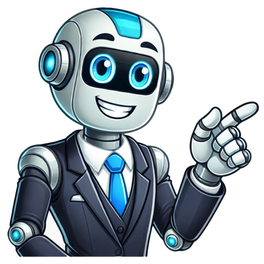
Continue



In general, it takes around two to six months to learn the fundamentals of Python. But you can learn enough to write your first short program in a matter of minutes. Developing mastery of Pythons vast array of libraries can take months or years.How long it takes you to learn Python will depend on several factors, including how much Python you need to know to achieve your desired goal. If you want to learn enough Python to automate a specific task at work, for example, you can likely achieve that more quickly than if you wanted to learn enough Python to get a job as a data analyst. In this article, you'll learn more about how long it may take you to learn Python, depending on your goals. Afterward, if you want to start learning Python today, you might consider enrolling in the University of Michigan's Python for Everybody Specialization to start building your skills. How long does it take to learn basic Python?Its possible to learn the basics of Python in two to six months, though this could be much more or much less, depending on how much time you dedicate to learning. The Python for Everybody Specialization on Coursera, for example, typically takes about four months to complete if youre spending six hours per week on the courses. If you can dedicate more time, lets say two hours per day, you could complete the Specialization in two months.In this and many other introductory courses, you might expect to learn the following foundational syntax and elements of Python:Variables and typesObject and data structures (strings, integers, floats, etc.)Indexing and slicingComparison operatorsFor and while loopsFunctionsLists, dictionaries, and tuplesReading and writing to filesClasses and objectsWeb scrapingApplication programming interfaces (APIs)Once youve built a foundational knowledge of Python, you can begin progressing your programming skills toward your own unique goals, whether it be a job as a data analyst or application developer or the ability to automate tasks at work.You can learn the fundamentals of Python 3 programming in just three months or less in the University of Michigan's Python 3 Programming Specialization: How long does it take to master Python?This depends on what mastery means to you.There are some 15.7 million Python developers in the world, according to Statista [1]. With so many people working on this open-source software, its always evolving to include new tools and capabilities. Youll never know everything there is to know about Python, and thats okay.Some other factors that can influence how quickly you pick up or master Python include:Previous programming experience: If youve written code before, you may find that you pick up Python more quickly.Learning method: Well-structured courses in line with your goals can sometimes accelerate your learning.Time devoted to learning: How much time can you devote to learning and practicing Python? Generally, its a good idea to commit a little time every day.Fortunately, you can leverage AI-assisted coding to learn Python with DeepLearning.AI's online course, AI Python for Beginners. You'll gain knowledge of the programming fundamentals you need to write programs in Python while also building in-demand AI skills in as little as 17 hours, or 3 weeks at 5 hours per week. Python users have access to tens (if not hundreds) of thousands of librariessets of useful functions meant to make coding easier. TensorFlow, for example, can help streamline machine learning programs, while Pandas offers access to flexible and responsive data structures. This means that mastering Python is an ongoing process where you learn what you need to know as you need it.Why learn Python?Python is widely considered among the easiest programming languages for beginners to learn. If youre interested in learning a programming language, Python is a good place to start. Its also one of the most widely used.The TIOBE Index for January 2025 lists Python as the most popular programming language, just ahead of C++ and Java [2]. As you learn, you can take advantage of the robust community of fellow learners and developers, as well as the job opportunities that come with knowing Python.What can you do with Python?Python is also very versatile. Since its a general-purpose language, Python can be used for a variety of tasks, including:Statistical analysisData visualizationMachine learningTask automation or scriptingWeb and software developmentLearn more: Python or R for Data Analysis: Which Should I Learn?Jobs that use PythonPython ranks among the worlds most popular programming languages in part because its used across a variety of industries and job roles. By learning to write Python, you can create opportunities for a variety of careers. Some job titles that use Python include:Data analyst - $85,766Backend developer - $90,659Quality assurance engineer - $96,829Operations automation engineer - $118,064Python developer - $97,931Full-stack developer - $92,004Data engineer - $106,383Data scientist - $117,781Machine learning engineer - $122,557*Salary data represents US average annual base pay from Glassdoor.Read more: 4 Data Analyst Career Paths: Your Guide to Leveling UpTips for learning PythonWhile learning a technical skill like programming with Python may sound intimidating, it may not be as difficult as you think. Keep these tips to enhance your learning.1. Code every day.Python is a language, and just like any other language, repetition is key to learning it. Dedicate time every dayeven if its just 15 minutesto practice coding. Many online Python courses, including Python for Everybody, are broken up into short video lectures, quizzes, and coding practice exercises. This type of structure can make it easier to find time to learn in your life.It may also help you learn more efficiently. Learning in small chunks, a technique known as microlearning, improves retention and engagement.2. Start with the fundamentals.No matter how you plan to use Python in the future, youll want to start with the same set of fundamentals. Learning the basics first will set you up for success when you go on to tackle more complex uses. Whether youre learning on your own or through a course, be sure to cover the fundamentals listed above (See How long does it take to learn basic Python?).3. Focus on logic over syntax.Understanding the how and the why of your lines of code in Python is more important than memorizing the syntax.Remembering to close your parentheses or include a colon before an indent will come with repetition. Plus, you can always look up how to structure your code on Google or Stack Overflow (an online community for programmers). But youll need to understand the logic of what youre trying to accomplish.As youre working through Python coding problems, you may find it helpful to handwrite an outline of what your code needs to do without worrying about syntax. This is called pseudocodea technique even experienced Python programmers use to plan out their programs.4. Let your goal guide your learning.Once youve built a foundation with the basics, the world of Python really opens up. As you progress, knowing your goal and letting it dictate your learning path is important.If youre building the skills for a new career as a data analyst, for example, youll probably want to learn Python skills like data scraping or visualization. If youre more interested in becoming a developer, you may focus on skills like version control and multi-process architecture.The types of libraries, frameworks, and the integrated development environment (IDE) you learn to work with will also vary based on your career goals.5. Join a Python community.Instead of learning to code as a solitary activity, surround yourself (virtually, anyway) with others who are learning Python. This can help boost your motivation while giving you a place to swap tips and tricks with other programmers.Quora and Reddit both have active Python communities. You could also join PySlackers to meet other Python enthusiasts on Slack or the Python Discord channel.Start advancing your Python skills todayPython is a popular and powerful programming language with a wide range of applications. Whether you already have a coding background or are just starting out, Coursera has a Python course for you: For a beginner-friendly introduction to Python, enroll in the University of Michigan's Python for Everybody Specialization. There, you'll learn the fundamentals of Python programming along with how to develop programs to collect, clean, analyze, and visualize data. To become a fluent Python programmer, try the University of Michigan's Python 3 Programming Specialization. In as little as three months, you'll learn the basics of Python 3 along with more advanced functions and gain the ability to write programs and fix errors coding errors. To learn Python for machine learning, explore IBM's Machine Learning with Python course. Build job-ready machine learning skills in Python, such as building, testing, and evaluating models, in as little as six weeks. Frequently asked questions (FAQ)Yes, its totally possible to teach yourself Python. Youll find a variety of resources, from YouTube videos to books to interactive games, that can help you develop your coding skills. Many online courses also allow you to learn at your own pace, but with added structure, a clear learning path, and sometimes a built-in community of other learners.Python is actually one of the best programming languages for beginners. Its syntax is similar to English, which makes it relatively easy to read and understand. With some time and dedication, you can learn to write Python, even if youve never written a line of code before.Its a common misconception that programming requires a lot of math. You dont have to be a math whiz to succeed with Python. It helps to have a basic understanding of arithmetic. Writing Python is more about being able to solve problems by breaking them down into smaller steps, then using your creativity to craft a solution.Unless you have a very specific reason for using Python 2, you should be using Python 3. Most companies use Python 3, plus Python 2 was sunsetted on January 1, 2020. This means that no more features, fixes, or security updates will be added.Python is a valuable skill to have on your resume, but most jobs require a set of skills. If youre going into data analytics, for example, youll likely need to have some proficiency in SQL, statistics, and data visualization. Developers may need to understand data structures, network basics, and testing methods.Show all 5 frequently asked questions Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. How can financial brands set themselves apart through visual storytelling? Our experts explainhow.Learn MoreThe Motorsport Images Collections captures events from 1895 to todays most recentcoverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of EditorsPicks.Browse Editors' FavoritesHow can financial brands set themselves apart through visual storytelling? Our experts explainhow.Learn MoreThe Motorsport Images Collections captures events from 1895 to todays most recentcoverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of EditorsPicks.Browse Editors' FavoritesHow can financial brands set themselves apart through visual storytelling? Our experts explainhow.Learn MoreThe Motorsport Images Collections captures events from 1895 to todays most recentcoverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of EditorsPicks.Browse Editors' Favorites Reddit and its partners use cookies and similar technologies to provide you with a better experience. By accepting all cookies, you agree to our use of cookies to deliver and maintain our services and site, improve the quality of Reddit, personalize Reddit content and advertising, and measure the effectiveness of advertising. By rejecting non-essential cookies, Reddit may still use certain cookies to ensure the proper functionality of our platform. For more information, please see our Cookie Notice and our Privacy Policy. Python is a programming language that powers much of todays technology.Banks use it to detect fraud, researchers process medical data with it, and developers build web applications. The good news? Pythons clean syntax and approachable design make it easy to learn, even without coding experience. In this guide, well share everything you need to know about Pythonhow to find useful resources, learn it fast, and land a job. What is Python?Should you Learn Python in 2025?Key benefits of learning Python: How Long Does it Take to Learn Python?Python vs. other languages: time to learn How to Learn Python in 7 Steps1. List your goals for learning Python2. Choose your learning methodSelf-learningGuided learning3. Get an intro to Python4. Build your first Python project5. Learn intermediary Python skills6. Design your portfolio in the process7. Become a part of the coding community The Top Python Careers for 2025The best ways to learn Python: resourcesOnline courses and tutorialsBooks and written resourcesVideo tutorials and channelsLearning tools and practice platforms FAQsIs Python easy to learn?How much Python do I need to know to get a job?Is Python a high-paying job?What is the salary for junior Python developers?Which is better, Java or Python?How to learn Python faster? Conclusion Python is a high-level programming language, meaning it handles various complex computer tasks automatically. Its also interpretedthe code runs line by line, making it easier to test and debug than languages like C++ or Java. Wondering what you can do with Python? Almost anything. For instance: Analyze massive amounts of data to predict sales or spot fraud Power web development projects, from websites to complex apps Build AI systems that recognize images or answer questions Create games and desktop applications Automate tasks like processing thousands of files Companies use code like this all the time to check if email addresses are real, pull important data from documents, or spot spam emails. Short answer: Yes. Theres a lot of demand for Python, with the TIOBE Index ranking it as the top programming language. Its adoption has grown by about 50% in recent years, and we expect this trend to continue. Prestigious companies like Google, Facebook, Amazon, NASA, and many others use Python for various applications. Its also extremely popular among tech startups of all kinds. So, if youre considering learning a programming language, Python is a safe bet for career prospects. It can also be useful for other professions, including marketing, business analytics, etc. Career opportunities: Python skills open doors to many lucrative and in-demand careers in technology, finance, healthcare, and more. Versatility: Pythons applicability spans web development, data analysis, automation, machine learning, and numerous other domains. Ease of learning: Pythons clear and concise syntax makes it one of the most beginner-friendly programming languages. Community and resources: It has a large and supportive community with various online tutorials, courses, and documentation. Future-proofing: Python continues to grow in popularity and expand into new fields like AI and machine learning. Which means its most likely to stay relevant for years to come. The timeline to learn Python varies widely, depending on your goals and prior programming experience.Heres what it might look like if youre learning consistently (at least 5-10 hours per week). Learning the basics: A few weeks: With focused effort, you can learn the basic Python syntax, data types, and operations in a few weeks. 1-3 months: This is a more typical timeframe for gaining a grasp of the core concepts and being able to write simple programs. Becoming proficient: 6-12 months: It can take several months of consistent practice and learning to become comfortable with Pythons more advanced features, libraries, and frameworks. Ongoing learning: Python is a vast language with numerous libraries and applications. Mastering it means staying up-to-date with all recent changes and developments. Python is often considered one of the easiest languages to learn. Acquiring basic knowledge takes 1-3 months. Java: More complex due to its syntax and concepts like memory management and multithreading. Learning Java can take longer, with basic proficiency achieved in about 3-6 months and advanced proficiency in 1-2 years. JavaScript: The learning curve for basic JavaScript syntax is similar to Python. However, understanding its use in web development, including frameworks like React or Angular, can extend the timeline to 6-12 months for basic mastery. C++: Known for its complexity and steep learning curve, basic C++ proficiency can take six months, with advanced mastery requiring 1-2 years. Ruby: You can master the basics in 1-3 months. Intermediate to advanced proficiency can take 6-12 months, particularly for web development with Ruby on Rails. Follow these practical steps to begin your Python learning journey: Before you even write a line of Python code, take some time to define your learning goals. This will help you move faster, focus on what matters, and stay motivated.Ask yourself these questions: Why do I want to learn Python? Are you looking to switch careers, automate work tasks, build a personal project, or simply expand your knowledge? What do I want to build or achieve with Python? Do you dream of creating web applications, analyzing big data, or diving into machine learning? Or you may want to automate some marketing analytics tasks. How much time am I willing to dedicate to learning Python? Be realistic about your schedule and set achievable milestones. Remember, your goals can be as specific or broad as you want and may change over time. The key is to have a clear direction to guide your learning journey. The good news is you dont need a computer science degree to become proficient in Python. There are two main learning approaches you can take from the comfort of your home: This involves independently using various resources to learn Python at your own pace.Its flexible, often free or low-cost, and allows you to tailor your learning experiences to your specific interests and schedule. Self-learning resources to get you started: However, this method usually takes more time, effort and dedicationand might not fit every learner. If you prefer structure and want to avoid jumping from one resource to another, guided learning is for you.This approach comes with support from instructors or mentors who provide expert guidance and speed up the learning process. It also helps you build practical skills faster and create a professional portfolio. Most guided learning platforms also offer certifications to validate your skillsa huge plus if youre looking to land a programming gig.Platforms to consider: Mimos Python Developer Career Path: This is the best option for beginners looking to build practical skills for the job market. It combines bite-sized lessons with interactive exercises and coding challenges to help you master the basics quickly. The platform allows you to work on projects and build your portfolio. Youll also have access to a community of like-minded learners for motivation and support. Coursera: Offers a vast selection of Python courses from top universities and companies. Youll find options ranging from beginner-friendly introductions to specialized programs in data science, machine learning, and web development.edX: Partners with prestigious institutions like Harvard and MIT to offer university-level Python courses. If youre looking for rigorous, academic-style learning, edX is an excellent choice. Check out this list of Python online courses with certificates to find the right fit for your learning goals. If youre looking for a budget-friendly way to start, there are many free online Python resources available. This guide on learning Python online for free covers the best options to help you get started without spending a dime. After choosing your learning path, its time to dive into the basics of Python. Start with exploring the most fundamental concepts: Creating variables: Learn how to store data like numbers, text, or other values in containers called variables for later use. For example, age = 25 (This stores the number 25 in a variable named age)Using variables: Explore how to access and manipulate the data youve stored in variables to perform calculations or make decisions in your code. For example, print(age + 5) (This adds 5 to the value stored in the age variable and prints the result)Formatting strings: Explore techniques to customize text output by inserting variable values or special characters into strings. For example, print(fYou are {age} years old.) (This prints You are 25 years old.)Comparing numbers and strings: Understand how to use operators like greater than, less than, and equal to to compare values and make your code respond accordingly. For example, if age >= 18: print(You are an adult.) (This prints You are an adult. only if the age is 18 or more)Types and comparisons: Learn about the different types of data (e.g., numbers, text) in Python and how to make sure youre comparing apples to apples, not apples to oranges. For example, 5 == 5. (This comparison is False because a number is not the same as a string containing the same digits)Input: Figure out how to get information from the user while your program is running, making it interactive and dynamic. For example, name = input(Whats your name?) (This asks the user for their name and stores it in the name variable) If you decide to take a Python course, the final list of essential topics will depend on the curriculum. For example, Mimos Intro to Python covers various themes, including working with variables, formatting strings, discovering types, and more. The best way to learn Python? Start building real-life projects right away. Getting too caught up in theoretical tasks often slows down the learning process. Instead, try applying new skills in practice. For example, each section in Mimos Python Career Path includes interactive projects like building a bot conversation. The guided tutorials come in small, actionable steps with increasing complexity to help you become proficient. If youre looking for more ideas for Python