Python is the __init__() method used to initialize object attributes."	python Copy code `class Person:` `  def __init__(self, name, age):` `  self.name = name` `  self.age = age` `p = Person("Alice", 25)` `print(p.name)  # Output: Alice` `print(p.age)  # Output: 25`
	Làm thế nào để lưu hình ảnh (image) về máy cục bộ khi bạn biết địa chỉ URL trong Python? Trong Python, bạn có thể lưu hình ảnh từ URL bằng thư viện requests hoặc urllib. Ý tưởng là tải nội dung ảnh và ghi vào file cục bộ ở chế độ nhị phân. "Dùng requests.get() hoặc urllib.request.urlretrieve() để tải ảnh từ URL và lưu vào máy."	How to save an image in Python locally when we know the URL address? In Python, you can save an image from a URL using the requests library or urllib. The idea is to download the image content and write it to a local file in binary mode. "Use requests.get() or urllib.request.urlretrieve() to download an image from a URL and save it locally."	main.py > ... `import requests` `url = "https://example.com/image.jpg"` `response = requests.get(url)` `if response.status_code == 200:` `  with open("local_image.jpg", "wb") as f:` `  f.write(response.content)` `import urllib.request` `url - "https://example.com/image.jpg"` `urllib.request.urlretrieve(url, "local_image.jpg")`
	Giải thích cách hoạt động của hàm join() trong Python. Hàm join() trong Python là một phương thức của chuỗi (string method), được dùng để nối các phần tử của một iterable (như list hoặc tuple) thành một chuỗi duy nhất, với ký tự phân tách được chỉ định giữa các phần tử.	Explain how the join () function in Python? The join() function in Python is a string method used to combine (concatenate) elements of an iterable (like a list or tuple) into a single string, with a specified separator between each element.	python Copy code `words = ['Python', 'is', 'fun']` `sentence = ' '.join(words)` `print(sentence)` kotlin Copy code `Python is fun`

	Làm thế nào để xác định và xử lý giá trị bị thiếu (missing values) trong một DataFrame? Trong Pandas, các giá trị bị thiếu thường được biểu diễn bằng NaN (Not a Number). Bạn có thể xác định và xử lý chúng bằng các hàm tích hợp sẵn.	How can you identify and deal with missing values in a dataframe? In Python's Pandas, missing values are usually represented as NaN (Not a Number). You can identify and handle them using built-in functions.	main.py > ... `import pandas as pd` `df = pd.DataFrame({` `'Name': ['Alice', 'Bob', None],` `'Age': [25, None, 30]` `})` `# 1. Identify missing values:` `print(df.isnull())  # Shows True where values are missing` `print(df.isnull().sum()) # Counts total missing values per column` `# 2. Handle missing values:` `# Remove missing values:` `df = df.dropna()` `# Fill missing values:` `df = df.fillna({'Name': 'Unknown', 'Age': df['Age'].mean()})`
	Tệp manage.py trong Python được dùng để làm gì? Trong Django (một framework web phổ biến của Python), tệp manage.py là một tiện ích dòng lệnh (command-line utility) được tự động tạo ra khi bạn khởi tạo dự án. - Nó giúp bạn thực hiện nhiều tác vụ quản trị như: - Chạy máy chủ phát triển (python manage.py runserver) - Áp dụng các thay đổi cơ sở dữ liệu (python manage.py migrate) - Tạo ứng dụng mới (python manage.py startapp appname) - Tạo tài khoản quản trị (python manage.py createsuperuser) - Chạy kiểm thử (python manage.py test) "manage.py đóng vai trò như một trình bao bọc (wrapper) quanh công cụ django-admin, giúp quản lý và vận hành dự án Django dễ dàng hơn."	What is the use of manage.py in Python? In Django (a popular Python web framework), the manage.py file is an auto-generated command-line utility that helps you interact with your Django project. It allows you to perform various administrative tasks, such as: - Running the development server (python manage.py runserver) - Applying database migrations (python manage.py migrate) - Creating new apps (python manage.py startapp appname) - Creating superusers (python manage.py createsuperuser) - Running tests (python manage.py test) "manage.py acts as a wrapper around Django's django-admin tool and helps manage your project more easily."	
