# Hello Python!

In this module we’ll introduce you to the Coursera platform and the course format. Then, we’ll dive into the basics of programming languages and syntax, as well as automation using scripting. We’ll also introduce you to the Python programming language and some of the benefits it offers. Last up, we’ll cover some basic functions and keywords of the language, along with some arithmetic operations.

Learning Objectives

Define the terms computer program, programming language, script, and automation

Use the print() function to output data to the screen

Explain the difference between the syntax and semantics of a programming language

List some of the characteristics of the Python language

Utilize basic Python arithmetic operators to obtain the results of mathematical expressions

## **Course Introduction**

### Specialization Introduction

Working in IT is more than just a job. It's a career path. Research shows that the field of IT support is a launchpad for future career growth and better wages. In fact, a study on the subject was recently conducted by the Harvard Business School Accenture and Burning Glass entitled Bridge the Gap. It found that among today's middle-skilled jobs which require training but not a formal college degree, IT support offers clear pathways to prosperity. We saw this phenomenon play out here at Google, in our IT support program. Those who push themselves to learn how to code in Python typically saw strong career growth. They built skills that are critical to accessing higher level positions in the IT field, and after honing those skills through hard work and determination, they advanced into more technical IT support specialists, systems administrators, technical solutions engineers, and even site reliability engineers. The common thread across all of these roles is knowing how to write code to solve problems and automate solutions. By expanding your toolbox to include coding skills, you open a window into the world of systems management that can lead you towards more advanced technical roles down the line. Python in particular is having a huge resurgence. According to the 2019 Stack Overflow Developer Survey, Python is the coding language most people want to learn. The second most loved by those who already know it and the fourth most popular overall. So why take this program to learn how to code in Python? Well, first it's geared towards people who are already in or aspiring to be in the field of IT. Maybe you're thinking bigger about your current IT role and want to work towards managing operations at scale, or maybe you're just starting out and looking to break into the IT industry. Perhaps you've taken our IT Support Professional Certificate program on Coursera already, or you have equivalent IT support knowledge with basic computing skills, like working with files and directories, familiarity with networking concepts, and understanding how to install software on your computer. In any case, this program is tailor-made for you. Second, this program offers three hands-on methods of teaching coding and Python and automation, code blocks, Jupyter notebooks, and Qwiklabs. Third, we've assembled an awesome group of Googlers who will serve as your instructors in each course. They all started their careers in IT support then learned programming and moved onto more technical roles like me. We can't wait to share our stories with you on how we use Python in our day-to-day life. Oh and hey, I should probably introduce myself. My name is Christine Rafla, and I'm a Systems Administrator at Google. I will be your instructor in this course. This program has been entirely designed and developed by Google and we even filmed each course at different cool Google spaces. It will introduce you to the Python programming language with a special focus on how this language applies to automating tasks in the world of IT systems support and administration. I'm super excited to be here with you. When I was younger, I had no idea that careers in IT even existed. There are a lot of reasons I wanted to participate in the certificate program. But one of my biggest motivations is that I want to see more women represented in the industry. I remember going to a System Administration Summit where there were hundreds of men and about three women that were sysadmins. A lot has changed since then but there's still so much we can do to bring new ideas and representation into the IT field. That's why I want to share my knowledge with as many people as possible. I love my job and I love the people I work with because they make it easy to ask for help and offer their guidance. This type of support network allows our team and ultimately our industry to be more successful. I understand from experience that it can feel pretty intimidating and maybe even a bit scary to learn a coding language. Just remember, everyone started where you are right now with the first command, the first script, and of course, the first of many errors. When I started out in my career, I strived to get everything perfectly right the first time I tried it. But that actually slowed down my progress. So don't be afraid to make mistakes, it will give you a leg up. So let's get down to it. What's ahead? The program begins with a crash course in Python where you will learn to write simple programs and understand their role in automation. Next, we'll get more hands-on focus on how Python interacts with the operating system. After that, we'll cover how to use Git and GitHub to manage versions of your code. Then we will focus on troubleshooting and debugging techniques to find and solve the root cause of problems in IT infrastructure. The next course covers automating at scale where you will learn to deploy configuration management on a fleet of either physical or virtual machines running in the Cloud. Last up, we will bring all this knowledge together and complete a final project designed to solve tasks that you might encounter in real-world IT settings. Bonus, you can post your projects to GitHub to show off your fancy new skills to employers or friends or both. That was a lot to rattle off. Are you excited or what? Now I'd like to quickly introduce you to my fellow instructors who you will get to meet along the way. Hey, my name is Roger Martinez. I'm a Linux System Administrator, and I'll be your instructor in the course on using Python's interacts with the operating system. Hi. I'm Kenny Sulma, and I'll be your guide in the course about using Git and GitHub to manage versions of your code. Hi there. I'm Amanda Ballas, and I'll be teaching you about troubleshooting and debugging. Hey, I'm Phelan Vendeville, and in my course, we'll learn about Automation at scale using Configuration Management and the Cloud. Thanks everyone. This all-star team was brought together to guide you on your adventures in coding. You're in very good hands. Okay, I think that's everything. Let's get ready to learn some new skills and maybe even have some laughs along the way. I'll see you in the next video. In this module we’ll introduce you to the Coursera platform and the course format. Then, we’ll dive into the basics of programming languages and syntax, as well as automation using scripting. We’ll also introduce you to Python programming...

