Dr. Oliver Diedrich

Ran an den Code!

Die passende Programmiersprache finden

Von wegen, Programmierer sind fleißig: Statt wiederkehrende Aufgaben von Hand zu erledigen, schreiben sie Code, der dasselbe schneller und besser macht. Das können Sie auch: Mit modernen Sprachen ist der Schritt vom Anwender zum Programmierer nicht groß.

Man muss kein Informatiker sein, um sinnvolle Dinge zu programmieren: Fast alles, was am Rechner monotone Handarbeit erfordert, lässt sich mit ein bisschen Programmcode automatisieren, ohne dass man dazu in die Tiefen von Algorithmen, Datenstrukturen und Protokollen eindringen müsste. Der Übergang vom simplen Batch- oder Shell-Skript, das lediglich ein paar Befehle aufruft, zum „richtigen" Programm ist durch Skriptsprachen wie Powershell oder Python fließend. Dank leistungsfähiger Bibliotheken und Frameworks können schon wenige Zeilen Code erstaunliche Dinge vollbringen.

Viele Aufgaben sind mit ein paar Zeilen eigenem Code schneller gelöst als mit dem Versuch, mit einem fertigen Programm etwas zu tun, wofür es nicht entwickelt wurde. Das heißt ja keineswegs, dass man gleich eine Office-Suite schreiben muss, nur weil der Tabellenkalkulation eine benötigte Funktion fehlt: Viele Anwendungen lassen sich über Makros, eine eingebaute Skriptsprache oder Plug-ins erweitern. Und zahlreiche Open-Source-Projekte warten nur auf neue Mitstreiter, die ihre Ideen einbringen, um die Software besser zu machen.

Welche Sprache wofür?

Programmieren kann man mit jeder Sprache lernen. Der größte Teil des Weges von der Idee zum Programm findet sowieso im Kopf statt: die Zerlegung des Problems in überschaubere Funktionseinheiten. Um beispielsweise die beste Suchmaschine aller Zeiten zu programmieren, müssen Sie zunächst den Suchbegriff erfragen, dann den Datenbestand durchstöbern, die Treffer sortieren und schließlich die Ergebnisse präsentieren. Das Schreiben des Codes in der konkreten Syntax der Programmiersprache spielt beim Programmieren gar keine so große Rolle.

Zudem sind die Grundkonzepte in den meisten Programmiersprachen ähnlich: Wer die Grundlagen in Python beherrscht, findet sich schnell in C, JavaScript, VisualBasic oder PHP zurecht. Auch die Methoden zum Testen des Codes und zur Fehlersuche sind bei allen Sprachen ähnlich.

Trotzdem ist es nicht ganz egal, auf welche Programmiersprache man sich stürzt. Zum einen sind Sprachen unterschiedlich komplex und damit unterschiedlich leicht zu erlernen: Vor dem ersten richtigen Java-Programm kommt man nicht drum herum, sich Gedanken über Klassen und Objekte zu machen, während man in Perl oder Python ähnlich wie in einem Batch-Skript einfach drauflos tippen kann. In C muss man viele Dinge mit vielen Code-Zeilen zu Fuß erledigen, die in moderneren Sprachen wie Python oder Ruby bereits eingebaut sind.

Auch die Voraussetzungen für den Start unterscheiden sich: Compilersprachen wie C und C++ benötigen – der Name sagt es schon – einen Compiler, typischerweise eingebettet in eine Entwicklungsumgebung wie Visual Studio oder Eclipse. Diese bieten viel Hilfestellung für erfahrene Entwickler, Programmiereinsteiger dürften sie jedoch eher verwirren. Einfacher startet es sich mit einer Skriptsprache wie Perl oder Python, die den meisten Linuxen beiliegen und unter Windows mit wenigen Mausklicks installiert sind: Hier kann man seine ersten Code-Zeilen direkt in den Interpreter oder in einen beliebigen Editor eintippen und gleich das Ergebnis sehen (siehe S. 118).

Nicht zuletzt sind die Lernkurven sehr unterschiedlich steil. In einer Sprache wie Python reicht es, ein paar Grundlagen zu Variablen, Kontrollstrukturen und Systemfunktionen zu kennen, um ein Programm zu schreiben, das etwas Sinnvolles tut. Die Standard-Bibliothek von Python enthält zudem eine Fülle fertiger Funktionen zum Lesen und Schreiben unterschiedlichster Datenformate, zum Datenaustausch über das Netzwerk, zum Komprimieren und Verschlüsseln. Das Einlesen und Parsen einer HTML-Seite ist so mit wenigen Zeilen Python-Code erledigt. In einer Low-Level-Sprache wie C benötigt man zusätzliche Bibliotheken, die man erst mal finden und installieren muss, und muss trotzdem mehr eigenen Code schreiben.