	Giải thích phương thức shuffle và đưa ra một ví dụ minh họa. Phương thức shuffle() trong Python thuộc module random. Nó được dùng để xáo trộn ngẫu nhiên thứ tự các phần tử trong một danh sách (list). Phương thức này thay đổi trực tiếp danh sách gốc, không tạo ra bản sao mới.	Explain the shuffle method and give an example. The shuffle() method in Python is part of the random module. It is used to randomly rearrange (shuffle) the elements of a list in place — meaning it modifies the original list instead of creating a new one.	python Copy code `import random` `numbers = [1, 2, 3, 4, 5]` `random.shuffle(numbers)` `print(numbers)`
	Phương thức nào có thể được sử dụng để tạo số ngẫu nhiên (random numbers) trong Python? Trong Python, bạn có thể tạo số ngẫu nhiên bằng các hàm trong module random. Một số phương thức phổ biến: 1. random.random() – Trả về một số thực ngẫu nhiên từ 0.0 đến 1.0. 2. random.randint(a, b) – Trả về một số nguyên ngẫu nhiên từ a đến b (bao gồm cả a và b). 3. random.uniform(a, b) – Trả về một số thực ngẫu nhiên từ a đến b. 4. random.choice(sequence) – Chọn một phần tử ngẫu nhiên từ một chuỗi hoặc danh sách.	What method can be used to generate random numbers in Python? In Python, you can generate random numbers using functions from the random module. Some commonly used methods are: 1. random.random() – Returns a random float between 0.0 and 1.0. 2. random.randint(a, b) – Returns a random integer between a and b (inclusive). 3. random.uniform(a, b) – Returns a random float between a and b. 4. random.choice(sequence) – Returns a random element from a sequence (like list or tuple).	python Copy code `import random` `print(random.random())  # Output: 0.374 (example)` `print(random.randint(1, 10))  # Output: 7 (example)` `print(random.uniform(1, 5))  # Output: 3.456 (example)` `print(random.choice([10, 20, 30]))  # Output: 20 (example)`

	*args và **kwargs có ý nghĩa gì trong Python?" Trong Python, *args và **kwargs được dùng trong định nghĩa hàm để nhận số lượng tham số linh hoạt. 1. *args (tham số không có tên): - Gom tất cả tham số vị trí dư thừa thành một tuple. - Dùng khi bạn không biết trước có bao nhiêu tham số vị trí sẽ được truyền vào. 2. **kwargs (tham số có tên): - Gom tất cả tham số từ khóa dư thừa thành một dictionary. - Dùng khi muốn xử lý các tham số đặt tên một cách linh hoạt.	What does *args, ** kwargs mean in Python? In Python, *args and **kwargs are used in function definitions to allow variable numbers of arguments. 1. *args (Non-keyword arguments): - Collects extra positional arguments passed to a function into a tuple. - Useful when you don't know in advance how many positional arguments will be provided. 2. **kwargs (Keyword arguments): - Collects extra keyword arguments passed to a function into a dictionary. - Useful when you want to handle named arguments dynamically.	python Copy code `def demo(*args, **kwargs):` `  print("args:", args)` `  print("kwargs:", kwargs)` `demo(1, 2, 3, name="Alice", age=25)` yaml Copy code `args: (1, 2, 3)` `kwargs: {'name': 'Alice', 'age': 25}`
	Flask có phải là mô hình MVC không? Nếu có, hãy giải thích bằng cách sử dụng mô hình MVC. Đúng, Flask có thể được dùng để triển khai mô hình MVC (Model-View-Controller), nhưng Flask không bắt buộc cấu trúc này. Là một micro-framework, Flask cho phép lập trình viên linh hoạt sắp xếp ứng dụng theo mô hình MVC. - Model: Quản lý dữ liệu và logic nghiệp vụ. Trong Flask, thường dùng ORM như SQLAlchemy hoặc thao tác trực tiếp với cơ sở dữ liệu. - View: Đại diện cho giao diện người dùng, thường là các template HTML được render bằng Jinja2 trong Flask. - Controller: Xử lý logic ứng dụng và routing, được thực hiện thông qua các route functions nhận request, xử lý dữ liệu (sử dụng model) và trả về view.	