C++ vs. C#

C++ jest uważany za język średniozaawansowany, co oznacza, że łączy w sobie cechy języków programowania wysokiego i niskiego poziomu. Jest to język wieloparadygmatowy, który obejmuje obsługę paradygmatów programowania proceduralnego, obiektowego, imperatywnego i ogólnego. Jako język niskiego poziomu daje dużą kontrolę nad kodem. C++ jest często preferowanym językiem dla dużych i wbudowanych systemów i jest znany ze swojej szybkości i elastyczności. C++ jest najczęściej implementowany jako język kompilowany, co oznacza, że kod źródłowy kompiluje się bezpośrednio do kodu maszynowego.

C# lub C Sharp to język wysokiego poziomu, który działa w .NET Framework. Jest to język wieloparadygmatowy, który obejmuje obsługę paradygmatów programowania ustrukturyzowanego, imperatywnego, refleksyjnego i współbieżnego. Kod języka C# jest zwykle kompilowany przy użyciu kompilacji just in time (JIT) w Common Language Runtime (CLR), składniku maszyny wirtualnej .NET Framework. Kod jest konwertowany na instrukcje maszynowe, które są wykonywalne na procesorze komputera.

-Języki C# i C++ mają różne podejścia do zarządzania pamięcią: język C# oferuje automatyczne wyrzucanie elementów bezużytecznych, ale język C++ nie. Oznacza to, że język C++ wymaga ręcznego zarządzania tym, co język C# robi automatycznie na maszynie wirtualnej. Zaletą ręcznego zarządzania pamięcią w C++ jest to, że masz również możliwość optymalizacji aplikacji od podstaw.

-Dziedziczenie

C++ Obsługuje dziedziczenie wielokrotne (klasa może dziedziczyć po więcej niż jednej klasie bazowej). W przypadku konfliktu nazw metod przy wielokrotnym dziedziczeniu, programista musi ręcznie rozwiązać ten konflikt.

C# Nie obsługuje dziedziczenia wielokrotnego (klasa może dziedziczyć tylko po jednej klasie). Zamiast tego stosuje interfejsy, które mogą być implementowane w wielu klasach.

-Klasy abstrakcyjne i interfejsy

C++ Klasy abstrakcyjne są tworzone przez deklarowanie przynajmniej jednej funkcji wirtualnej czysto abstrakcyjnej (wirtualna funkcja = 0). C++ nie posiada natywnej koncepcji interfejsów, ale można je symulować za pomocą abstrakcyjnych klas czysto wirtualnych.

C# Klasy abstrakcyjne definiuje się przez słowo kluczowe abstract. Mogą one zawierać zarówno metody abstrakcyjne, jak i zwykłe. Interfejsy są integralną częścią języka (definiowane przez interface), a klasy mogą implementować wiele interfejsów.

-Polimorfizm

C++ Polimorfizm osiąga się za pomocą funkcji wirtualnych. Aby funkcja była wirtualna, trzeba ją zadeklarować słowem kluczowym virtual. Polimorfizm wymaga wskaźników lub referencji.

C# Polimorfizm osiągany jest podobnie jak w C++ za pomocą słowa kluczowego virtual oraz słowa override przy nadpisywaniu. W C# istnieje także słowo kluczowe new, które można użyć do ukrycia metody z klasy bazowej, zamiast ją nadpisywać.

C# vs. Python

Python to interpretowany, zorientowany obiektowo język programowania wysokiego poziomu z dynamiczną semantyką. Jego wysokopoziomowe wbudowane struktury danych, w połączeniu z dynamicznym typowaniem i dynamicznym wiązaniem, sprawiają, że jest bardzo atrakcyjny do szybkiego tworzenia aplikacji, a także do wykorzystania jako język skryptowy lub klejenie do łączenia ze sobą istniejących komponentów. Prosta, łatwa do nauczenia składnia Pythona kładzie nacisk na czytelność, a tym samym zmniejsza koszty utrzymania programu. Python obsługuje moduły i pakiety, co zachęca do modularności programu i ponownego użycia kodu. Interpreter Pythona i obszerna biblioteka standardowa są dostępne w formie źródłowej lub binarnej bez opłat dla wszystkich głównych platform i mogą być swobodnie dystrybuowane.