projects, weve compiled a list of some of the best beginner-friendly Python projects. After mastering the basics, its time to level up your Python game. Focus on the following intermediate concepts: Flow control: The backbone of decision-making and repetition in Python programs.Lists: Versatile collections of items (like numbers, words, or other objects) that can be ordered, changed, and accessed individually using their position or index.Functions: Reusable blocks of code that perform specific tasks. They help organize code and avoid repetition. Tuplesaresimilar to lists butcant be changed once you create them. Theyre perfect for storing information you know wont need to be modified, like a persons name, age, and job title. Dictionaries: They store information as key-value pairs. For example, you could have a dictionary where the keys are names and the values are phone numbers. You can look up a phone number by searching for the persons name. Setsarecollections of unique items that dont allow duplicates. This is handy if you want to ensure that your data doesnt contain repeated entries. Modules: Reusable pieces of code stored in separate files. They allow you to organize your code into manageable sections and reuse it across multiple projects. APIs: Interfaces that allow your program to interact with other software. Using APIs can enable your programs to leverage external services and data. String and Lists operations: Mastering these will help you become proficient at sorting items, searching within text data, and manipulating elements to generate desired results Object-oriented programming (OOP)is a way to organize computer programs by grouping data and actions together into reusable units called objects. For example, each character, item, and enemy in a video game is an object with its own properties and actions. When studying OOP, youmaster concepts like classes, inheritance, and polymorphism. Mimo has all those topics covered within the same Career Path: A well-crafted coding portfolio shows your abilities and helps you stand out to potential employers or clients.Start by adding each test project you complete. Include a description, the tools and technologies used, and the results achieved. Its important to provide context and highlight your problem-solving skills. There are many ways to create sample projects for your portfolio: Craft mini-projects: Create small, personal projects showcasing your coding skills. For example, you could create a website for a friend or a simple app. GitHub: Host your code on GitHub. It acts as your online resume, showing your problem-solving skills. Open source: You can also contribute to existing open-source projects and learn from experienced developers. Participate in online challenges: Test your skills and build small projects through online coding challenges. If youre learning with Mimo, you can simplify this task by using its portfolio builder. At the end of the Python Career Path, youll have eight ready-to-go projects for your portfolio. There are many advantages to joining a coding community. You can quickly find answers to your questions, learn from others experiences, and gain insights into solving common problems.Communities and forums to check out: Official Python forum: This is the central hub for all things Python, where you can ask questions, participate in discussions, and stay up-to-date on the latest developments. r/learnpython (Reddit): A subreddit dedicated to Python learners, with plenty of resources, tutorials, and support from fellow learners and experienced developers. PySlackers (Slack): An active Slack community for Python enthusiasts, with channels for various topics and skill levels. Stack overflow: A popular Q&A platform where you can ask questions and share solutions. Local Python user groups: Many cities have local Python user groups that hold meetings, workshops, and other events. Check the Python website for groups near you. Mimos community: If youre using Mimo, you can connect with the community to get guidance, share insights, and collaborate with your fellow learners. As we have already discovered, the Python job market offers many opportunities, and this trend will only continue to grow. Here are some roles you might come across: Web developer: Building and maintaining websites or web applications using a complex Python web framework like Django or Flask. Here, youll need a solid grasp of Python basics, HTML, CSS, and JavaScript. Data analyst: Collecting, cleaning, analyzing, and visualizing data using Python libraries like Pandas, NumPy, and Matplotlib. Go for this if you have strong analytical skills and familiarity with statistics. Software engineer: Developing and testing software applications using Python. This position requires a deep understanding of software development principles, algorithms, and data structures. Backend engineer: Building the server-side logic of web applications, handling databases, APIs, and security. Backend engineering requires in-depth knowledge of Python frameworks, databases, and cloud technologies. Data scientist: Using Python to extract insights from complex datasets, build predictive models, and apply machine learning techniques. Youll thrive better here if you have (or are willing to acquire) strong Python and data science skills, including working with libraries like scikit-learn and TensorFlow. Machine learning engineer: Designing, developing, and deploying machine learning models in production environments. To thrive, you need expertise in Python, machine learning algorithms, and cloud infrastructure. Python architect: Defining the technical architecture of Python-based systems, making critical decisions about technologies and frameworks. This role requires a deep understanding of the Python ecosystem and experience in system design. To find Python jobs, explore online job boards like LinkedIn, Indeed, Glassdoor, and SimplyHired. For freelance opportunities, use platforms like Upwork, Freelancer, and Toptal. Here are some of the resources you can use to learn Python faster. Corey Schafers YouTube Channel Programming with Mosh Lets address some of the most common questions about learning Python. Python is considered one of the easiest programming languages to learn, especially for beginners.It has a simple, readable syntax and a large, supportive community.Many concepts in Python are straightforward, making it a great first language for those new to coding. To get an entry-level Python job, you should understand the basics, like data types, control structures, functions, and object-oriented programming.Familiarity with common libraries (like NumPy or Pandas) and version control (Git) is also valuable.Make sure to build a portfolio of projects to showcase your skills to potential employers. Python skills can lead to well-paying jobs, especially in fields like data science, machine learning, and web development.Salaries vary based on experience, location, and specific role, but Python developers often earn competitive wages in the tech industry. For example, Python developers make an average of $124,016/year or $59.13/hour in the US as of January 2025. Entry-level Python developers can expect salaries of around $70,000 per year, depending on the company, location, and specific role. As you gain experience and skills, your earning potential increases. Keep in mind that salaries can vary significantly based on factors like the cost of living in different areas. At the same time, there are many remote Python developer jobs, meaning you could essentially work from anywhere. Neither Java nor Python is universally better each has strengths. Python is often easier to learn and read, making it great for beginners and rapid development. Java is widely used in enterprise environments and Android development.Choose based on your goals: Python for data science or scripting, Java for large-scale applications, or Android apps. To learn Python faster: Practice coding daily, even if just for short periods. Work on projects that interest you and keep you motivated. Use interactive coding platforms and join online communities to get feedback. Break down complex concepts into smaller, manageable parts. Use interactive platforms like Mimo that help you learn by doing. Python is a dynamic and popular coding language thats easy to pick up, making it a perfect choice for beginners. You can use it to build a career in various fields, from data science to engineering. To get started: List your learning goals. Figure out what motivates and excites you the most. Dive into the basics like variables, lists, and functions. Practice by creating small projects and keep exploring more complex topics. Join the Python community for help and inspiration. Create your portfolio and start applying for jobs. With dedication and the right resources, you can become a Python pro quickly. The job market for Python developers is hot right now, and learning it can open many doors. Ready to get started?Mimo can help you master Python and build a portfolio with practice-led and interactive lessons. Start for free today! Welcome! Are you completely new to programming?If not then we presume you will be looking for information aboutwhy and how to get started with Python.Fortunately anexperienced programmer in any programming language (whatever it may be)can pick up Python very quickly.It's also easy for beginners to use and learn, sojump in! Installing Python is generally easy, and nowadaysmany Linux and UNIX distributions include a recent Python.Evensome Windows computers (notably those from HP) now come with Pythonalready installed.If you do need to install Python and aren't confident about thetask you can finda few notes on theBeginnersGuide/Downloadwiki page, but installation is unremarkable on most platforms. If you want to know whether a particular application, or a librarywith particular functionality, is available in Python there are anumber of possible sources of information. The Python web siteprovides aPython Package Index(also known as the Cheese Shop, a reference to the Monty Pythonscript of that name).There is also asearch page for a number of sources of Python-relatedinformation. Failing that, justGoogle for a phrase including the word ''python''and you may well get the result you need.If all else fails, ask on thepython newsgroupand there's a good chance someone will put you on the right track. If you have a question, it's a good idea to try theFAQ, which answers the most commonlyasked questions about Python. If you want to help to develop Python, take a look at thedeveloper area for further information.Please note that you don't have to be an expert programmerto help. The documentation is just as important as thecompiler, and still needs plenty of work! Python is a high level, interpreted and general-purpose dynamic programming language that focuses on code readability. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management. It is one of the worlds most popular, in-demand programming languages. This is for many reasons: Its easy to learn. Its super versatile. It has a huge range of modules and libraries. The fact that it can support the vast majority of third-party modules, it acts as a cherry on the cake. There are some quite interesting modules that are considered worthy of sharing with others. In this article, some of the modules are discussed which will come in handy no matter if you are a beginner or a professional. As most of them are third party modules, they don't come in-built with Python and are needed to install. Installation of third party modules can be seen here. NOTE: Some of these code may not work in Python 2. Hence I strongly recommend you to try them in Python 3. This module was created to enable cross-platform copy-pasting in Python which was earlier absent. The pyperclip module has copy() and paste() functions that can send text to and receive text from your computer's clipboard. Sending the output of your program to the clipboard will make it easy to paste it on an email, word processor, or some other software. Pyperclip does not come with Python. To install it, follow the directions for installing third party modules. After installing the module, enter the following into IDE: Python3 # Python program to# demonstrate pyperclip module # This will import pyperclipimport pyperclippyperclip.copy("Hello world !")pyperclip.paste() pyperclip.copy("Isn't pyperclip interesting?")pyperclip.paste() Of course, if something outside your program changes the clipboard contents, the paste() function will return it. For example, if this sentence is copied to the clipboard and then paste() is called, the output would look like this: 'For example, if this sentence is copied to the clipboard and then paste() is called, the output would look like this:' Emojis have become a way to express and to enhance simple boring texts. Now, the same gems can be used in Python programs too. Yes, really! You now have the ultimate power to use emojis in your code. For this, emoji module is needed to be installed. In terminal. Use: pip install emoji To upgrade to the latest packages of emojis. Here's how it can be done: pip install emoji --upgrade Python3 from emoji import emojizeprint(emojize(":thumbs_up:")) Use the emoji cheat sheet to find your favorite emoji. Alternatively, encode() function can be used from emojis module to convert Unicode to emojis: Python3 import emojisemojified = emojis.encode("There is a :snake: in my boot !")print(emojified) Hope you try it! Stuck on a coding problem? Wish to visit StackOverflow without leaving the terminal? With howdoi, you can do it! Install the howdoi module by: pip install howdoi Or install from Python by: python setup.py install Ask whatever question you have and it will try it's best to answer it. Python3 howdoi make trees in Pythonhowdoi commit in gitFrom now, you don't need to open those browsers for a quick search and get those hefty amounts of ads and distractions. Just howdoi! Python3 1== howdoi use Howdoi in Python As if howdoi wasn't enough, we can now import the entire Wikipedia! Yes, We can now import Wikipedia in Python using Wikipedia module. Use the incessant flow of knowledge with Python for daily needs. Install it as: pip install wikipedia And use it as: Python3 import wikipediaresult = wikipedia.page("GeeksforGeeks")print(result.summary) If you wish to get a particular number of sentences from the summary, just pass that as an argument to the summary() function: Python3 import wikipediaprint(wikipedia.summary("Debugging", sentences = 2)) This can create new types in a fully dynamic way. It's the same as creating a class but something new which you can show to your friends. Python3 # Python program to# create new type object # Creates a new type objectNewType = type("NewType", (object,), {"attr": "hello newtype"})New = NewType() # Print the type of objectprint(type(New)) # Print the attribute of objectprint(New.attr) Output: hello newtype The above code is same as: Python3 # Creates a classclass NewType: attr = "hello newtype" # Initialize an objectNew = NewType() # Print the type of objectprint(type(New)) # Print the attribute of objectprint(New.attr) Output: hello newtype Probably not the best module but still worth a try! Ever wondered what python does under the hood? With the standard library module dis, you can look easily. Python3 # This will import# dis moduleimport dis def test(number): return (str(number)+str(number)) def newFunc(string): print("Hello", string) # This will display the# disassembly of test():dis.dis(test) # This will display the# disassembly of newFunc()dis.dis(newFunc) Output: Result: 8 0 LOAD_GLOBAL 0 (str) 3 LOAD_FAST 0 (number) 6 CALL_FUNCTION 1 (1 positional, 0 keyword pair) 9 LOAD_GLOBAL 0 (str) 12 LOAD_FAST 0 (number) 15 CALL_FUNCTION 1 (1 positional, 0 keyword pair) 18 BINARY_ADD 19 RETURN_VALUE 3 0 LOAD_GLOBAL 0 (print) 3 LOAD_CONST 1 ('Hello') 6 LOAD_FAST 0 (string) 9 CALL_FUNCTION 2 (2 positional, 0 keyword pair) 12 POP_TOP 13 LOAD_CONST 0 (None) 16 RETURN_VALUE That's overwhelming and also amazing! The reason this module is here is because this is quite fun! It's basically an easter egg in Python 3 which is used in Google App Engines. It was added to Google App Engines just as a medium to amuse the users. Install it with: pip install antigravity And then type this in your IDE to see the magic: Python3 1==This opens up a page in your web browser which contains a comical abstract of Python developed for your delight. Congratulations! You know have the ability to fly or for now the ability to visit this link . You may have used the sys module before but did you know you could exit your program early using it? We can cause the program to terminate by calling the sys.exit() function. Since this function is in the sys module, firstly, the sys module should be imported. This is not a third party module and comes built-in with Python so there is no need to install it. Python3 # This will import # sys moduleimport sys while True: print("Type 'exit' to exit") response = input() if response == "exit": print("Exiting the program") sys.exit() print("You typed", response) If the input is : "Geeky""GeeksforGeeks""exit" The output will be : Type 'exit' to exitYou typed GeekyType 'exit' to exitYou typed GeeksforGeeksType 'exit' to exitExiting the program Urllib module is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the urlopen function and is able to fetch URLs using a variety of different protocols. Urllib is a package that collects several modules for working with URLs, such as: urllib.request for opening and reading. urllib.parse for parsing URLs urllib.error for the exceptions raised urllib.robotparser for parsing robot.txt files Python3 1== # This will import urlopen# class from urllib modulefrom urllib.request import urlopen page = urlopen(")print(page.headers)The output would be: Server: ApacheStrict-Transport-Security: max-age=3600; includeSubDomainsAccess-Control-Allow-Credentials: trueX-Frame-Options: DENYX-Content-Type-Options: nosniffContent-Type: text/html; charset=UTF-8X-Akamai-Transformed: 9 - 0 pmb=mRUM,3Vary: Accept-EncodingCache-Control: must-revalidate, max-age=3, s-maxage=21600Date: Fri, 04 Oct 2019 04:57:37 GMTTransfer-Encoding: chunkedConnection: closeConnection: Transfer-EncodingServer-Timing: cdn-cache; desc=HITServer-Timing: edge; dur=1 You can also see the coding of the website by using read() function: Python3 # This will import urlopen# class from urllib module from urllib.request import urlopenpage=urlopen(") # Fetches the code # of the web pagecontent = page.read() print(content) Output: Yes, a turtle can be imported. Don't worry it's not slow. Turtle is a Python module to draw. It has a huge application and a number of methods which you can learn about in here. But with just a few basics, pretty cool stuff can be done. This module comes built-in with Python so there is no need to install it. Python3 1== # This will import turtle moduleimport turtle myTurtle = turtle.Turtle()myWin = turtle.Screen() # Turtle to draw a spiraldef drawSpiral(myTurtle, linelen): myTurtle.forward(linelen) myTurtle.right(90) drawSpiral(myTurtle, linelen-10) drawSpiral(myTurtle, 80)myWin.exitonclick() Output: Dev In this article, well explore the best Rust frameworks for web development, including Actix Web, Rocket, Axum, warp, Leptos, Cot, and Loco. Abiodun Solomon May 28, 2025 11 min read UX Design This post breaks down what design debt is and gives you 6 smart ways to fix it, without killing team velocity. Eric Chung May 28, 2025 6 min read Dev A single line of CSS can change how users feel about your UI. Learn how to leverage the cursor property to signal intent, improve interaction flow, and elevate accessibility. Chizaram Ken May 28, 2025 6 min read Dev We explore the benefits of building an app with React, TypeScript, and Vite, and compare its performance to the same app built with CRA. Clara Ekekenta May 28, 2025 7 min read Dev Implement a scheduling assistant using the OpenAI API and function calling, no Model Context Procedure (MCP) required. Kapeel Kokane May 28, 2025 11 min read Product Management Itzik Mitzmacher talks about how to avoid becoming a feature factory by shifting your product teams focus from outputs to outcomes. Sibel Bagcilar May 28, 2025 7 min read UX Design Here, youll learn what bottom sheets are, when and why theyre useful, and how to make them as usable and as accessible as possible. Daniel Schwarz May 27, 2025 5 min read UX Design Design system docs dont have to be a mess. Heres how to make yours clear, searchable, and worth using daily. Neel Dozome May 27, 2025 6 min read Dev Tight deadline? Bloated CSS options? This guide breaks down six modern CSS frameworks, from Beer CSS to CodeStitch, that help you build fast, beautiful UIs in 2025. Murat Yksel May 27, 2025 10 min read Product Management Canva offers hundreds of thousands of free templates and 100+ million design assets to appeal to all different types of users. Kimberly Shyu May 27, 2025 5 min read Product Management Shayani Roy talks about balancing innovation with speed, thoughtfulness, and commitment to solving customer problems. Jessica Srinivas May 27, 2025 7 min read UX Design Heres an in-depth look at shadows, some techniques for executing them, and how to create them in Figma with plugins. Chinwike Maduabuchi May 23, 2025 6 min read Tight deadline? Bloated CSS options? This guide breaks down six modern CSS frameworks, from Beer CSS to CodeStitch, that help you build fast, beautiful UIs in 2025. Murat Yksel May 27, 2025 10 min read Learn the ins and outs of Next.js middleware, which allows you to perform actions before a request is completed and modify the response accordingly. Temitope Oyedele May 23, 2025 9 min read The Google Gemini AI model has integrated AI-powered features to improve the debugging experience in web development. Emmanuel John May 23, 2025 5 min read The Node.js 24 release included significant updates. Explore 10 features you might not be using yet but absolutely should be. Emmanuel John May 22, 2025 8 min read Watch Now This tutorial has a related video course created by the Real Python team. Watch it together with the written tutorial to deepen your understanding: Variables in Python In Python, variables are symbolic names that