Is Flask an MVC model? If true, justify this using the MVC pattern. Flask can be used to implement the MVC (Model-View-Controller) pattern, but it is not strictly enforced. Flask is a micro-framework, so it gives developers flexibility to structure their application in an MVC way. - Model: Handles data and business logic. In Flask, this is usually done using ORMs like SQLAlchemy or by directly interacting with databases. - View: Represents the user interface, typically HTML templates rendered with Jinja2 in Flask. - Controller: Handles application logic and routing, implemented via Flask route functions that take requests, process data (using models), and return rendered views.	graphql Copy code `app/` `├── models.py  # Model: database tables and business logic` `├── templates/  # View: HTML templates (Jinja2)` `├── routes.py  # Controller: routes and request handling` `└── app.py  # Main application`
	Làm thế nào để kiểm tra xem tất cả ký tự trong một chuỗi có phải là alphanumeric không? Trong Python, bạn có thể dùng phương thức isalnum() của chuỗi để kiểm tra xem tất cả ký tự trong chuỗi có phải là chữ và số hay không (không bao gồm khoảng trắng hay ký tự đặc biệt). Nó trả về True nếu tất cả là alphanumeric, ngược lại False. "Dùng phương thức isalnum(), trả về True nếu tất cả ký tự là chữ hoặc số."	How can we check if all characters in a string are alphanumeric? In Python, you can use the isalnum() string method to check if all characters in a string are alphanumeric (letters and numbers only, no spaces or special characters). It returns True if all characters are alphanumeric, otherwise False. "Use the isalnum() method; it returns True if all characters in a string are letters or digits.	python Copy code `text1 = "Python123"` `text2 = "Python 123"` `print(text1.isalnum())  # Output: True` `print(text2.isalnum())  # Output: False (space is not alphanumeric)`

	Một số module tích hợp (built-in modules) được sử dụng nhiều nhất trong Python là gì? Python có nhiều module tích hợp sẵn (built-in modules) cung cấp các chức năng hữu ích mà không cần cài đặt thêm. Một số module được sử dụng nhiều nhất bao gồm: 1. os – Tương tác với hệ điều hành (file, thư mục, biến môi trường). 2. sys – Truy cập các tham số và hàm hệ thống. 3. math – Thực hiện các phép toán toán học (hàm lượng giác, logarit…). 4. random – Tạo số ngẫu nhiên và các thao tác ngẫu nhiên. 5. datetime – Làm việc với ngày và giờ. 6. son – Mã hóa và giải mã dữ liệu JSON. 7. re – Thao tác với biểu thức chính quy (regex). 8. collections – Các kiểu dữ liệu container nâng cao như Counter, deque, OrderedDict. "Các module tích hợp phổ biến gồm os, sys, math, random, datetime, json, re và collections."	What are some of the most used built-in modules in Python? Python comes with many built-in modules that provide useful functionality without requiring installation. Some of the most commonly used built-in modules are: 1. os – Interact with the operating system (files, directories, environment variables). 2. sys – Access system-specific parameters and functions. 3. math – Perform mathematical operations (trigonometry, logarithms, etc.). 4. random – Generate random numbers and perform random operations. 5. datetime – Work with dates and times. 6. json – Encode and decode JSON data. 7. re – Perform regular expression operations. 8. collections – Specialized container datatypes like Counter, deque, OrderedDict. "Common built-in modules include os, sys, math, random, datetime, json, re, and collections."	
	Những công cụ (tools) nào dùng để debug và thực hiện phân tích tĩnh (static analysis) trong Python? Trong Python, có nhiều công cụ để debug và phân tích tĩnh: 1. Công cụ debug: - pdb – Trình gỡ lỗi tích hợp sẵn, cho phép chạy từng bước, đặt breakpoint và kiểm tra biến. - ipdb – Phiên bản nâng cao của pdb với tích hợp IPython. - Debugger của IDE – Các IDE như PyCharm, VS Code, Spyder cung cấp debugger tích hợp với breakpoint, watch, và call stack. 2. Công cụ phân tích tĩnh (static analysis): - pylint – Kiểm tra chuẩn code, lỗi, và khả năng có lỗi tiềm ẩn. - flake8 – Kiểm tra style code theo PEP 8 và lỗi lập trình. - mypy – Kiểm tra kiểu dữ liệu dựa trên type hints. - pyflakes – Phát hiện lỗi mà không kiểm tra style code. *"Để debug dùng pdb, ipdb, hoặc debugger của IDE; để phân tích tĩnh dùng pylint, flake8, mypy, hoặc pyflakes."