## 

### Course Introduction

If you work in IT, computer programming skills open up an incredible amount of opportunity. Being able to write scripts and programs that tell your computer to perform a task equips you with an invaluable tool. Not only does it make your work easier and more efficient, it can help you grow faster and advance further in your IT career. But how do you even start to learn a programming language like Python? How do you recognize when to tell a computer to perform a task? And how do you then write a program to actually get your computer to complete the task you want it to do? The thought of learning to write a program in Python can make you feel a whole bunch of emotions, excitement, anticipation, that feeling of wanting to dive right in and get going and also fear. You might ask yourself, can I really learn how to code or do I have it in me? I'm here to tell you, yes, you can absolutely do this. Learning how to program can be scary and intimidating, but at the same time it's really fun and really exciting. In coding as in life, if we're going to get philosophical, the most rewarding work is usually a bit challenging, but ultimately well worth the effort. Of course, I'm able to say all this from experience, especially the cheesy parts. My name is Christine Rafla. I'm a systems administrator at Google and I'm going to be your instructor and guide in this course. The role of a sysadmin can vary a lot from company to company and even within different teams in the same company. I happen to work in the corporate identity and access management operations team, which is a really long way of saying that we make sure that everyone is represented correctly and if they need to access certain resources, they can. What I love the most about being a sysadmin is that the role has so many diverse functions. We handle loads of unique problems and edge cases from tinkering with different systems to collaborating with other teams. I am always learning something new, so it's really hard to get bored. It all starts with knowing how to automate, if you're an IT support specialist, a systems administrator, or in a role somewhere in between knowing how to get computers to do the hard work for you will set you apart from others in similar IT roles and make your life much easier. Think about it, would you rather manually deploy 100 computers on your own or tell your computer to do it all for you all at once? No-brainer, right, having a coding skill set can help you grow into more specialized roles like a systems administrator, Cloud Solutions engineer, DevOps specialist, site reliability engineer, or who knows maybe even web developer or data analyst. The point is, being able to write a program is an essential tool in your IT toolkit and more and more employers are looking for these skills in the people they hire. If you've ever learned a new skill, like playing a musical instrument, speaking a foreign language, knitting, or skateboarding. You know that getting good at something new requires a lot of practice. For me, I love to learn new languages and I'm proud to say I speak Spanish, Arabic, French, and I even know ten words in Russian. Our world is shaped by the words and the languages we speak and while some words may be unique to one language you can always find similarities that help you learn and understand. Being able to connect the dots between cultures allows me to see things others might not, kind of sounds like this applies to IT programming, huh? My point is, whether you're learning French or Python, it's never easy. You have to start small, learn the basics and practice those until you master them. Only then can you move on to more complex and impressive stuff and I'm here to help you along that path along with my colleagues who you'll meet in later courses. We'll start slow, master the foundations together and you'll soon be ready for more challenging stuff. So are you wondering why we filmed this course in a cabin on a lake in Canada. The truth is we're actually in a game room at one of the Google offices in Sunnyvale, California. We chose a different themed office space for each course of the program just to mix things up and I think I scored with this one. I should warn my manager that I'll be hanging out in this one well after the course ends because it's super comfy. By the end of this course, you'll understand the benefits of programming in IT roles. You'll be able to write simple programs using Python, figure out how the building blocks of programming fit together, and combine all this knowledge to solve a complex programming problem. That's right, by the end of this course, you're going to write a program in Python that's designed to solve a real-world IT problem, super exciting right? We'll start off by diving into the basics of writing a computer program. You'll get hands-on experience with programming concepts through interactive exercises and real-world examples. You'll quickly start to see how computers can perform a multitude of tasks. You just have to write code that tells them what to do. Along the way, we'll be talking about automation, which is the process of getting computers to automatically do a task that us humans normally have to do by hand. Now, some of the stuff can get a little complex and confusing. I promise to do my best to make these lessons clear and easy to understand, but if you get stuck at any point, please feel free to re-watch the videos. Practice as much as you like and take the time you really need to understand these topics. The goal of this course isn't to teach you everything there is to know about software engineering because yikes, that would be a long course. Instead, we're going to introduce you to some of the key concepts of programming and scripting that will empower you to spot opportunities for automation in real life. You're about to learn a skill that can help you take your career to whole new levels. Are you excited? I'm excited, so let's jump in!