-Wydajność: C# jest generalnie szybszy niż Python, ponieważ jest językiem kompilowanym. Python jest językiem interpretowanym, co oznacza, że jest wolniejszy niż C#.

-System typów: Python jest językiem typowanym dynamicznie, co oznacza, że typ zmiennej jest określany w czasie wykonywania. Z drugiej strony C# jest językiem statycznie typowanym, co oznacza, że typ zmiennej jest określany w czasie kompilacji.

-Odzyskiwanie pamięci: Python ma automatyczny system odzyskiwania pamięci, co oznacza, że automatycznie zwalnia pamięć. Z drugiej strony język C# ma również system odzyskiwania pamięci, ale wymaga on więcej ręcznej interwencji, aby efektywnie zarządzać pamięcią.

-Deklaracja klasy

Klasy w C# muszą być jawnie zadeklarowane z określeniem modyfikatorów dostępu (np. public, private, internal).

W pythonie klasy są definiowane przy użyciu słowa kluczowego class, ale nie wymaga się modyfikatorów dostępu.

-Dziedziczenie

W C# dziedziczenie jest jawne i można dziedziczyć tylko po jednej klasie (brak wielodziedziczenia), ale możliwe jest dziedziczenie po wielu interfejsach. Python z innej strony obsługuje wielodziedziczenie

-Polimorfizm

W C# polimorfizm jest uzyskiwany za pomocą dziedziczenia i interfejsów. Metody wirtualne i nadpisywanie pozwalają na dynamiczne dostosowywanie metod w klasach pochodnych.

Python obsługuje polimorfizm naturalnie dzięki dynamicznemu typowaniu. Każdy obiekt może być używany, o ile implementuje odpowiednie metody, co daje dużą elastyczność.

-Interfejsy

W C# interfejsy są oddzielnym typem definiującym kontrakt, który klasy mogą implementować

Python nie posiada natywnych interfejsów. Zamiast tego można używać mechanizmów takich jak abc (Abstract Base Class) do definiowania interfejsów:

Python vs. JavaScript

JavaScript to jednowątkowy, synchroniczny język programowania i skryptów szeroko stosowany w tworzeniu stron internetowych. Odgrywa kluczową rolę zarówno w tworzeniu front-endu, jak i back-endu, umożliwiając dynamiczne i interaktywne doświadczenia użytkowników. W przeciwieństwie do języków kompilowanych, JavaScript jest interpretowany przez przeglądarki internetowe, dzięki czemu jest bardzo dostępny i wszechstronny. Jest to język o luźnym typie (lub słabo typowanym), co oznacza, że typy zmiennych są określane w czasie wykonywania.

JavaScript zwiększa interaktywność strony internetowej, umożliwiając dynamiczne aktualizacje zawartości, obsługę kliknięć przycisków, ustawianie aktualizacji czasowych etc.

Python vs. JS: bezpośrednie porównanie

-Definicja obiektów

Obiekty w Pythonie są instancjami klas. Klasy definiują zarówno właściwości, jak i metody. Python jest językiem silnie typowanym, co wpływa na sposób zarządzania obiektami.

Obiekty w JavaScript są bardziej elastyczne i mogą być tworzone jako literały obiektowe, funkcje konstrukcyjne lub klasy (od ES6). JavaScript jest językiem dynamicznie typowanym, co pozwala na większą swobodę w definiowaniu obiektów.

-Klasy i dziedziczenie

Python obsługuje dziedziczenie klas, w tym wielokrotne dziedziczenie. Klasy mogą dziedziczyć metody i właściwości z innych klas, co umożliwia tworzenie złożonych hierarchii.