*	What are the tools for debugging and performing static analysis in Python? In Python, there are several tools for debugging and static analysis: 1. Debugging tools: - pdb – The built-in Python debugger, allows stepping through code, setting breakpoints, and inspecting variables. - ipdb – An enhanced version of pdb with IPython integration. - IDE debuggers – Most IDEs like PyCharm, VS Code, and Spyder provide integrated debugging with breakpoints, watches, and call stack inspection. 2. Static analysis tools: - pylint – Checks for coding standards, errors, and potential bugs. - flake8 – Linter for style guide enforcement (PEP 8) and code errors. - mypy – Performs type checking based on type hints. - pyflakes – Detects errors without performing style checks. *"For debugging use pdb, ipdb, or IDE debuggers; for static analysis use pylint, flake8, mypy, or pyflakes."*	
	Khi nào phần else trong cấu trúc try-except-else được thực thi? Trong Python, phần else trong cấu trúc try-except-else chỉ được thực thi khi không có ngoại lệ xảy ra trong khối try. - Nếu khối try chạy thành công, else sẽ chạy. - Nếu có lỗi xảy ra, khối except sẽ chạy và else bị bỏ qua. "Phần else chỉ chạy khi try không có lỗi."	When will the else part of try-except-else be executed? In Python, the else block in a try-except-else statement is executed only if no exceptions occur in the try block. - If the code in try runs successfully without errors, the else block executes. - If an exception occurs, the except block runs instead, and the else block is skipped. "The else part runs only if try has no exceptions."	python Copy code `try:` `  result = 10 / 2` `except ZeroDivisionError:` `  print("Cannot divide by zero!")` `else:` `  print("Division succeeded, result is", result)` csharp Copy code `Division succeeded, result is 5.0`

	Viết đoạn code để hoán đổi (swap) hai số trong Python. Trong Python, bạn có thể hoán đổi hai số rất đơn giản bằng tuple unpacking, không cần biến trung gian. "Bạn có thể hoán đổi hai số bằng a, b = b, a hoặc dùng biến tạm thời.	Write a code to swap two numbers in Python. In Python, you can swap two numbers in a very simple way using tuple unpacking, without needing a temporary variable. "You can swap two numbers using a, b = b, a or with a temporary variable."	main.py > ... 1 2 a = 5 3 b = 10 4 5 # Tuple unpacking 6 a, b = b, a # Swap 7 print("a =", a) # Output: a = 10 8 print("b =", b) # Output: b = 5 9 10 # Temporary variable 11 temp = a 12 a = b 13 b = temp 14
	Cho ví dụ code về deep copy và shallow copy. Trong Python, shallow copy và deep copy được dùng để sao chép đối tượng nhưng hành vi khác nhau: - Shallow copy: Sao chép đối tượng bên ngoài, nhưng các đối tượng lồng nhau vẫn tham chiếu đến đối tượng gốc. - Deep copy: Sao chép toàn bộ đối tượng, bao gồm cả các đối tượng lồng nhau, hoàn toàn độc lập với bản gốc. "Shallow copy sao chép đối tượng ngoài nhưng giữ tham chiếu đến các đối tượng lồng, deep copy sao chép toàn bộ, kể cả các đối tượng lồng nhau."	Show code examples of deep and shallow copying. In Python, shallow copy and deep copy are used to copy objects, but they behave differently: - Shallow copy: Copies the object itself, but nested objects are still references to the original. - Deep copy: Copies the object and all nested objects, creating a completely independent copy. "Shallow copy copies the outer object but keeps references to nested objects, while deep copy duplicates everything including nested objects."	main.py > ... 1 import copy 2 3 # Danh sách gốc với danh sách lồng 4 original = [1, 2, [3, 4]] 5 6 # Shallow copy 7 shallow = copy.copy(original) 8 shallow[2][0] = 99 9 print("Original sau shallow copy:", original) # Danh sách lồng bị thay đổi 10 # Output: 11 # Original sau shallow copy: [1, 2, [99, 4]] 12 13 # Deep copy 14 deep = copy.deepcopy(original) 15 deep[2][1] = 88 16 print("Original sau deep copy:", original) # Danh sách gốc không thay đổi 17 # Output: 18 # Original sau deep copy: [1, 2, [99, 4]] 19