### Program Surveys

### 

### Google Cert Participant Entry Survey

### 

### Take a minute to Set Yourself up for Success

### 

### Welcome to the Course!

## **Introduction to Programming**

### The Beginning of Your Programming Journey

As the Chinese proverb says, a journey of a thousand miles begins with a single step. Today's a big day, you're taking your first step in your journey to learn how to write scripts in Python. It's going to be a little challenging at times, but really it's not that scary. We'll go slow and give you everything you need to fully grasp each concept before we move along. In the next few videos, you'll discover the fundamental concepts of computer programming. You'll learn what programming language is, what scripting is, what languages are out there other than Python, and how this all relates to IT. We'll also have you coding before you know it with small coding exercises we've cooked up to give you hands-on practice with Python. This will include writing your very first Python script. But always keep in mind, if at any point along the way you feel lost or confused, don't panic. You can watch thevideos as many times as you need to let the concept sink in, plus you can ask questions in the discussion forums, which is one of the best ways to find extra information and connect with other learners. When I was asked to participate in this program, it made me think about whenI first started to code. If I could give that younger version of myself a piece of advice, this is what I would tell her, it never works the first time. Seriously, as a newbie, I expected it all to work like magic. I thought that following the rules and getting it right the first time would prove my value as a coder, but that's just not true, not even the best of the best. If you expect to write perfect code on the first shot you're going to be disappointed. You hear that younger self? Try not to feel overwhelmed by the details. Connecting the dots only comes with experience, so the best way to learn is to just jump in. The truth is everyone learns at their own pace. If you already know some of these concepts, feel free to skip ahead to the parts that interest you the most. If you're starting from scratch, take as long as you need for each concept. The assessments will be right there waiting for you when you're done, and if at any point you start doubting yourself, remember, even the most advanced programmers started thinking, Python? What's Python? Well, we're about to learn all about it, so let's dive in. Next up, we'll be doing a rundown of what programming is.

