**python**

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### **What can Python do?**

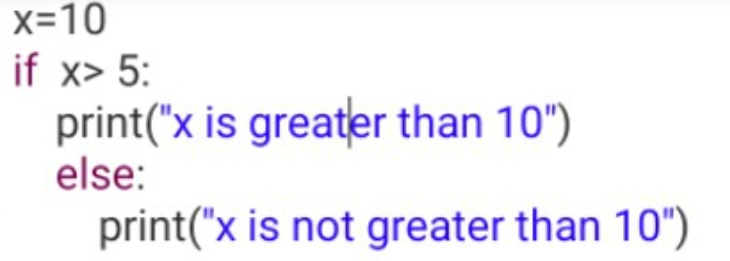
* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

Certainly! Here's a simplified comparison of Python syntax with other programming languages:

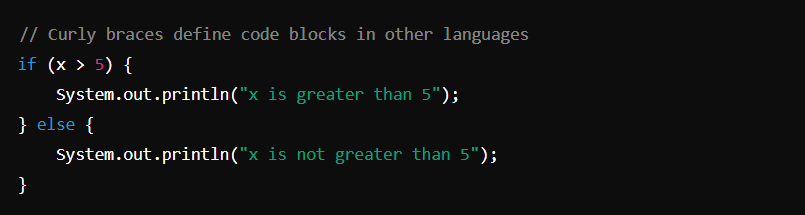
1. **\*\*Indentation\*\*:** In Python, indentation (spaces or tabs at the beginning of lines) is used to define blocks of code, like loops or functions. Other languages often use curly braces `{}` for this.

Example :-

* **Python**:



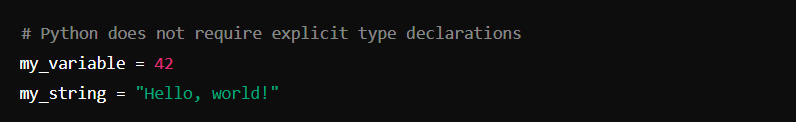
* **Other languages** (e.g., C-like syntax):



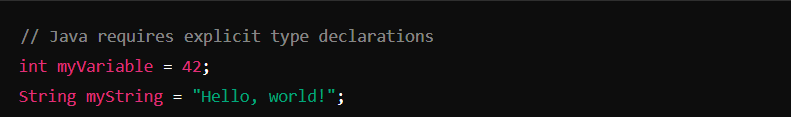
1. **\*\*Dynamic Typing\*\*:** You don't need to declare variable types in Python; it figures out the type when the code runs. Other languages, like C or Java, require you to declare types explicitly.

Example :-

 **Python**:



 **Other languages** (e.g., Java):



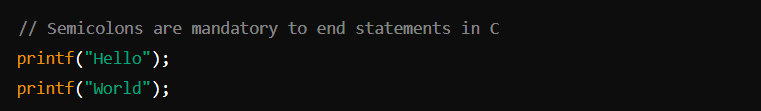
1. **\*\*Semicolons\*\*:** Python doesn't need semicolons to end statements (although you can use them to write multiple statements on one line). Many other languages, like C and Java, require semicolons at the end of each statement.

Example:-

 **Python**:



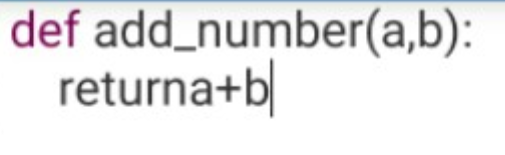
 **Other languages** (e.g., C):



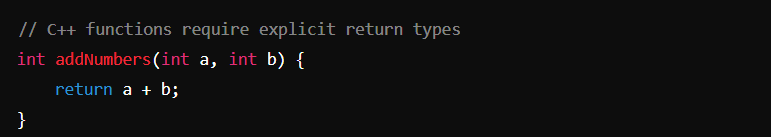
1. **\*\*No Explicit Return Type\*\*:** Python functions don't need a specific return type declared. In languages like C++ or Java, you have to specify what type of data a function will return.