	Viết đoạn code để đảo ngược (reverse) nhiều giá trị từ một hàm. Trong Python, một hàm có thể trả về nhiều giá trị dưới dạng tuple, và bạn có thể đảo ngược chúng bằng tuple unpacking hoặc cú pháp slicing [::-1]. "Trả về nhiều giá trị dưới dạng tuple và dùng slicing [::-1] để đảo ngược."	Write a code to reverse multiple values from functions. In Python, a function can return multiple values as a tuple, and you can reverse them using tuple unpacking or the [::-1] slicing. "Return multiple values as a tuple and use slicing [::-1] to reverse them."	python Copy code def get_values(): return 1, 2, 3, 4 # Get values and reverse them a, b, c, d = get_values()[::-1] print(a, b, c, d) # Output: 4 3 2 1
	Những bước nào để tạo mảng 3D, 2D và 1D trong Python? Trong Python, bạn có thể tạo mảng 1D, 2D và 3D bằng thư viện NumPy.	What are the steps to create 3D, 2D, and 1D arrays? In Python, you can create 1D, 2D, and 3D arrays using the NumPy library.	main.py > ... 1 import numpy as np 2 3 array_1d = np.array([1, 2, 3, 4]) 4 print("Mảng 1D:", array_1d) 5 6 array_2d = np.array([[1, 2], [3, 4]]) 7 print("Mảng 2D:\n", array_2d) 8 9 array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]]) 10 print("Mảng 3D:\n", array_3d) 11

	Viết đoạn code để kiểm tra hai từ có phải là anagram hay không. Trong Python, hai từ là anagram nếu chúng chứa cùng các chữ cái với cùng tần số, bất kể thứ tự. Một cách đơn giản là sắp xếp các chữ cái và so sánh.	Write a code to check whether two words are anagrams or not. In Python, two words are anagrams if they contain the same letters with the same frequency, regardless of order. One simple way is to sort the letters and compare.	main.py > ... 1 def are_anagrams(word1, word2): 2 return sorted(word1.lower()) == sorted(word2.lower()) 3 4 # Test 5 print(are_anagrams("listen", "silent")) # Output: True 6 print(are_anagrams("hello", "world")) # Output: False 7 8 9 # Another way is to use collections.Counter 10 from collections import Counter 11 12 def are_anagrams(word1, word2): 13 return Counter(word1.lower()) == Counter(word2.lower()) 14
	Viết đoạn code để kiểm tra mode trong một danh sách các số. Trong Python, mode của danh sách là số xuất hiện nhiều nhất. Bạn có thể dùng module statistics để tìm mode. Lưu ý: Nếu có nhiều số cùng tần số xuất hiện cao nhất, statistics.mode() sẽ trả về số xuất hiện đầu tiên.	Write a code to test for the mode in a list of numbers. In Python, the mode of a list is the number that appears most frequently. You can use the statistics module to find it. Note: If multiple numbers have the same highest frequency, statistics.mode() will return the first one encountered.	main.py > ... 1 from statistics import mode 2 3 numbers = [1, 2, 2, 3, 4, 2, 5, 3, 3, 3] 4 5 # Tìm mode 6 most_frequent = mode(numbers) 7 print("Mode của danh sách là:", most_frequent) 8 9 # Output: 10 # Mode của danh sách là: 3 11
	Cho ví dụ cách truy cập dataset từ bảng tính công khai (publicly shared spreadsheet) ở định dạng CSV được lưu trên Google Drive. Để truy cập một Google Spreadsheet công khai ở định dạng CSV, bạn có thể dùng thư viện pandas và link CSV của file. Các bước: 1. Đảm bảo bảng tính là public. 2. Lấy link CSV xuất ra. Với Google Sheet URL: https://docs.google.com/spreadsheets/d/FILE_ID/edit#gid=0 Link CSV sẽ là: https://docs.google.com/spreadsheets/d/FILE_ID/export?format=csv 3. Dùng pandas.read_csv() để tải dữ liệu:	Show how to access the dataset of a publicly shared spreadsheet in the format of CSV stored in Google Drive. To access a publicly shared Google Spreadsheet in CSV format, you can use Python's pandas library with the file's CSV export link. Steps: 1. Make sure the spreadsheet is public. 2. Get the CSV export link. For a Google Sheet with URL like: https://docs.google.com/spreadsheets/d/FILE_ID/edit#gid=0 The CSV link will be: https://docs.google.com/spreadsheets/d/FILE_ID/export?format=csv 3. Use pandas.read_csv() to load it:	python Copy code import pandas as pd url = "https://docs.google.com/spreadsheets/d/FILE_ID/export?format=csv" df = pd.read_csv(url) print(df.head()) This will **load the spreadsheet as a DataFrame**, allowing you to analyze it in Python.