Für Ihr Projekt

Die Sprachwahl sollte sich aber auch danach richten, was Sie programmieren wollen. Wenn Sie schon ein konkretes Projekt im Sinn haben, ergibt sich die Programmiersprache häufig von selbst. Steht beispielsweise die Zielplattform fest, nimmt man am besten die Sprache und die Entwicklungsumgebung, die der Hersteller empfiehlt.

Sie wollen eine Android-App schreiben? Willkommen in der Java-Welt und viel Spaß mit Android Studio. Oder eine App für iOS? Dann lautet die Antwort derzeit Objective-C und Xcode – mit der Aussicht, demnächst zu Apples neuer, moderner Programmiersprache Swift zu wechseln. Desktop-Anwendungen für Windows schreibt man typischerweise mit Visual Studio in C#, C++ oder Visual Basic. Für systemnahe Linux-Programme ist C die Sprache der Wahl, ansonsten kann man Linux-Anwendungen mit nahezu jeder Sprache schreiben. Viele Linux-Programmierer verzichten auf eine integrierte Entwicklungsumgebung und schreiben ihren Code im Editor, um dann Compiler oder Interpreter auf der Kommandozeile aufzurufen.

Web-Anwendungen werden immer noch gerne in PHP geschrieben, aber ohne JavaScript ist eine moderne Web-Oberfläche kaum mehr vorstellbar. Dank Node.js können Sie mittlerweile auch den Server-Teil Ihrer Web-App in JavaScript schreiben. Manche Aufgaben lassen sich am besten mit domänenspezifischen Sprachen erledigen: R ist spezialisiert auf die statistische Auswertung von Daten, SQL perfekt geeignet zur Abfrage von Datenbanken.

Wenn Sie sich beruflich qualifizieren wollen, sollte Java – seit Jahren erste Wahl bei

Crashkurs Programmieren

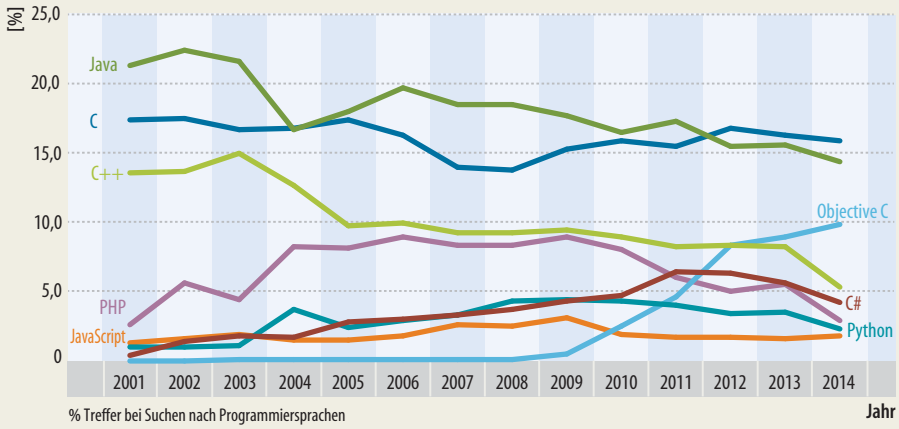
Das gleiche Programm in C, Java, Haskell, Python, PHP und JavaScript: Der Code gibt die Zahlen von 1 bis 10 zusammen mit ihrem Quadrat aus.

der Entwicklung von Unternehmensanwendungen – ganz oben auf Ihrer Liste stehen. Werfen Sie ruhig auch mal einen Blick auf das Programmiersprachen-Ranking von RedMonk oder den TIOBE-Index, der Sprachen nach ihrer Popularität sortiert – darin fließt auch die Nachfrage auf dem Arbeitsmarkt ein. Beide Ranglisten haben ihre Probleme, aber einige Schlussfolgerungen kann man daraus durchaus ziehen: So zählen neben Java die C-Familie (C, C++, Objective-C und C#), PHP, Python und JavaScript zu den populärsten Sprachen.

Eher abraten würden wir von exotischen und Cross-Platform-Ansätzen. Natürlich klingt es für einen Web-Entwickler verlockend, iOS- und Android-Anwendungen mit einem Framework wie PhoneGap oder Appcelerator in HTML, CSS und JavaScript zu programmieren, aber das Ergebnis wird letztlich nicht mit nativ entwickelten Apps mithalten können.

Die populärsten Sprachen

Laut TIOBE Index sind Java und C seit zehn Jahren die unangefochtenen Spitzenreiter unter den Programmiersprachen.



Für alle und alles

Einer Vielzahl von Alltagsproblemen kann man mit jeder Programmiersprache zu Leibe rücken. Die Programmieraufgaben von Codingame.com beispielsweise lassen sich in über 20 verschiedenen Sprachen von C über JavaScript bis Haskell lösen (siehe Seite 130). Dass wir Python wegen des relativ einfachen Einstiegs, der Mächtigkeit der Sprache und den umfangreichen Bibliotheken in solchen Fällen für einen guten Kandidaten halten, haben Sie vielleicht schon bemerkt. Nicht umsonst ist Python auch die Programmierlernsprache Nummer eins auf dem Raspberry Pi.