## 

### What is programming?

At a basic level, a computer program is a recipe of instructions that tells your computer what to do. When you write a program, you create a step by step recipe of what needs to be done to complete a task and when your computer executes the program it reads what you wrote and follows your instructions to the letter. How nice is that? The recipe is written in a code called programming language. Programming languages are actually similar to human spoken languages since they have a syntax and semantics. Now if it's been awhile since our last grammar class, here's a quick refresher on syntax and semantics. In a human language, syntax is the rules for how a sentence is constructed while semantics refers to the actual meaning of the statements. In English, sentences generally have both a subject, that's a person, place, or thing and a predicate, usually a verb and a statement that explains what the subject is doing. Let's take the sentence, Paula loves to program in Python as an example. In this sentence, Paula is the subject and loves to program in Python Is the predicate. To form a sentence that others can understand, you need to know both the syntax that constructs the sentence and the semantics that gives it meaning. The same applies to programming languages. In a programming language like Python, the syntax is the rules for how each instruction is written and the semantics are the effects the instructions have. Much like spoken languages, there are lots of programming languages to choose from. Each has its own history, features, and applications but they all share the same fundamental ideas. So once you understand the basic concepts in one programming language, it becomes much easier to learn another. Lastly, computers always do exactly what they're told. So when you write a program, it's important to be super clear about what you want the computer to do. Learning the syntax and semantics of the programming language you choose will allow you to do just that. Make sense? Before we continue, let's spend a moment on terminology. In the next few videos you'll hear the term script being used a bunch. So what's the difference between a script and a program? The line between the two can be a bit blurry. In this course, we'll use the terms interchangeably. In general, you can think of scripts as programs with a short development cycle that can be created and deployed rapidly. In other words, a script isa program that is short, simple, and can be written very quickly. In this course we'll focus on a specific scripting language called python which we'll use to learn the basics of programming. We'll learn about the python syntax, the rules of how to write a python program, and the semantics or meaning of the different pieces involved. Before we start learning how to code and having you write your first python script, let's talk more about what automation is and why it's useful.

Question

Did you get all that? Let's check with a quick question!

Why do we need to learn the syntax and semantics of a programming language?

* ~~To be able to easily switch to a different programming language~~
* ~~So that we know which part is the subject and which one is the predicate~~
* To allow us to clearly express what we want the computer to do
* ~~To understand why our computer crashes~~

Correct

You nailed it! Knowing the syntax and understanding the semantics of a programming language allows us to tell the computer what we want it to do.

### What is automation?

Although we might not realize it, we reap the benefits of automation all the time in our daily lives. Do you ever pay your bills with scheduled payments or use a self check out at the grocery store? I always set my coffee machine to start brewing before I've even gotten out of bed. The promise of fresh coffee makes early mornings way easier. Automation is the process of replacing a manual step with one that happens automatically. Take a traffic light for example, which continuously regulates the flow of vehicles at an intersection. A traffic light requires human intervention only when it needs repairs or maintenance. The automatic regulation of traffic means that humans don't have to stand at the intersection manually signaling why cars should stop or go. Instead, people can concentrate on more complex, creative, or difficult tasks like focusing on where you're driving. What's more,traffic lights don't get tired, bored, or accidentally display a greenlight when they mean red. This highlights another benefit of automation consistency. Let's face it, us humans are flawed and sometimes we make mistakes, a human performing the same tasks hundreds of times will never be as consistent as a machine doing the same thing. But for all of its advantages automation isn't a solution for every situation, some tasks just aren't suited for automation. For example, they may require a degree of creativity or flexibility that automatic systems can't provide or for more complicated or less frequently executed tasks creating the automation may actually be more effort or cost than it's worth. Think about when you get a haircut. What would it take to automate the actions of cutting hair with a machine? The client's height,the shape of their head, their current hair length,and desired hairstyle would all need to be taken into account when designing the automatic system. We need to replicate the creativity and skills of a trained specialist along with extensive testing to ensure the client's safety and quality haircut. And if you've ever had a bad experience at a hair salon, you know quality can be subjective. In this case, the cost and effort of automation just isn't worth the benefits an automatic haircut would provide which is why we don't have robot hairstylists. Not too complex, right? Automation is a powerful tool when used in the right place at the right moment. It can save time, reduce errors,increase consistency, and provide a way to centralize solutions and mistakes, making them easier to fix. Throughout this course, and in upcoming ones we'll be talking about when it makes sense to apply automation and exactly how you do it. Eventually knowing when and where to use automation will become automatic for you.

Question

Let's check that this all made sense with a quick question!

What’s automation?

* ~~The process of telling a computer what to do~~
* ~~The process of installing traffic lights~~
* ~~The process of getting a haircut~~
* The process of replacing a manual step with one that happens automatically

Correct

Right on! By replacing a manual step with an automatic one we create automation that helps us reduce unnecessary manual work.