	Viết đoạn code cộng hai số nguyên mà không dùng toán tử +, khi các số lớn hơn 0. Bạn có thể cộng hai số nguyên dương mà không dùng toán tử + bằng cách sử dụng toán tử bitwise. Ý tưởng là dùng XOR (^) để cộng mà không có nhớ và AND (&) dịch trái để tính phần nhớ, lặp lại đến khi không còn nhớ. "Dùng XOR để cộng không nhớ, AND dịch trái để lấy phần nhớ, lặp đến khi phần nhớ bằng 0."	Write a code to add two integers without using the '+' operator when the numbers are greater than zero. You can add two positive integers without using the + operator by using bitwise operations. The idea is to use XOR (^) for addition without carry and AND (&) shifted left for the carry, repeating until there is no carry left. "Use bitwise XOR for addition and AND shifted left for carry, loop until carry is zero."	python Copy code `def add_without_plus(a, b):` `  while b != 0:` `  carry = a & b  # Find carry` `  a = a ^ b  # Add without carry` `  b = carry << 1  # Shift carry` `  return a` `# Test` `x = 7` `y = 5` `print(add_without_plus(x, y))  # Output: 12`
	Viết đoạn code nhận một chuỗi các số và kiểm tra xem các số có duy nhất (unique) hay không. Trong Python, bạn có thể kiểm tra các số trong chuỗi có duy nhất hay không bằng cách chuyển chuỗi thành set và so sánh độ dài với chuỗi gốc. "Chuyển chuỗi thành set và so sánh độ dài, hoặc kiểm tra từng số với set để đảm bảo tính duy nhất."	Write a code that is given a sequence of numbers and it checks if the numbers are unique. In Python, you can check if numbers in a sequence are unique by converting the sequence to a set and comparing its length with the original sequence. "Convert the sequence to a set and compare lengths, or check each number with a set to ensure uniqueness."	main.py > ... `def are_numbers_unique(numbers):` `  return len(numbers) == len(set(numbers))` `# Kiểm tra` `nums1 = [1, 2, 3, 4, 5]` `nums2 = [1, 2, 2, 3, 4]` `print(are_numbers_unique(nums1))  # Kết quả: True` `print(are_numbers_unique(nums2))  # Kết quả: False` `# Cách khác dùng vòng lặp:` `def are_numbers_unique(numbers):` `  seen = set()` `  for num in numbers:` `  if num in seen:` `  return False` `  seen.add(num)` `  return True`
	Viết chương trình chuyển đổi định dạng ngày từ yyyy-mm-dd sang dd-mm-yyyy. Trong Python, bạn có thể chuyển đổi chuỗi ngày từ yyyy-mm-dd sang dd-mm-yyyy bằng module datetime. "Dùng datetime.strptime() để đọc yyyy-mm-dd, sau đó strftime() để định dạng dd-mm-yyyy."	Write a program to convert the date format from yyyy-mm-dd to dd-mm-yyyy. In Python, you can convert a date string from yyyy-mm-dd to dd-mm-yyyy using the datetime module`. "Use datetime.strptime() to parse yyyy-mm-dd, then strftime() to format dd-mm-yyyy."	python Copy code `from datetime import datetime` `date_str = "2025-11-13"  # Original format yyyy-mm-dd` `# Convert to datetime object` `date_obj = datetime.strptime(date_str, "%Y-%m-%d")` `# Format to dd-mm-yyyy` `new_date_str = date_obj.strftime("%d-%m-%Y")` `print(new_date_str)  # Output: 13-11-2025`

	Viết chương trình Python để tạo dãy Fibonacci. Trong Python, bạn có thể tạo dãy Fibonacci bằng vòng lặp hoặc đệ quy. "Bắt đầu với [0,1] và dùng vòng lặp cộng hai số cuối cùng để tạo dãy Fibonacci."	Write a program in Python to create a Fibonacci series. In Python, you can create a Fibonacci series using a loop or recursion. "Start with [0,1] and use a loop to append the sum of the last two numbers to build the Fibonacci series."	main.py > ... 1 def fibonacci(n): 2 fib_series = [0, 1] # First two numbers 3 for i in range(2, n): 4 fib_series.append(fib_series[i-1] + fib_series[i-2]) 5 return fib_series[:n] 6 7 # Test 8 n = 10 9 print(fibonacci(n)) 10 11 # output 12 # [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
	Viết đoạn code để tạo một chuỗi duy nhất từ các phần tử trong một danh sách (list). Trong Python, bạn có thể tạo một chuỗi duy nhất từ các phần tử trong danh sách bằng phương thức join().	Write a code to create a single string from elements in a list. In Python, you can create a single string from a list of elements using the join() method.	python Copy code words = ["Python", "is", "fun"] # Combine list elements into a single string with spaces single_string = " ".join(words) print(single_string) # Output: Python is fun # Combine without spaces single_string_no_space = "".join(words) print(single_string_no_space) # Output: Pythonisfun