Example:-

 **Python**:



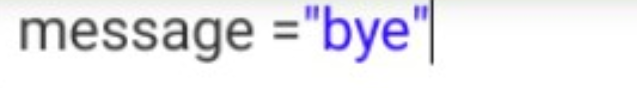
 **Other languages** (e.g., C++):



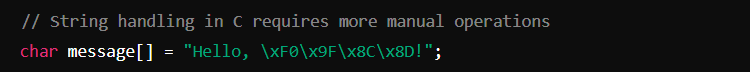
1. **\*\*String Handling\*\*:** Python handles strings in a straightforward way with built-in support for Unicode and powerful string operations. Other languages may require more manual handling of strings.

Example:-

 **Python**:



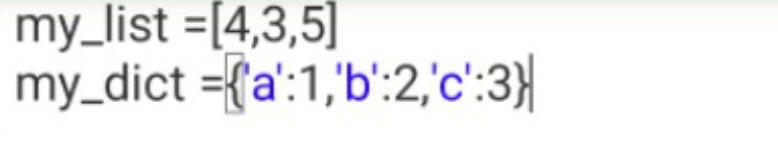
 **Other languages** (e.g., C):



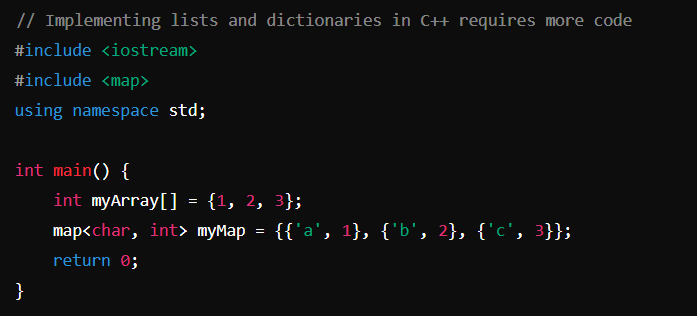
1. **\*\*Lists and Dictionaries\*\*:** Python has built-in support for lists (arrays) and dictionaries (hash maps), which makes handling collections of data easier compared to languages where you might need to write more code to manage these structures.

Example:-

 **Python**:



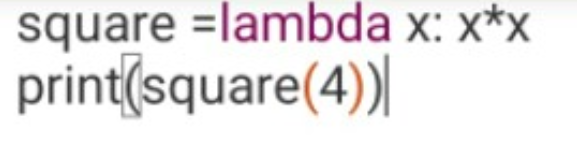
**Other languages** (e.g., C++):



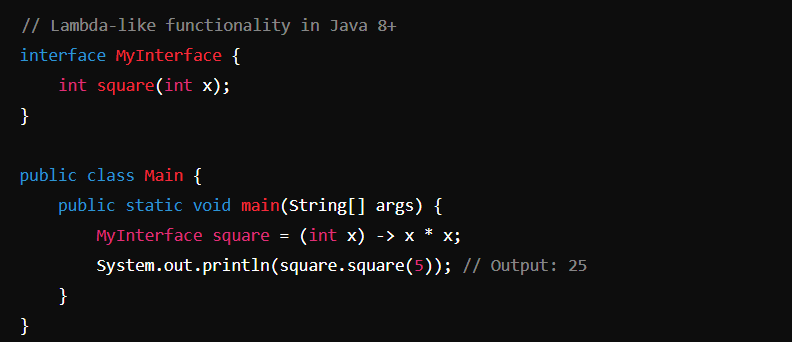
1. **\*\*Lambda Functions\*\*:** Python supports lambda functions for creating small, anonymous functions. Other languages may have similar features but with different syntax.