refer to objects or values stored in your computers memory. They allow you to assign descriptive names to data, making it easier to manipulate and reuse values throughout your code. You create a Python variable by assigning a value using the syntax variable_name = value.By the end of this tutorial, youll understand that:Variables in Python are symbolic names pointing to objects or values in memory.You define variables by assigning them a value using the assignment operator.Python variables are dynamically typed, allowing type changes through reassignment.Python variable names can include letters, digits, and underscores but cant start with a digit. You should use snake case for multi-word names to improve readability.Variables exist in different scopes (global, local, non-local, or built-in), which affects how you can access them.You can have an unlimited number of variables in Python, limited only by computer memory.To get the most out of this tutorial, you should be familiar with Pythons basic data types and have a general understanding of programming concepts like loops and functions.Dont worry if you dont have all this knowledge yet and youre just getting started. You wont need this knowledge to benefit from working through the early sections of this tutorial. Take the Quiz: Test your knowledge with our interactive Variables in Python: Usage and Best Practices quiz. Youll receive a score upon completion to help you track your learning progress: Interactive Quiz Variables in Python: Usage and Best Practices In this quiz, you'll test your understanding of variables in Python. Variables are symbolic names that refer to objects or values stored in your computer's memory, and they're essential building blocks for any Python program. In Python, variables are names associated with concrete objects or values stored in your computers memory. By associating a variable with a value, you can refer to the value using a descriptive name and reuse it as many times as needed in your code.Variables behave as if they were the value they refer to. To use variables in your code, you first need to learn how to create them, which is pretty straightforward in Python.The primary way to create a variable in Python is to assign it a value using the assignment operator and the following syntax: In this syntax, you have the variables name on the left, then the assignment (=) operator, followed by the value you want to assign to the variable at hand. The value in this construct can be any Python object, including strings, numbers, lists, dictionaries, or even custom objects. Here are a few examples of variables: In this code, youve defined several variables by assigning values to names. The first five examples include variables that refer to different built-in types. The last example shows that variables can also refer to custom objects like an instance of your SomeCustomClass class.Apart from a variables value, its also important to consider the data type of the value. When you think about a variables type, youre considering whether the variable refers to a string, integer, floating-point number, list, tuple, dictionary, custom object, or another data type.Python is a dynamically typed language, which means that variable types are determined and checked at runtime rather than during compilation. Because of this, you dont need to specify a variables type when youre creating the variable. Python will infer a variables type from the assigned object.Note: In Python, variables themselves dont have data types. Instead, the objects that variables reference have types.For example, consider the following variables: In this example, name refers to the "Jane Doe" value, so the type of name is str. Similarly, age refers to the integer number 19, so its type is int. Finally, subjects refers to a list, so its type is list. Note that you dont have to explicitly tell Python which type each variable is. Python determines and sets the type by checking the type of the assigned value.Because Python is dynamically typed, you can make variables refer to objects of different data types in different moments just by reassigning the variable: Now, age refers to a string, and subjects refer to a set object. By assigning a value of a different type to an existing variable, you change the variables type.Variables are an essential concept in Python programming. They work as the building blocks for your programs. So far, youve learned the basics of creating variables. In this section, youll explore different ways to use variables in Python.Youll start by using variables in expressions. Then, youll dive into counters and accumulators, which are essential for keeping track of values during iteration. Youll also learn about other common use cases for variables, such as temporary variables, Boolean flags, loop variables, and data storage variables.In Python, an expression is a simple statement that Python can evaluate to produce and return a value. For example, consider the following expressions that compute the circumference of two different circles: Each of these expressions represents a specific computation. To build the expressions, youve used values and the multiplication operator (*). Python evaluates each expression and returns the resulting value. The above expressions are sort of rigid. For each expression, you have to repeat the input values, which is an error-prone and repetitive task.Now consider the following examples: In this example, you first define variables to hold the input values. Then, you use those variables in the expressions. Note that when you build an expression using variables, Python replaces the variable by its value. As shown in the example, you can conveniently reuse the values in different expressions.Another important point to note is that now you have descriptive names to properly identify the values used in the expressions.To summarize, variables are great for reusing values in expressions and running computations with data that varies over time. In general, variables let you name or label objects so that you can reference and manipulate them later in the program. In the following sections, youll see use cases of variables in practice.A counter is an integer variable that allows you to count objects. Counters typically have an initial value of zero, which increments to reflect the number of times a given object appears. To illustrate, say that you need to count the objects that are strings in a given list of objects. In this situation, you can do something like the following: In this example, you create the str_counter variable by initializing it to 0. Then, you run a for loop over a list of objects of different types. Inside the loop, you check whether the current object is a string using the built-in isinstance() function. If the current object is a string, then you increment the counter by one.At the end of the loop, str_counter has a value of 3, reflecting the number of string objects in the input list.Note: The highlighted line in the above example uses the expression str_counter += 1, which is a shortcut for str_counter = str_counter + 1. The += operator is known as the augmented addition operator.You reuse the expression str_counter += 1 in each iteration of the loop to increment the value of str_counter, making it change over time. This dynamic updating is a key feature of variables. As the name variables suggests, they are designed to hold values that can vary over time.Accumulators are another common type of variable used in programming. An accumulator is a variable that you use to add consecutive values to form a total that you can use as an intermediate step in different calculations.A classic example of an accumulator is when you need to compute the sum of numeric values: In this example, the loop iterates over a list of numbers and accumulates each value in total. You can also use accumulators as parts of larger computations, for example to calculate the mean of a list of numbers: Then, you take advantage of total to compute the average using the built-in len() function. You could have used an object counter instead of len(). Similarly, Python comes with several accumulator functions that you can often use instead of explicit accumulators. For example, you can use sum() instead of calculating total as above.Temporary variables hold intermediate results that you need for a more elaborate computation. A classic use case for a temporary variable is when you need to swap values between variables: In this example, you use a temporary variable called temp to hold the value of a so that you can swap values between a and b. Once youve done the swap, temp is no longer needed.Note: In Python, theres a clean and elegant way to swap values between variables without using temporary variables. Youll learn about this topic in the section on iterable unpacking.For a more elaborate example of using temporary variables, consider the following function that calculates the variance of a sample of numeric data: The expression that you use as the functions return value is quite involved and challenging to understand. Its also difficult to debug because youre running multiple operations in a single expression.To make your code easier to understand and debug, you can take advantage of an incremental development approach that uses temporary variables for intermediate calculations: In this alternative implementation of variance(), you calculate the variance in several steps. Each step is represented by a temporary variable with a meaningful name, making your code more readable.Boolean flags help you manage control flow and decision-making in your programs. As their name suggests, these variables can be either True or False. You can use them in conditionals, while loops, and Boolean expressions.Suppose you need to perform two different actions alternatively in a loop. In this case, you can use a flag variable to toggle actions in every iteration: Every time this loop runs, the conditional checks the value of toggle to decide which course of action to take. At the end of the loop, you change the value of toggle using the not operator. The next iteration will run the alternative action.Flags are also used as function arguments. Consider the following toy example: In this example, the verbose argument is a Boolean variable that lets you decide which greeting message to display.Youll find that a few Python built-in functions use flag arguments. The sorted() function is a good example: The sorted() function takes an iterable as an argument and returns a list of sorted objects. This function has a reverse argument that is a flag and defaults to False. If you set this argument to True, then you get the objects sorted in reverse order.In practice, youll find that Boolean variables are often named using the is_ or has_ naming pattern: In this example, the is_adult variable is a flag that changes depending on the value of age. Note that the is_ naming pattern allows you to clearly communicate the variables purpose. However, this naming convention is just a common practice and not a requirement or something that Python reinforces.Loop variables help you process