	Viết chương trình để kiểm tra một số có phải là số nguyên tố (prime) hay không. Trong Python, số nguyên tố là số lớn hơn 1 và chỉ chia hết cho 1 và chính nó. Bạn có thể kiểm tra bằng vòng lặp.	Write a program to check if a number is a prime. In Python, a prime number is a number greater than 1 that has no divisors other than 1 and itself. You can check it using a simple loop.	python Copy code def is_prime(n): if n <= 1: return False for i in range(2, int(n**0.5) + 1): if n % i == 0: return False return True # Test num = 29 print(f"{num} is prime? {is_prime(num)}") # Output: True
	Viết chương trình tính median trong Python sử dụng mảng NumPy. Trong Python, bạn có thể tính median của một tập dữ liệu bằng NumPy với hàm np.median().	Write a program to calculate the median in Python using the NumPy arrays. In Python, you can calculate the median of a dataset using NumPy with the np.median() function.	python Copy code import numpy as np # Sample NumPy array data = np.array([10, 20, 30, 40, 50]) # Calculate median median_value = np.median(data) print("Median is:", median_value) # Output: 30.0

	Viết chương trình để thực hiện thuật toán đổi chỗ trực tiếp (interchange sort). Interchange Sort là thuật toán sắp xếp đơn giản, tương tự bubble sort, so sánh từng phần tử với tất cả các phần tử khác và hoán đổi nếu chúng sai thứ tự. "So sánh mỗi phần tử với tất cả phần tử phía sau và hoán đổi nếu sai thứ tự."	Write a program to execute the interchange sort algorithm. Interchange Sort is a simple sorting algorithm similar to bubble sort, where every element is compared with all other elements and swapped if they are out of order. "Compare each element with all subsequent elements and swap if out of order."	main.py > ... 1 def interchange_sort(arr): 2 n = len(arr) 3 for i in range(n-1): 4 for j in range(i+1, n): 5 if arr[i] > arr[j]: 6 arr[i], arr[j] = arr[j], arr[i] 7 return arr 8 9 # Test 10 numbers = [29, 10, 14, 37, 13] 11 sorted_numbers = interchange_sort(numbers) 12 print("Sorted array:", sorted_numbers) 13 14 # Output: 15 # Sorted array: [10, 13, 14, 29, 37] 16
	Viết chương trình để thực hiện thuật toán chọn trực tiếp (selection sort). Selection Sort là thuật toán sắp xếp, chọn phần tử nhỏ nhất (hoặc lớn nhất) từ phần chưa sắp xếp và hoán đổi với phần tử đầu tiên của phần chưa sắp xếp. "Lặp lại chọn phần tử nhỏ nhất từ phần chưa sắp xếp và hoán đổi với phần tử đầu tiên chưa sắp xếp."	Write a program to execute the selection sort algorithm.Write a program to execute the selection sort algorithm. Selection Sort is a sorting algorithm where the smallest (or largest) element is repeatedly selected from the unsorted portion and swapped with the first unsorted element. "Repeatedly select the minimum element from the unsorted part and swap it with the first unsorted element."	main.py > ... 1 def selection_sort(arr): 2 n = len(arr) 3 for i in range(n): 4 # Assume the minimum is the first unsorted element 5 min_idx = i 6 for j in range(i+1, n): 7 if arr[j] < arr[min_idx]: 8 min_idx = j 9 # Swap the found minimum with the first unsorted element 10 arr[i], arr[min_idx] = arr[min_idx], arr[i] 11 return arr 12 13 # Test 14 numbers = [64, 25, 12, 22, 11] 15 sorted_numbers = selection_sort(numbers) 16 print("Sorted array:", sorted_numbers) 17 18 # Output: 19 # Sorted array: [11, 12, 22, 25, 64] 20
	Viết chương trình để thực hiện thuật toán chèn trực tiếp (insertion sort). Insertion Sort là thuật toán sắp xếp đơn giản, lấy từng phần tử và chèn vào vị trí đúng trong phần mảng đã sắp xếp. "Lấy từng phần tử và chèn vào phần mảng đã sắp xếp bằng cách dịch các phần tử lớn hơn sang phải."	Write a program to execute the insertion sort algorithm. Insertion Sort is a simple sorting algorithm where elements are picked one by one and inserted into their correct position in the sorted portion of the array. "Pick each element and insert it into the sorted part of the array by shifting larger elements to the right."	main.py > ... 1 def insertion_sort(arr): 2 for i in range(1, len(arr)): 3 key = arr[i] 4 j = i - 1 5 # Move elements greater than key to one position ahead 6 while j >= 0 and arr[j] > key: 7 arr[j + 1] = arr[j] 8 j -= 1 9 arr[j + 1] = key 10 return arr 11 12 # Test 13 numbers = [12, 11, 13, 5, 6] 14 sorted_numbers = insertion_sort(numbers) 15 print("Sorted array:", sorted_numbers) 16 17 # Output: 18 # Sorted array: [5, 6, 11, 12, 13]