Example:-

 **Python**:



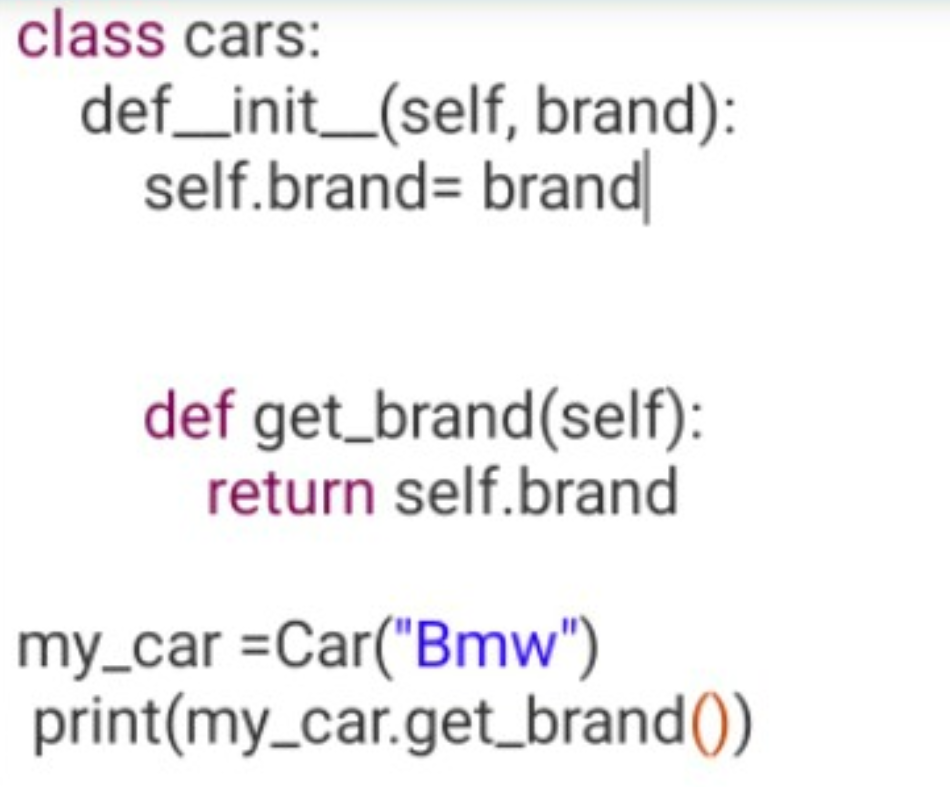
 **Other languages** (e.g., Java):



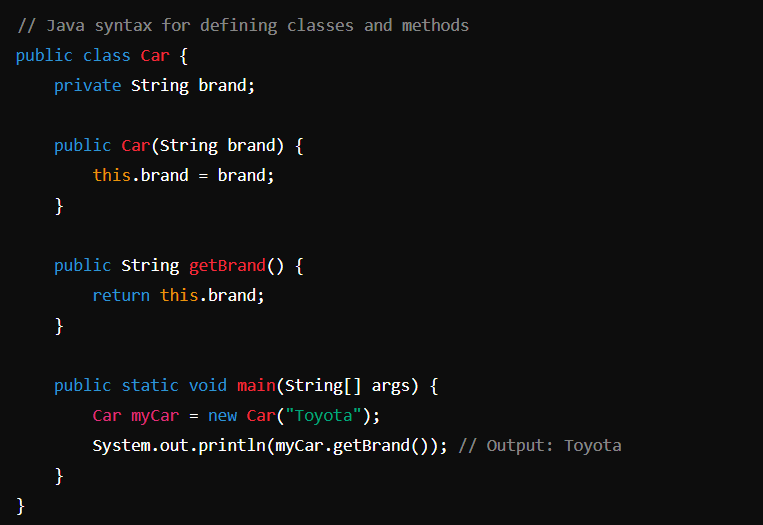
8. **\*\*Object-Oriented Programming\*\*:** Python supports object-oriented programming with simpler syntax compared to languages like Java or C++. Defining classes and methods in Python is more concise and readable.