Überhaupt, die Bibliotheken: Machen Sie nicht den Fehler, das Rad neu erfinden zu wollen! Wer selbst Routinen schreibt, um XML- oder CSV-Dateien einzulesen, eine Webseite übers Netz zu laden oder eine Liste alphabetisch zu sortieren, investiert seine Energie an den falschen Stellen. Derartige Probleme wurden schon hundertfach gelöst; nutzen Sie die Vorarbeit anderer Entwickler und konzentrieren Sie sich auf das, was Ihre Anwendung ausmacht.

Für häufige Aufgaben gibt es Frameworks, die Ihnen viel Arbeit abnehmen. Mit Spiele-Engines wie Unity, der Unreal- oder der CryEngine können Sie sich schneller um den eigentlichen Inhalt Ihres Spiels kümmern, statt sich Gedanken zu machen, welches Objekt zuerst in den Speicher der Grafikkarte geladen wird. Web-Frameworks erleichtern das Entwickeln von Web-Anwendungen in Ihrer Lieblings-Programmiersprache: Beliebt sind Symfony für PHP, Ruby on Rails, Django für Python und Play für Java. Bei Client-seitigem JavaScript nehmen Ihnen jQuery und AngularJS viel Arbeit ab. Grafische Anwendungen auf dem Desktop machen Gtk+ oder Qt plattformunabhängig. Meist spart Ihnen ein Framework schon bei einem einzigen Projekt genug Zeit, damit sich die Einarbeitung lohnt.

Wenn wir Ihnen jetzt Lust gemacht haben, mit dem Programmieren (wieder) anzufangen: Das Web wimmelt nur so von Plattformen, auf denen Sie auch exotische Sprachen lernen können (S. 124). Zum Loslegen brauchen Sie nicht viel: Ein einfacher Editor, am besten mit Syntax-Hervorhebung für die Sprache Ihrer Wahl, und ein Compiler oder Interpreter reichen bereits aus. Wenn die Projekte komplexer werden, lohnt irgendwann der Umstieg auf eine integrierte Entwicklungsumgebung mit Debugger, Profiler, Projekt- und Versionsverwaltung.

(odi@ct.de)

Konzepte

Grundsätzlich unterscheidet man zwischen **imperativen** und **deklarativen** Programmiersprachen. Erstere schreiben dem Computer Schritt für Schritt vor, was er zu tun hat; letztere versuchen, das zu lösende Problem so zu beschreiben, dass es der Rechner (genauer gesagt: der Compiler oder Interpreter) versteht und eine Lösung dafür findet.

Prozedurale Programmiersprachen – die Klassiker der imperativen Programmierung – haben die Welt vom verworrenen, unwartbaren Spaghetticode der Computer-Frühzeit befreit. Sie strukturieren Quelltexte, indem sie ein Programm in überschaubare Funktionseinheiten zerlegen – je nach Sprache Prozedur, Funktion, Routine, Modul oder Unterprogramm genannt. Diese Prozeduren lassen sich von anderen Stellen des Programms aus mit Parametern aufrufen und können ein Ergebnis zurückliefern. Ein typischer Vertreter der prozeduralen Sprachen ist C.

Objektorientierte Sprachen – populäre Vertreter sind Java, C++ und C# – verpacken Daten („Attribute“) und die Funktionen, die mit diesen Daten arbeiten („Methoden“), in Objekte, die miteinander interagieren. Das klingt ziemlich abstrakt, soll es aber einfacher machen, die wirkliche Welt in Software abzubilden: Ein Objekt „Kunde“ hat Attribute wie Kundennummer, Name, Adresse, Zahlungsinformationen et cetera und bietet Methoden, über die man beispielsweise die Adresse ändern oder eine neue Bestellung abwickeln kann. Für jeden Kunden gibt es ein eigenes Kunden-Objekt; die Klasse „Kunde“ legt fest, wie so ein Kunden-Objekt aussieht. Objektorientierte Sprachen sollen das Design und die Implementierung komplexer Programme erleichtern.

Funktionale Sprachen gehören zu den deklarativen Ansätzen und sind vor allem bei Mathematikern und Informatikern beliebt. Manche Klassen von Problemen, die in anderen Programmiersprachen viel Aufwand erfordern, lassen sich mit Haskell und Co. sehr elegant mit wenigen Zeilen Code lösen. Funktionen im Sinne der funktionalen Programmierung haben keine Seiteneffekte, was diese Sprachen ideal für die Entwicklung verteilt arbeitender Software macht. Funktionale Konzepte tauchen zunehmend auch in prozeduralen und objektorientierten Sprachen auf.

c't