### Uses for Automation

Scripts can be used for automating specific tasks. Automation is used to replace a repetitive manual step with one that happens automatically. Humans are fallible. They can become tired, make mistakes, fail to follow instructions, be inconsistent in their job performance, and more. In contrast, automated processes complete instructions exactly as coded, in a consistent manner. They can run 24 hours a day, everyday, without tiring. For many tasks that are appropriate for automation, it can be more cost effective to use automation than human labor.

Appropriate uses for automation include:

The automatic timing and regulation of traffic lights

A repetitive task that is at high risk for human error

Sending commands to a computer

Detecting and removing duplicates of data

Sending automated emails that are personalized by pulling individual names from a database and plugging them into the email

Updating a large number of file permissions

Reporting on system data, like disk or memory usage

Installing software

Generating reports

Deploying a file or a computer program to all computers on a company network

Using a configuration management system to deploy software patches, after a human has designed the system

Populating an e-commerce site with products

Setting the home directory and access permissions for users

Automation is not always an appropriate or complete solution

Automation cannot perform all human work. Tasks that call for human creativity, social connection, psychology, flexibility, ingenuity, evaluation, and/or complex analytic work are not good candidates for full automation. Sometimes automation can be used to perform one or more subtasks of a larger set of tasks – but – human intervention is required to complete the tasks. The following are some examples of tasks that cannot or should not be fully automated:

Items that require human evaluation and analytic skills:

Designing a configuration management system

Investigating and troubleshooting all end user problems

Writing a computer program

Building a new startup business

Items that require human creativity and/or an eye for aesthetic qualities:

Designing an attractive webpage (AI can do this, but simple automation cannot)

Wedding photography

Haircuts and styling

Items that cannot be automated due to basic physics:

Troubleshooting or repairing machines that cannot power on or boot up

Items that need human interaction, psychology, and/or evaluation skills:

Interviewing and hiring new employees

Customer service (chat bots cannot address every customer service need)

Items that should not be fully automated due to costs and safety:

Grocery store checkout process, including bagging groceries

Tasks that are less expensive to perform manually

Artificial Intelligence

It is important to understand that basic automation is not the same as artificial intelligence. Automation is used to explicitly instruct a machine on how to perform a task. Artificial intelligence (AI) involves training a computing machine to perform more complex tasks through a process called machine learning. This process prepares the AI software to perform new tasks without a human needing to program explicit instructions for each task. Although AI is often used for automating human tasks, AI automation is much more complex than basic automation.

### Getting Computers to Work for You

Working in IT, a lot of what we do boils down to using a computer to perform a certain task. In your job you might create user accounts, configure the network, install software, backup existing data, or execute a whole range of other computer based tasks from day to day. Back in my first IT job, I realized that every day I came into work I typed the same three commands to authenticate into systems. Those credentials timed out everyday by design, for security reasons, so I created a script that would automatically run these commands for me every morning to avoid having to type them myself. Funny enough, the team that monitors anomalous activity discovered my little invention and contacted me to remove it, oops. Tasks performed by a computer that need to be done multiple times with little variation are really well suited for automation, because when you automate a task you avoid the possibility of human errors, and reduce the time it takes to do it. Imagine this scenario: your company had a booth at a recent conference and has gathered a huge list of emails from people interested in learning more about your products. You want to send these people your monthly email newsletter, but some of the people on the list are already subscribed to receive it. So how do you make sure everyone receives your newsletter, without accidentally sending it to the same person twice? Well, you could manually check each email address one by one to make sure you only add new ones to the list, sounds boring and inefficient, right? It could be, and it's also more error prone, you might accidentally miss new emails, or add emails that were already there, or it might get so boring you fall asleep at your desk. Even your automated coffee machine won't help you out there. So what could you do instead? You could get the computer to do the work for you. You could write a program that checks for duplicates, and then adds each new email to the list. Your computer will do exactly as it's told no matter how many emails there are in the list, so it won't get tired or make any mistakes. Even better, once you've written the program you can use the same code in future situations, saving you even more time, pretty cool, right? It gets better, think about when you're going to send these emails out, if you send them out manually you'll have to send the same email to everybody, personalizing the emails would be way too much manual work. If instead you use automation to send them, you could have the name and company of each person added to the email automatically. The result? More effective emails, without you spending hours inserting names into the text. Automating tasks allows you to focus on projects that are a better use of your time, letting computers do the boring stuff for you. Learning how to program is the first step to being able to do this. If you want to get computers to do the work for you, you're in the right place. Earlier in this video I told you about the first task I ever automated, now I want to tell you about the coolest thing I ever automated. It was a script that changed a bunch of access permissions for a whole lot of Google Internal Services. The script traversed a large directory tree with tons of different files, checked the file contents, and then updated the permissions to the services based on the conditions that I laid out in the script. Okay, I admit I'm a total nerd, but I still think it's really cool. Next up, it's time to share your ideas. What things would you like to automate using programming? While these discussion prompts are optional, they're really fun. Seriously, they let you get to know your fellow learners a bit, and collaborate on ideas and insights. Make sure you read what others are saying, they may give you ideas that you haven't even thought of. After that, you're ready to take your very first quiz of the course. Don't worry, it's just for practice.