Example:-

 **Python**:



 **Other languages** (e.g., Java):



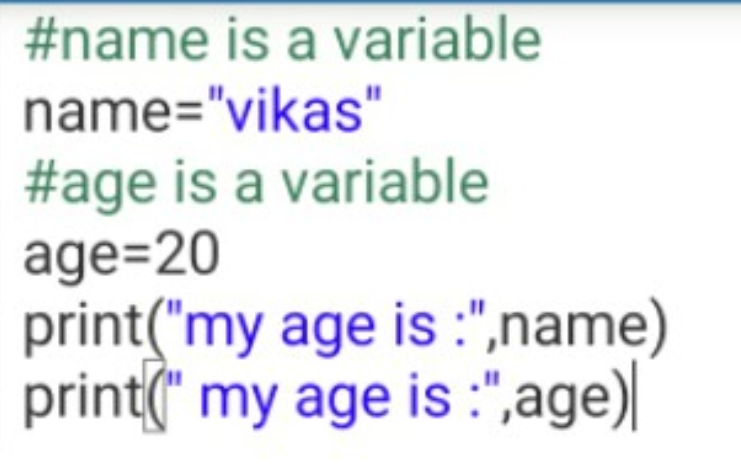
**Difference between POPS and OOPS**

| **Feature** | **POPS (Point of Purchase Systems)** | **OOPS (Order of Payment Systems)** |
| --- | --- | --- |
| **Purpose** | Designed for retail environments to facilitate the sale of goods to customers. | Used in various businesses to manage and track payments and orders. |
| **Main Functionality** | Includes inventory management, sales tracking, and customer management. | Focuses on payment processing, invoicing, and order management. |
| **User Interface** | Often has a graphical user interface suitable for retail employees. | May have a more complex interface for managing orders and payments. |
| **Integration** | Typically integrated with barcode scanners, receipt printers, and cash drawers. | Often integrated with accounting software and online payment gateways. |
| **Examples of Use** | Retail stores, supermarkets, and convenience stores. | E-commerce platforms, subscription services, and service-based businesses. |
| **Payment Handling** | Handles immediate payment transactions at the point of sale. | Manages various payment methods, including deferred payments and invoices. |
| **Inventory Management** | Integral part of the system, tracks stock levels in real-time. | May include basic inventory features but not as robust as POPS. |
| **Customer Interaction** | Direct interaction with customers during the purchase process. | May have limited direct customer interaction, focuses more on backend operations. |
| **Reporting** | Generates sales reports, inventory levels, and customer purchase history. | Provides reports on payments received, outstanding invoices, and order statuses. |
| **Security** | Emphasizes secure transactions at the point of sale. | Focuses on secure handling of payment information and order processing. |

**Variables**

a variable is used to store data that can be referenced and manipulated in your code. Variables can hold different types of data, such as numbers, strings, lists, or even complex objects. Here's a basic overview of variables in Python:

Example:-



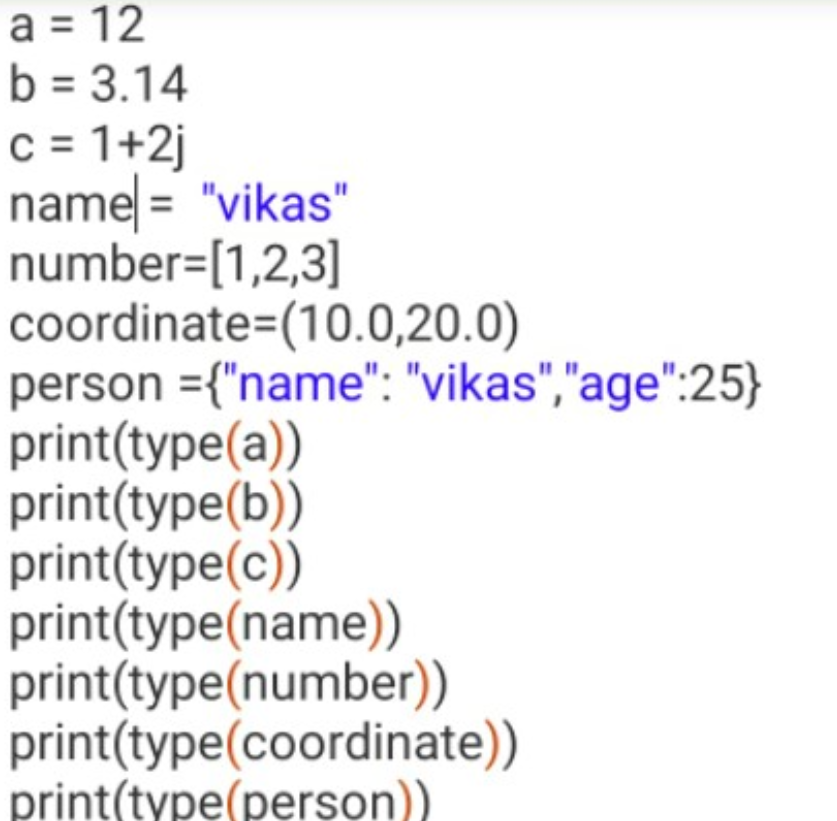
**Variable Naming Rules :-**

* Variable names must start with a letter (a-z, A-Z) or an underscore (\_).
* The rest of the name can contain letters, numbers, or underscores.
* Variable names are case-sensitive (age and Age are different variables).
* Avoid using Python reserved words (keywords) as variable names (e.g., if, else, for, while).

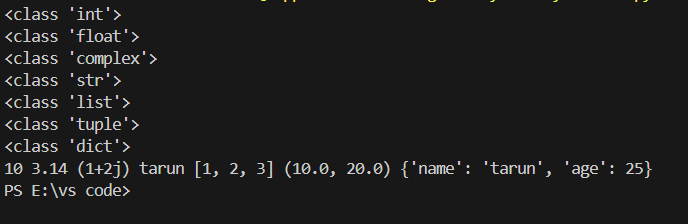
**Data types**

data types are classifications that dictate how the interpreter will treat and store different kinds of data. Here are the primary built-in data types in Python.

Example:-



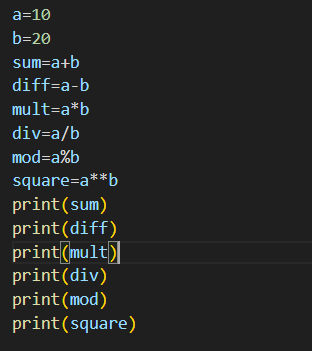
Output:-



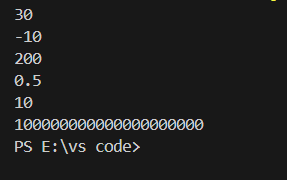
**arithmetic operator**

Arithmetic operators in Python are used to perform basic mathematical operations. Here is a list of the arithmetic operators available in Python along with examples:

Example:-



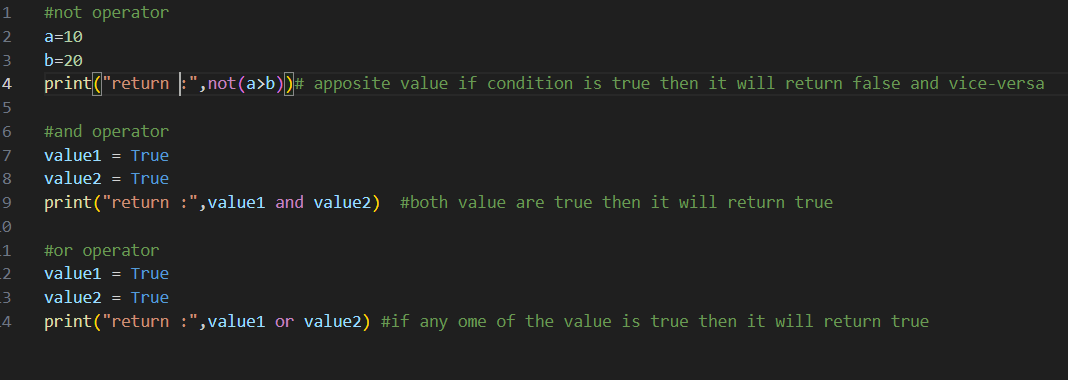
Output:-



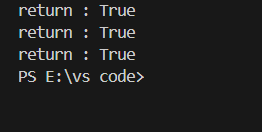
**logical operator**

logic operator are used to combine multiple condition together and evaluate them as a single Boolean expression. There are three types of logical operators in python: and, or, not.