JavaScript używa dziedziczenia prototypowego, gdzie obiekty mogą dziedziczyć właściwości i metody od innych obiektów. W ES6 wprowadzono syntaktykę klas, co ułatwia pracę z dziedziczeniem.

-Modyfikatory dostępu

W Pythonie nie ma formalnych modyfikatorów dostępu, ale programiści używają konwencji (np. prefiksy \_ i \_\_) do oznaczania, które atrybuty i metody są prywatne lub chronione.

JavaScript również nie ma formalnych modyfikatorów dostępu w tradycyjnym sensie, ale od ES2022 wprowadzono prywatne pola i metody (prefiks #), co pozwala na lepsze enkapsulowanie danych.

-Konstruktor i destruktor

W Pythonie klasy mogą mieć konstruktor (\_\_init\_\_) do inicjalizacji obiektów oraz destruktor (\_\_del\_\_) do wykonywania działań przy usuwaniu obiektów.

JavaScript obsługuje konstrukcję obiektów przez funkcje konstrukcyjne lub klasy, ale nie ma zdefiniowanych destruktorów. Zarządzanie pamięcią jest realizowane przez garbage collection.

-Typy statyczne vs. dynamiczne

Python jest dynamicznie typowany, co oznacza, że typy są określane w czasie wykonywania. Można używać adnotacji typów, ale nie są one obowiązkowe.

JavaScript również jest dynamicznie typowany, co pozwala na elastyczność w pracy z obiektami, ale może prowadzić do błędów typów w czasie wykonania.

JavaScript vs. PHP

PHP (Preprocesor hipertekstu) to popularny język skryptowy typu open source, używany głównie do tworzenia stron internetowych. Jest osadzony w HTML i jest wykonywany po stronie serwera w celu generowania dynamicznych stron internetowych. PHP może wchodzić w interakcję z bazami danych, obsługiwać formularze i zarządzać sesjami, co czyni go wszechstronnym narzędziem do tworzenia aplikacji internetowych. Jest szeroko obsługiwany przez dostawców usług hostingowych i dobrze integruje się z bazami danych, takimi jak MySQL.

-Definicja obiektów

W JavaScript obiekty są podstawowym typem danych i mogą być tworzone w różnorodny sposób, na przykład przez literały obiektowe, funkcje konstrukcyjne, lub klasy (wprowadzone w ES6). Obiekty w JavaScript są dynamiczne i mogą być modyfikowane w czasie rzeczywistym.

PHP wprowadza obiektowość od wersji 5. Obiekty są instancjami klas, a klasy definiują zarówno właściwości, jak i metody. PHP jest językiem silnie typowanym, co wpływa na sposób zarządzania obiektami.

-Klasy i dziedziczenie

JavaScript wspiera dziedziczenie prototypowe. Oznacza to, że obiekty mogą dziedziczyć właściwości i metody od innych obiektów, a nie od klas. W ES6 wprowadzono również klasy, które syntaktycznie przypominają klasy w językach takich jak Java czy C#, ale wciąż opierają się na prototypach.

PHP używa tradycyjnego modelu obiektowego, gdzie klasy mogą dziedziczyć z innych klas. PHP wspiera również wielokrotne dziedziczenie poprzez interfejsy oraz cechy (traits), co daje programistom większą elastyczność.

-Modyfikatory dostępu

W JavaScript nie ma formalnych modyfikatorów dostępu. Zazwyczaj programiści używają konwencji, takich jak prefiksy \_ dla właściwości prywatnych. Od ES2022 wprowadzono prywatne pola i metody (prefiks #), co pozwala na lepsze enkapsulowanie danych.

PHP ma trzy poziomy modyfikatorów dostępu: public, protected i private, co umożliwia precyzyjne zarządzanie dostępem do właściwości i metod w klasach.

-Konstruktor i destruktor

JavaScript Funkcje konstrukcyjne i klasy (od ES6) mogą mieć metody konstrukcyjne, ale JavaScript nie ma zdefiniowanych destruktorów. Zarządzanie pamięcią jest realizowane przez garbage collection.