Question

Which of the following tasks do you think are good candidates for automation? Check all that apply.

Investigating reports that customers are having difficulty accessing your company's external website

This should not be selected

Not quite. Investigating reports requires manual work, as each instance of a problem might have different root causes.

Installing software on laptops given to new employees when they are hired

Correct

Right on! Installing and configuring software is a task that can be automated. Ensuring that everyone gets the exact same setup and reducing the amount of manual work needed for each new employee.

Designing a configuration management system for deploying software patches

This should not be selected

Not quite. While configuration management systems are a great tool for automation, designing such a system requires creative work which is best left to humans.

Periodically scanning the disk usage of a group of file servers

Correct

You nailed it! Scanning the disk usage is a task that can be easily automated. By letting the computer do it, you won't have to worry about forgetting to do it whenever it's needed.

### Study Guide: Introduction to Programming

Your first practice quiz is coming up soon. This handy study guide should help you prepare for that quiz. The practice quizzes do not count towards your grade in this course. Practice quizzes are opportunities for you to check your understanding of the materials before you take the graded assessments at the end of each module.

Key Terms

Programming code - Programming code is a set of written computer instructions, guided by rules, using a computer programming language. It might help to think of the computer instructions as a detailed, step-by-step recipe for performing tasks. The instructions tell computers and machines how to perform an action. Programming code may also be referred to as source code or scripts.

Programming languages - Programming languages are similar to human spoken languages in that they both use syntax and semantics. Programming languages are used to write computer programs. Some common programming languages include Python, Java, C, C++, C#, and R.

Syntax - Syntax is a set of rules for how statements are constructed in both human and computer languages. Programming syntax includes rules for the order of elements in programming instructions, as well as the use of special characters and their placements in statements. This concept is similar to the syntax rules for grammar and punctuation in human language.

Semantics - Semantics refers to the intended meaning or effect of statements, or collections of words, in both human and computer languages. Semantic errors are also referred to as logical errors.

Computer program - A computer program is a step-by-step list of instructions that a computer follows to reach an intended goal. It is important to be clear and precise about the actions a computer program is supposed to perform because computers will do exactly what they are instructed to do. Computer programs can be long, complex, and accomplish a variety of tasks. They are often developed by computer programmers and software engineers, but anyone can learn to create them. Computer programs may involve a structured development cycle. They can be written in a wide variety of programming languages, such as Python, Java, C++, R, and more. The completed format of a program is often a single executable file.

Script - Scripts are usually shorter and less complex than computer programs. Scripts are often used to automate specific tasks. However, they can be used for complex tasks if needed. Scripts are often written by IT professionals, but anyone can learn to write scripts. Scripts have a shorter, less structured development cycle as compared to the development of complex computer programs and software. Scripts can be written in a variety of programming languages, like Python, Javascript, Ruby, Bash, and more. Some scripting languages are interpreted languages and are only compatible with certain platforms.