data during iteration in for loops and sometimes in while loops. In a for loop, the variable takes the value of the current element in the input iterable each time you go through the loop: In this example, you iterate over a list of colors using a for loop. The loop variable, color, holds the current color in each iteration. This way, you can do something with the current item while you iterate over the data.Pythons for loops can have multiple loop variables. For example, say that you want to map each color with its index. In this case, you can do something like the following: In this example, you use the built-in enumerate() function to generate indices while you iterate over the input data. Note how this loop has two variables. To provide multiple loop variables, you use a comma-separated series of variables.You can also use variables to control while loops. Heres a toy example: In this example, the loop variable works as a counter that defines the number of iterations. Inside the loop, you use the augmented subtraction operator (-=) to update the variables value.Data storage variables allow you to work with containers of values, such as lists, tuples, dictionaries, or sets. For example, say that youre writing a contact book application that uses a list of tuples to store the information of your contacts: This contacts variable allows you to manipulate your data using a single and descriptive name. You can use the variables in a loop, for example: The first loop iterates over the items in the contact list and prints them as tuples. The second loop uses three loop variables to process every piece of data individually.Note that inside the loop, you can use the variables as needed. You dont need to use all the variables or use them in the same order. However, when you have an unused variable, then you should name it using an underscore to point out that its a throwaway variable.So, you could rewrite the loop header to look like this: for name, phone, _ in contacts:. In this case, the underscore represents the email variable, which you arent using in the body of the loop.The examples youve seen so far use short variable names. In practice, variable names should be descriptive to improve the codes readability, so they can also be longer and include multiple words.In the following sections, youll learn about the rules to follow when creating valid variable names in Python. Youll also learn best practices for naming variables and other naming-related practices.Variable names in Python can be any length and can consist of uppercase letters (A-Z) and lowercase letters (a-z), digits (0-9), and the underscore character (_). The only restriction is that even though a variable name can contain digits, the first character of a variable name cant be a digit.Note: Python currently has full Unicode support, and you can use many unicode characters in variable names. For example, the following variable names are valid: These variable names may be uncommon in Python code, but theyre completely valid. You can use them in code that performs scientific calculations when you want the code to reflect the notation used in the target discipline.All of the following are valid variable names: These variables follow the rules for creating valid variable names in Python. They also follow best naming practices, which youll learn about in the next section.The variable name below doesnt follow the rules: The variable name in this example starts with a number, which isnt allowed in Python. Therefore, you get a SyntaxError exception.Its important to know that variables are case-sensitive. Lowercase and uppercase letters arent treated the same: In this example, Python interprets the names as different and independent variables. So, casing is something to consider when youre creating variable names in Python.Nothing stops you from creating two different variables in the same program called age and Age, or, for that matter, agE. However, this practice isnt recommended because it can confuse people trying to read your code and even yourself after a while. In general, you should use lowercase letters when creating your variable names.The use of underscore characters is also significant. Youll use an underscore to separate multiple words in a variable name: In these variable names, you use the underscore character as a separator for multiple words. This is a way to improve the readability of your code by substituting the space character with an underscore. To illustrate this, consider how your variables would look without the underscores: Even though these names are technically valid, they can be challenging to read and understand at a glance. The lack of separation makes it harder to grasp the meaning of each variable quickly, and they require more effort to interpret. Using underscores improves the clarity of your code and makes it more maintainable.You should always give a variable a descriptive name that clearly explains the variables purpose. Sometimes,

you can find a single word to name a given variable: Variables always refer to concrete objects, so their names should be nouns. You should try to find specific names for your variables that uniquely identify the referred object. Names like variable, data, or value may be too generic. While these names can work for short examples, theyre not descriptive enough for production code.In general, you should avoid single-letter names: Single-letter names may be hard to decipher, making your code difficult to read, especially when you use them in expressions with other similar names. Of course, there are exceptions. For example, if youre working with nested lists, then you can use single-letter names to identify indices: Its common to use letters like i, j, and k to represent indices, so you can use them in the right context. Its also common to use x, y, and z to represent point coordinates, so these are also okay to use.Using abbreviations to name variables is discouraged, in favor of using the complete name: Its best practice to use a complete name instead an abbreviated name because its more readable and clear. However, sometimes abbreviations are okay when theyre widely accepted and used: In these examples, cmd is a commonly used abbreviation for command and msg is commonly used for message. A classic example of a widely used abbreviation in Python is the cls name, which you should use to identify the current class object in a class method.Sometimes, you need multiple words to build a descriptive variable name. When using multi-word names, you can struggle to read them if there isnt a distinguishable boundary between words: This variable name is difficult to read. You have to pay close attention to figuring out the words boundaries so that you can understand what the variable represents.The most common practices for multi-word variable names are the following:Snake case: Lowercase words are separated by underscores. For example: number_of_graduates.Camel case: The second and subsequent words are capitalized to make word boundaries easier to see. For example: numberOfGraduates.Pascal case: Similar to camel case, except the first word is also capitalized. For example: NumberOfGraduates.The Style Guide for Python Code, also known as PEP 8, contains naming conventions that list suggested standards for names of different object types. Regarding variables, PEP 8 recommends using the snake case style.When you need multi-word names, its common to combine an adjective as a qualifier with a noun: In these examples, you create descriptive variable names by combining adjectives and nouns, which can dramatically improve your codes readability. Another point to consider is to avoid multi-word names that start with my_, like my_file, my_color, and so on. The my_ part doesnt really add anything useful to the name.Flag variables are another good example of when to use a multi-word variable name: In these examples, youve used an underscore to separate the words, making their boundaries visible and quick to spot.When it comes to naming lists and dictionaries, you should use plural nouns in most situations: Using plural nouns in these examples makes it clear that the variable refers to a container that stores several objects of similar types.When naming tuples, you should consider that theyre commonly used to store objects of different types or meanings. So, its okay to use singular nouns: Although these tuples store multiple objects, they represent a single entity. The first tuple represents an RGB (red, green, blue) color, while the second represents a row in a database table or some other tabular data.A widely used naming convention for variables in Python is to use a leading underscore when you need to communicate that a given variable is what Python defines as non-public. A non-public variable is a variable that shouldnt be used outside its defining module. These variables are for internal use only: Copied! In this module, you have a non-public variable called _timeout. Then, you have a couple of functions that work with this variable. However, the variable itself isnt intended to be used outside the containing module.Python reserves a small set of words known as keywords that are part of the languages syntax. To get the list of Pythons keywords, go ahead and run the following code: In most cases, you wont be able to use these words as variable names without getting an error: This behavior is true for most keywords. If you need to use a name that coincides with a keyword, then you can follow PEP 8s recommendation, which is to add a trailing underscore to the name: In this example, youve added an underscore character at the end of the keyword, which enables you to use it as a variable name. Even though this convention works, sometimes its more elegant to do something like the following: Now, your variables name is way more descriptive and specific, which improves your codes readability.There are also soft keywords, which are keywords only in specific contexts. For example, the match keyword is only considered a keyword in structural pattern matching.Because match is a soft keyword, you can use it to name variables: In this example, you import the re module to use regular expressions. This example only searches for a basic expression in a target text. The idea here is to show that match can be used as a valid variable name even though its a keyword.Another practice that you should avoid is using built-in names to name your variables. To illustrate, say that youre learning about Python lists and run the following code: In this example, youve used list as the name for a list object containing numeric values. This shadows the original object behind the name, which prevents you from using it in your code: Now, calling list() fails because youve overridden the built-in name in your code. So, youre better off avoiding built-in names to define your variables. This practice can make your code fail in different ways.Note: For a complete list of built-in names, run the following code in an interactive session: The list of built-in names is quite long. To make sure that your variable name doesnt shadow one of the names