PHP obsługuje zarówno konstruktory, jak i destruktory w klasach. Konstruktory są wywoływane automatycznie przy tworzeniu instancji, a destruktory przy usuwaniu obiektów.

-Typy statyczne vs. dynamiczne

JavaScript jest językiem dynamicznie typowanym, co oznacza, że typy są określane w czasie wykonywania. Umożliwia to elastyczne zarządzanie obiektami, ale może prowadzić do błędów typów.

PHP 7 i nowsze wersje wprowadziły możliwość deklarowania typów argumentów i typów zwracanych w funkcjach, co wprowadza pewien poziom statycznego typowania, ale wciąż pozostaje w dużej mierze dynamicznie typowanym językiem.

PHP vs. Java

Java to obiektowy język programowania wysokiego poziomu, oparty na klasach, zaprojektowany tak, aby miał jak najmniej zależności implementacyjnych. Jest to język programowania ogólnego przeznaczenia, który umożliwia programistom pisanie raz i uruchamianie w dowolnym miejscu (WORA), co oznacza, że ​​skompilowany kod Java może działać na wszystkich platformach obsługujących język Java bez konieczności ponownej kompilacji. Aplikacje Java są zazwyczaj kompilowane do kodu bajtowego, który można uruchomić na dowolnej wirtualnej maszynie Java (JVM), niezależnie od podstawowej architektury komputera. Składnia języka Java jest podobna do składni języków C i C++, ale ma mniej funkcji niskiego poziomu niż którekolwiek z nich. Środowisko wykonawcze Java zapewnia możliwości dynamiczne (takie jak odbicie i modyfikacja kodu w czasie wykonywania), które zazwyczaj nie są dostępne w tradycyjnych językach kompilowanych.

Porównanie:

-Model Programowania

Java Jest językiem w pełni obiektowym. Wszystko w Javie jest obiektem (z wyjątkiem typów prymitywnych). Oferuje silne wsparcie dla programowania obiektowego, w tym dziedziczenie, polimorfizm, kapsułkowanie oraz interfejsy.

PHP Wprowadził obiektowość w wersji 5, ale przedtem był głównie językiem proceduralnym. Obecnie wspiera programowanie obiektowe, jednak wciąż zachowuje kompatybilność z kodem proceduralnym.

-Dziedziczenie i Polimorfizm

Java Obsługuje dziedziczenie wielopoziomowe i interfejsy, co pozwala na tworzenie złożonych hierarchii klas. Polimorfizm jest realizowany poprzez przesłanianie metod.

PHP Również wspiera dziedziczenie, ale nie ma możliwości dziedziczenia wielokrotnego bezpośrednio. Można korzystać z interfejsów i traitów, co daje pewną elastyczność.

-Kapsułkowanie

Java stosuje modyfikatory dostępu (public, protected, private), co pozwala na precyzyjne zarządzanie dostępem do pól i metod.

PHP oferuje podobne modyfikatory dostępu, jednak ich wykorzystanie w praktyce może być mniej powszechne.

-Składnia i Style Programowania

Java składnia jest bardziej rygorystyczna. Wymaga zdefiniowania typów danych i jest statycznie typowany, co oznacza, że typy muszą być określone na etapie kompilacji.

PHP składnia jest bardziej elastyczna i dynamiczna. Typy danych mogą być określane w locie, co pozwala na szybsze prototypowanie, ale może prowadzić do trudności w utrzymaniu dużych projektów.

PHP to przede wszystkim język skryptowy po stronie serwera, przeznaczony do tworzenia stron internetowych, umożliwiający programistom szybkie tworzenie dynamicznych stron internetowych i aplikacji. Jest interpretowany w czasie wykonywania i jest znany ze swojej łatwości użycia i możliwości szybkiego rozwoju. Natomiast Java jest wszechstronnym, obiektowym językiem programowania, niezależnym od platformy dzięki filozofii „zapisz raz, uruchom gdziekolwiek” dzięki wirtualnej maszynie Java (JVM).