Automation - Automation is used to replace a repetitive manual step with one that happens automatically.

Output - Output is the end result of a task performed by a function or computer program. Output can include a single value, a report, entries into a database, and more.

Input - Input is information that is provided to a program by the end user. Input can be text, voice, images, biometrics, and more.

Functions - A function is a reusable block of code that performs a specific task.

Variables - Variables are used to temporarily store changeable values in programming code.

### Practice Quiz: Introduction to Programming

Your grade: 100%

Your latest: 100%•

Your highest: 100%•

To pass you need at least 80%. We keep your highest score.

Question 1

* ~~What’s a computer program?~~
* ~~A set of languages available in the computer~~
* ~~A process for getting duplicate values removed from a list~~
* A list of instructions that the computer has to follow to reach a goal

A file that gets copied to all machines in the network

Correct

You nailed it! At a basic level, a computer program is a recipe of instructions that tells your computer what to do.

Question 2

What’s the syntax of a language?

The rules of how to express things in that language

The subject of a sentence

The difference between one language and another

- The meaning of the words

Correct

Right on! In a human language, syntax is the rules for how a sentence is constructed, and in a programming language, syntax is the rules for how each instruction is written.

Question 3

What’s the difference between a program and a script?

* There’s not much difference, but scripts are usually simpler and shorter.
* ~~Scripts are only written in Python.~~
* ~~Scripts can only be used for simple tasks.~~
* ~~Programs are written by software engineers; scripts are written by system administrators.~~

Correct

You got it! The line between a program and a script is blurry; scripts usually have a shorter development cycle. This means that scripts are shorter, simpler, and can be written very quickly.

4.

Question 4

Which of these scenarios are good candidates for automation? Select all that apply.

- Generating a sales report, split by region and product type

Correct

Excellent! Creating a report that presents stored data in specific ways is a tedious task that can be easily automated.

Creating your own startup company

Helping a user who’s having network troubles

- Copying a file to all computers in a company

Correct

Nice work! A task like copying files to other computers is easily automated, and helps to reduce unnecessary manual work.

Interviewing a candidate for a job

Investigating the root cause of a machine failing to boot

5.

Question 5

What are semantics when applied to programming code?

The rules for how a programming instruction is written

The difference in number values in one instance of a script compared to another

- The intended meaning or logic of coded statements

The end result of a programming instruction

Correct

Nice job! The intended meaning or effect of coded statements are referred to as semantics.

## **Introduction to Python**

### What is Python?

Welcome back. How did you do on your first quiz? If you got most of the questions right, great job. If not, no worries it's all part of learning. We'll be here to help you check that you've really got your head around these concepts with regular quizzes like this. If you ever find a question tricky, go back and review the videos and then try the quiz again. You want to feel super comfortable with what you've learned before jumping into the next lesson. Remember, take your time. I will be here whenever you're ready to move on. Okay. Feeling good? Great. Let us dive in. In this course, we will use the Python programming language to demonstrate basic programming concepts and how to apply them to writing scripts. We have mentioned that there are a bunch of programming languages out there. So why pick Python? Well, we chose Python for a few reasons. First off, programming in Python usually feels similar to using a human language. This is because Python makes it easy to express what we want to do with syntax that's easy to read and write. Check out this example.

Execute the following code and see what happens. Feel free to change it and run it as many times as you want.

friends = ['Taylor', 'Alex', 'Pat', 'Eli']

for friend in friends:

print("Hi " + friend)