Example:-



Output:-



**if-else condition**

The if, elif, and else statements are used to perform conditional logic. Here’s a basic example to illustrate how these statements are used:

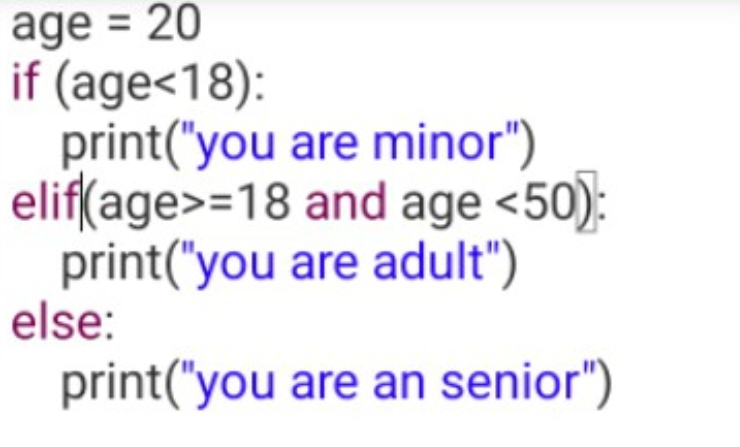
Here’s a breakdown of each component:

1. **\*\*if Statement\*\*:** The if statement evaluates a condition. If the condition is True, the block of code under the if statement is executed.

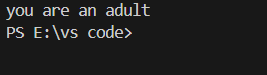
2. **\*\*elif Statement\*\*:** The elif (short for "else if") statement is used to check multiple expressions for True and execute a block of code as soon as one of the conditions is True. You can have multiple elif statements.

3**. \*\*else Statement\*\*:** The else statement is optional and catches anything which isn't caught by the preceding conditions. If none of the previous conditions are True, the block of code under the else statement is executed.

Example:-



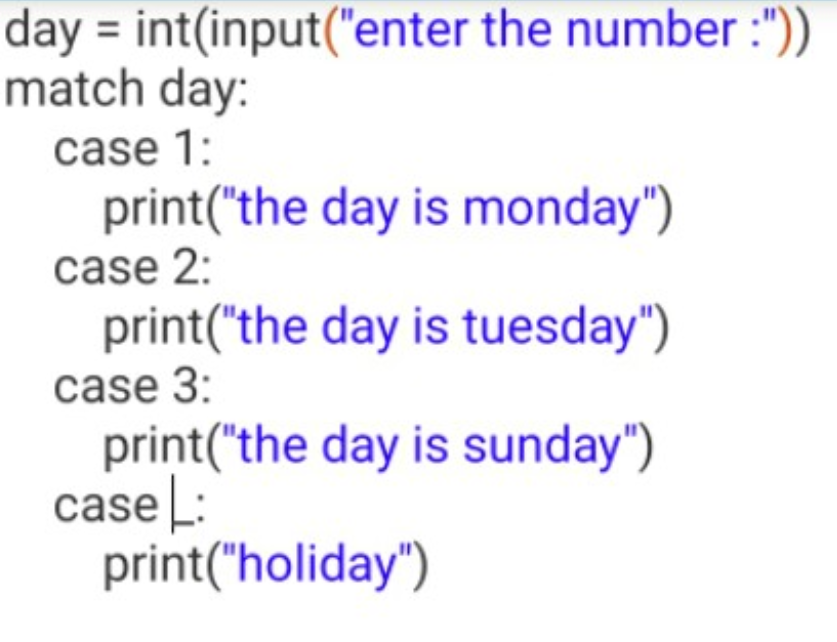
Output:-



**Match case statement in python**

In Python 3.10 and later, the match statement (similar to switch-case statements in other languages) is introduced for pattern matching. It allows for more readable and concise handling of multiple conditions.

Example:-



Output:-