in this list, you can do something like "name" in dir(builtins). If this check returns True, then its best to find a different name.Note that this recommendation is also valid for names that refer to objects defined in third-party libraries that you use in your code.When you start digging deeper into how Python variables work internally, you discover several interesting features worth studying. In the following sections, youll explore some of the core features of variables so that you can better understand them.What happens when you create a variable with an assignment? This is an important question in Python because the answer differs from what youd find in many other programming languages.Python is an object-oriented programming language. Every piece of data in a Python program is an object of a specific type or class. Consider this code: When presented with the statement 300, Python does the following operations:Creates an integer objectGives it a value of 300Displays it on the screenYou can see that an integer object is created using the built-in type() function: A Python variable is a symbolic name that refers to or points to an object like 300. Once an object is assigned to a variable, you can refer to it by the variables name, but the data itself is still contained within the object.For example, consider the following variable definition: This assignment creates an integer object with a value of 300 and makes the variable n point to that object. The diagram below shows how this happens:Variable Assignment In Python, variables dont store objects. They point or refer to objects. Every time you create an object in Python, its assigned a unique number, which is then associated with the variable.The built-in id() function returns an objects identifier or identity: In CPython, the standard Python distribution, an objects identity coincides with its memory address. Therefore, CPython variables store memory addresses. Through these memory addresses, variables access the concrete objects stored in memory.You can create multiple variables that point to the same object. In other words, variables that hold the same memory address: In this example, Python doesnt create a new object. It creates a new variable name or reference, m, which points to the same object that n points to:Multiple References to a Single Object Next, suppose you do something like this: Now, Python creates a new integer object with the value 400, and m becomes a reference to it:References to Separate Objects Finally, say that you run the following statement: Now, Python creates a string object with the value "foo" and makes n a reference to that:Orphaned Object Because of the n and m reassignments, you no longer have a reference to integer object 300. Its orphaned, and you have no way to access it again.When the references to an object drop to zero, the object is no longer accessible. At that point, its lifetime is over. Python reclaims the allocated memory so it can be used for something else. In programming terminology, this process is known as garbage collection.In many programming languages, variables are statically typed, which means theyre initially declared to have a specific data type during their lifetime. Any value assigned to that variable during its lifetime must be of the specified data type.Python variables arent typed this way. In Python, you can assign values of different data types to a variable at different moments: In this example, youre making the value variable refer to or point to an object of another type. Because of this feature, Python is a dynamically typed language.Its important to note that changes in a variables data type can lead to runtime errors. For example, if a variables data type changes unexpectedly, you may face type-related bugs or even get an exception: In this example, the variable type changes during the codes execution. When value points to a string, you can use the .upper() method to convert the letters into uppercase. However, when the type changes to float, the .upper() method isnt available, and you get an AttributeError exception.You can use type hints to add explicit type information to your variables. To do this, you can use the following Python syntax: The square brackets arent part of the syntax. They denote that the enclosed part is optional. Yes, you can declare a Python variable without assigning it a value: The variable declaration on the first line works and is valid Python syntax. However, this declaration doesnt really create a new variable for you. Thats why when you try to access the number variable, you get a NameError exception. Even though number isnt defined, Python has recorded the type hint: When it comes to basic data types such as numbers and strings, type hints may look superfluous: If youre familiar with Pythons built-in types, then you wont need to add type hints to these variables because youll soon know that you have a string, integer, and floating-point value, respectively. So, in this situation, youre okay to skip the type hint. Also, your static type checker or linter wont complain.The story changes when you use collection types, such as lists, tuples, dictionaries, and sets. With these types, it makes sense to provide type hints for their contained data.To illustrate, say that you have the following dictionary of colors: In this case, itd be great to have information about the data types of keys and values. You can provide that information using the following type hint: In this updated code, youre explicitly communicating that your colors dictionary will store its keys and values as strings.Why is this type hint important? Say that in another part of your code, you have another color dictionary defined as shown below: At some point, you might be working with both dictionaries and wont have an unambiguous way to know which dictionary is needed. To work around the issue, you can add a type hint to the second dictionary as well: The type hint in this example is a bit more elaborate, but it clearly states that your dictionary will have string keys and tuples of three integers as values.Its important to note that type-hinting variables that refer to container data types is especially useful when you initialize the variable with an empty container: In these examples, you have two empty lists. Because they have type hint information, you can quickly figure out the data type of the content. Now you know that the first list will contain strings, while the second list will hold tuples, each consisting of three items: a string, an integer, and another string value.Note that in this example, your static type checker or linter will explicitly complain about the type hint of your lists if you dont provide any type information.Later in your code, you can append items to each list using the correct data type: First, you use the .append() method to add new fruits as strings to the end of fruits, and then you add new rows to the end of rows.In Python, youll find a few alternative ways to create new variables. Sometimes, defining several variables simultaneously with the same initial value is convenient or needed. To do this, you can use a parallel assignment.In other situations, you may need to initialize several variables with values from a sequence data type, like a list or tuple. In this case, you can use a technique called iterable unpacking.Youll also find situations where you need to retain the value that results from a given expression. In this case, you can use an assignment expression.In the following sections, youll learn about all these alternative or complementary ways to create Python variables.Python also allows you to run multiple assignments in a single line of code. This feature makes it possible to assign the same value to several variables simultaneously: The parallel assignment in this example initializes three different but related variables to False simultaneously. This way of creating and initializing variables is more concise and less repetitive than the following: By using parallel assignments instead of dedicated assignments, you make your code more concise and less repetitive.Iterable unpacking is a cool Python feature also known as tuple unpacking. It consists of distributing the values in an iterable into a series of variables. In most cases, the number of variables will match the number of items in the iterable. However, you can also use the *variable syntax to grab several items in a list.You can use iterable unpacking to create multiple variables at a time using an iterable of values. For example, say that you have some data about a person and want to create dedicated variables for each piece of data: If you didnt know about iterable unpacking, then your first approach might be to distribute the data into different variables manually, as shown below: This code works. However, using indices to extract the data may lead to an error-prone and hard-to-read result. Instead of using this technique, you can take advantage of iterable unpacking and end up with the following code: Now, your code looks cleaner and more readable. So, when you find yourself creating variables from iterables using indices, consider using unpacking instead.A great use case for unpacking is when you need to swap the values between two variables: In the highlighted line, you swap the values of a and b without using a temporary variable, as you saw before. In this example, its important to note that the iterable to the right of the equal sign is a tuple of variables.Assignment expressions allow you to assign the result of an expression used in a conditional or a while loop to a name in one step. For example, consider the following loop that takes input from the keyboard until you type the word "stop": This loop works as expected. However, this code has the drawback that it unnecessarily repeats the call to input().You can rewrite the loop using an assignment expression and end up with the following code: In the expression (line := input("Type some text: ")), you create a new variable called line to hold a reference to the input data. This data is also returned as the expressions result, which is finally compared to the "stop" word to finish the loop. So, assignment expressions are another way to create variables.To learn more about assignment expressions, check out The Walrus Operator: Pythons Assignment Expressions.The concept of scope defines how variables and names are looked up in your code. It determines the visibility of a variable within your code. The scope of a variable depends on the place in your code where you create the variable.In Python, you may find up to four different scopes, which can be presented using the LEGB acronym. The letters in this acronym stand for local, enclosing, global, and built-in scopes. In the following sections, youll learn the basics of variable scopes in Python and how this concept can affect the way you use your variables.Global variables are those that you create at the module level. These variables are visible within the containing module and in other modules that import them. So, for example, if youre using the Python REPL, then the current