	Viết chương trình để thực hiện thuật toán sắp xếp nổi bọt (bubble sort). Bubble Sort là thuật toán sắp xếp đơn giản, hoán đổi các phần tử kề nhau nếu chúng ở sai thứ tự, lặp lại cho đến khi mảng được sắp xếp. "Duyệt mảng nhiều lần, hoán đổi các phần tử kề nhau nếu sai thứ tự, lặp lại đến khi mảng được sắp xếp."	Write a program to execute the bubble sort algorithm. Bubble Sort is a simple sorting algorithm where adjacent elements are repeatedly swapped if they are in the wrong order. "Loop through the list multiple times, swapping adjacent elements if they're in the wrong order until sorted."	main.py > ... 1 def bubble_sort(arr): 2 n = len(arr) 3 for i in range(n): 4 # Last i elements are already sorted 5 for j in range(0, n-i-1): 6 if arr[j] > arr[j+1]: 7 # Swap 8 arr[j], arr[j+1] = arr[j+1], arr[j] 9 return arr 10 11 # Test 12 numbers = [64, 34, 25, 12, 22, 11, 90] 13 sorted_numbers = bubble_sort(numbers) 14 print("Sorted array:", sorted_numbers) 15 16 # Output: 17 # Sorted array: [11, 12, 22, 25, 34, 64, 90] 18
Trick		Giải thích: - my_list=[] là mutable, được tạo 1 lần khi function định nghĩa. - Các lần gọi sau cùng dùng list cũ, không tạo mới.	main.py main.py > ... 1 def add_item(item, my_list=[]): 2 my_list.append(item) 3 return my_list 4 5 print(add_item(1)) 6 print(add_item(2)) 7 # output 8 # [1] 9 # [1, 2] 10
		Giải thích: Python làm floor division → làm tròn xuống số nguyên gần nhất.	main.py main.py 1 print(-7 // 3) 2 3 # output: 4 # -3 5
		Giải thích: start:end:step, step=2 → lấy 1 ký tự 1 lần nhảy 2.	main.py main.py > ... 1 a = "Python" 2 print(a[::2]) 3 4 # output 5 # "Pto" 6
		Giải thích: - Python cache số nguyên nhỏ từ -5 đến 256. - Nếu a = 257; b = 257 → False.	main.py main.py > ... 1 a = 256 2 b = 256 3 print(a is b) 4 5 # output 6 # True 7

		Giải thích: Python cho phép so sánh “chuỗi”: 1 < x and x < 10.	main.py main.py > ... 1 x = 5 2 print(1 < x < 10) 3 4 # output 5 # True 6
		Giải thích: *b lấy tất cả phần “giữa” giữa a và c.	main.py main.py > ... 1 a, *b, c = [1, 2, 3, 4] 2 print(b) 3 4 # Output: [2, 3] 5
		Giải thích: tuple immutable, list copy không ảnh hưởng tuple gốc.	main.py main.py > ... 1 a = (1,2,3) 2 b = list(a) 3 b[0] = 100 4 print(a) 5 6 # output: (1, 2, 3) 7
		Giải thích: else chạy khi loop không bị break.	main.py main.py > ... 1 for i in range(3): 2 print(i) 3 else: 4 print("Done") 5 6 # output: 7 # 0 8 # 1 9 # 2 10 # Done 11
		Giải thích: Lambda capture biến theo reference, lúc gọi n=2 (lớp cuối cùng). Nếu muốn fix: lambda x, n=n: x+n.	main.py main.py > ... 1 funcs = [lambda x: x+n for n in range(3)] 2 print([f(0) for f in funcs]) 3 4 # Output: [2, 2, 2] 5
		Đáp án: Không lỗi, nhưng ghi đè tên đã import, dễ gây nhầm lẫn.	main.py main.py 1 import math 2 from math import * 3 4 # output 5 # không lỗi 6
		Giải thích: Floating point không chính xác tuyệt đối, 0.1+0.2 ≈ 0.30000000000000004.	main.py main.py 1 print(0.1 + 0.2 == 0.3) 2 3 # output 4 # False 5

		Giải thích: a và b cùng trỏ tới 1 list, thay đổi cùng ảnh hưởng.	main.py main.py > ... 1 a = b = [] 2 a.append(1) 3 print(b) 4 5 # output 6 # [1] 7
		Giải thích: - and trả về giá trị đầu False hoặc giá trị cuối True - or trả về giá trị đầu True hoặc giá trị cuối False	main.py main.py > ... 1 x = 0 2 y = 1 3 print(x and y) 4 print(x or y) 5 6 # output 7 # 0 8 # 1 9
		Giải thích: Python intern small strings, tiết kiệm bộ nhớ.	main.py main.py > ... 1 a = "hello" 2 b = "hello" 3 print(a is b) 4 5 # This will print True because Python caches small strings and reuses them. 6 # output 7 # True 8
		Giải thích: Key trong dict phải immutable, list mutable không dùng được.	main.py main.py > ... 1 my_dict = {[1,2]: "value"} 2 3 # output 4 # Error: unhashable type: 'list' 5