global scope is defined in a module called __main__.This module defines your current global scope. All the variables defined in this module are global, so you can use them at anytime during an interactive session run: In this code snippet, you define the value variable and call the built-in dir() function to check the list of names defined in your current global scope. At the end of the list, youll find the 'value' entry, which corresponds to your variable.Now, say that you use this same interactive session to run several pieces of code while you try out some cool features of Python. After all those examples, you need to use the value variable again: Youll notice that the variables will still be available for you. Thats because this value variable is global to your code.You can also import variables defined outside your current module by using the import statement. A typical example of this practice is when you have a module that defines a variable to hold some configuration parameters. You can import this variable into your global scope and use it as youd use any other global variable.As a quick example, say that you have the following config.py module: This file defines a dictionary that contains several configuration parameters for a hypothetical application. You can import this variable to your apps global scope and use the settings parameters as needed: With the import in the first line, youve brought the settings variable into your current global scope. You can use the variable from any part of your code.Local variables are those that you define inside a function. These variables can help you store the results of intermediate computations under a descriptive name. They can make your code more readable and explicit.Consider the following example: Local variables are only visible within their defining function. Once the function returns, the variables disappear. Thats why you cant access integer in the global scope.Similarly, non-local variables are those that you create in a function that define inner functions. The variables local to the outer function are non-local to the inner functions. Non-local variables are useful when youre creating closure functions and decorators. Heres a toy example that illustrates how global, local, and non-local variables work and how to identify the different scopes in Python: In this example, you first create a global variable at the module level. Then, you define a function called outer_func(). Inside this function, you have nonlocal_variable, which is local to outer_func() but non-local to inner_func().In inner_func(), you create another variable called local_variable, which is local to the function itself.Note: You can access global and non-local variables from within an inner function. However, to update a global or non-local variable from within an inner function, you need to explicitly use the global and nonlocal statements.The calls to print() are intended to show how you can access variables from different scopes inside a function.Heres how this function works: In summary, global variables are accessible from every place in your code. Local variables are visible in their defining function. Non-local variables are visible in their defining or enclosing function and in any inner function that lives in the enclosing function.You can create variables inside custom Python classes when using object-oriented programming tools. These variables are called attributes. In practice, you can have class and instance attributes.Class attributes are variables that you create at the class level, while instance attributes are variables that you attach to instances of a given class. These attributes are visible only from within the class or its instances. So, classes define a namespace, which is similar to the scope. To illustrate how class and instance attributes work, say that you need a class to represent your companys employees. The class should keep information about the employee at hand, which you can store in instance attributes like .name, .position, and so on. The class should also keep a count of how many employees are currently in the company. To implement this feature, you can use a class attribute.Heres a possible implementation of your class: Copied! In this Employee class, you define a class attribute called count and initialize it to 0. Youll use this attribute as a counter to keep track of the number of Employee instances. Class attributes like .count are common to the class and all its instances.Inside the initializer, .__init__(), you define three instance attributes to hold the name, position, and salary of the employee. The last line of code in .__init__() increments the counter by 1 every time you create a new employee. To do this, you access the class attribute on the class object itself.Note: You cant change the value of a class attribute using the self object. For example, if you do something like self.count += 1 instead of Employee.count += 1, then you create a new instance attribute that shadows the class attribute. You can alternatively access the class attribute with type(self).count instead of Employee.count. Finally, you have a method to display the employees profile according to their current information. This method shows that you need to use the self argument to access instance attributes within a class.Heres how you can use this class in your code: In this code, you first create two instances of Employee by calling the class constructor with appropriate arguments. Then, you call the display method on both employees and get the corresponding information. Finally, you access the .count attribute on Employee to get the current number of employees.Its important to note that you can access instance attributes like .name or .position using the dot notation on the target instance: Instance attributes are specific to one instance, so you cant access them through the class. In contrast, class attributes are common to the class and all its instances: To access a class attribute, you can either use an instance or the class itself. However, to change a class attribute directly, you need to use the class. Go ahead and give it a try with the following example: In this example, you try to update the count attribute using an instance, john. This action results in you attaching a new instance attribute to john instead of updating the value of Employee.count. To update the class attribute, you need to use the class itself.In Python, you can explicitly remove variables or, more generically, names from a given scope using the del statement: In this code snippet, you first create a new variable called city in your current global scope. Then, you use the del statement to remove the variable from its containing scope. When you try to access the variable again, you get a NameError exception. You should know that while del removes the reference to an object, it doesnt necessarily free the memory immediately. Pythons garbage collector claims the memory once there are no more references to the object.You now know the fundamentals of using variables in Python, including how to create and use them in your code. Youve learned that Python variables can point to objects of different data types at different moments, which makes Python a dynamically typed language.Youve also learned to use variables in expressions and other common use cases like counters, accumulators, and Boolean flags. Additionally, youve explored best practices for naming variables.In this tutorial, youve:Created variables and assigned values to themChanged a variables data type dynamicallyUsed variables in expressions, counters, accumulators, and Boolean flagsFollowed best practices for naming variablesCreated, accessed, and used variables in their specific scopesWith these skills, you can now confidently manage data in your Python programs, write readable and maintainable code, and apply best practices to ensure the quality of your code. Take the Quiz: Test your knowledge with our interactive Variables in Python: Usage and Best Practices quiz. Youll receive a score upon completion to help you track your learning progress: Interactive Quiz Variables in Python: Usage and Best Practices In this quiz, you'll test your understanding of variables in Python. Variables are symbolic names that refer to objects or values stored in your computer's memory, and they're essential building blocks for any Python program. Now that you have some experience with Python variables, you can use the questions and answers below to check your understanding and recap what youve learned.These FAQs are related to the most important concepts youve covered in this tutorial. Click the Show/Hide toggle beside each question to reveal the answer.In Python, variables are symbolic names for objects or values stored in memory. They let you assign meaningful names to data, making it easier to manipulate and reuse values in your code.You set a variable in Python by assigning a value to a name using the assignment operator (=), placing the variable name on the left and the value you want to assign on the right. For example, number = 42 assigns the value 42 to the variable name number.Yes, you can change a variables data type by assigning it a new value of a different type. Python is dynamically typed, so it determines the type at runtime.Python itself doesnt assign types to variables, but the objects that variables reference have types, and Python infers the type of a variable from the value assigned to it.You can use variables to store data for expressions, counters, accumulators, Boolean flags, and more.The four rules for naming Python variables are: they must start with a letter or an underscore, can only contain letters, digits, and underscores, theyre case-sensitive, and cant be a reserved keyword.Use descriptive and meaningful names, follow the snake case convention for multi-word names, and avoid using keywords or built-in names.The scope of a variable determines where in your code you can access it. Variables can have local, non-local, global, or built-in scope.In Python, theres no fixed limit to the number of variables you can have, as it largely depends on the available memory and resources of your computer. Watch Now This tutorial has a related video course created by the Real Python team. Watch it together with the written tutorial to deepen your understanding: Variables in PythonDSA to Development: A Complete GuideBeginner to AdvanceJAVA Backend Development - LiveIntermediate and AdvanceTech Interview 101 - From DSA to System Design for Working ProfessionalsBeginner to AdvanceFull Stack Development with React & Node JS - LiveBeginner to AdvanceJava Programming Online Course [Complete Beginner to Advanced]Beginner to AdvanceC++ Programming Course Online - Complete Beginner to AdvancedBeginner to AdvancePage 2

How easy is python. How to learn python fast and easy. Is python easy to learn for beginners. How easy is it to learn javascript after python. Is python hard to learn for beginners. How easy is it to learn python after r. How easy is it to learn python after java. How easy is it to learn c++ after python. Does python is easy to learn.