Hi Taylor

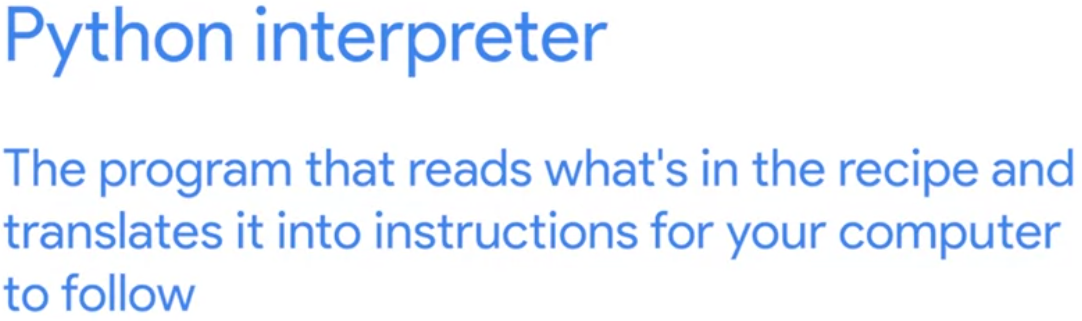
Hi Alex

Hi Pat

Hi Eli

There is a lot to unpack here so don't worry if you don't understand it right away, we'll get into the nitty-gritty details later in the course. But even if you've never seen a line of code before, you might be able to guess what this code does. It defines a list with names of friends and then creates a greeting for each name in the list. Now it is your turn to make friends with Python. Try it out and see what happens. Throughout this course, you will execute Python code using your web browser.

We'll start with some small coding exercises using code blocks just like the one you experimented with. Later on as you develop your skills, you'll work on larger more complex coding exercises using other tools. Getting good at something takes a whole lot of practice and programming and Python is no different. We recommend that you practice every example we share in this course on your own. If you do not have Python installed on your machine, no worries, you can still practice using an online Python interpreter. Check out the next reading for links to the most popular Python interpreters available online. Now I am sure you are wondering what the heck is a Python interpreter.



In programming, an interpreter is the program that reads and executes code. Remember how we said a computer program is like a recipe with step-by-step instructions? Well, if your recipe is written in Python, the Python interpreter is the program that reads what is in the recipe and translates it into instructions for your computer to follow. Eventually, you'll want to install Python on your computer so you can run it locally and experiment with it as much as you like. We'll guide you through how to install Python in the upcoming course but you don't have to have it installed to get your first taste of Python. You can practice with the quizzes we provide and with the online interpreters and code pads that we'll give you links to in the next reading. We'll provide a whole bunch of exercises for you but feel free to come up with your own and share them in the discussion forums. Feel free to get creative. This is your chance to show off your new skills.

### Python Resources

* More About Python **Using Python on your own**The best way to learn any programming language is to practice it on your own as much as you can. If you have Python installed on your computer, you can execute the interpreter by running the python3 command (or just python on Windows), and you can close it by typing exit() or Ctrl-D.  
  If you don’t already have Python installed on your machine, that’s alright. We’ll explain how to install it in an upcoming course.  
  **Python practice resources**In the meantime, you can still practice by using one of the many online Python interpreters or codepads available online. There’s not much difference between an interpreter and a codepad. An interpreter is more interactive than a codepad, but they both let you execute code and see the results.  
  Below, you’ll find links to some of the most popular online interpreters and codepads. Give them a go to find your favorite.
  + <https://www.python.org/shell/>
* <https://www.onlinegdb.com/online_python_interpreter>
* <https://repl.it/languages/python3>
* <https://www.tutorialspoint.com/execute_python3_online.php>
* <https://rextester.com/l/python3_online_compiler>
* <https://trinket.io/python3>
* **Additional Python resources**While this course will give you information about how Python works and how to write scripts in Python, you’ll likely want to find out more about specific parts of the language. Here are some great ways to help you find additional info:
  + Read the [official Python documentation](https://docs.python.org/3/)
* .
* Search for answers or ask a question on [Stack Overflow](https://stackoverflow.com/)
* .
* Subscribe to the Python [tutor](https://mail.python.org/mailman/listinfo/tutor)
* mailing list, where you can ask questions and collaborate with other Python learners.
* Subscribe to the [Python-announce](https://mail.python.org/mailman/listinfo/python-announce-list)
  + mailing list to read about the latest updates in the language.
* **Python history and current status**Python was released almost 30 years ago and has a rich history. You can read more about it on the [History of Python](https://en.wikipedia.org/wiki/History_of_Python) Wikipedia page or in the section on the [history of the software](https://docs.python.org/3.0/license.html) from the official Python documentation.  
  Python is a rapidly evolving language, with new versions released regularly to introduce fresh features, enhance performance, address security vulnerabilities, and improve overall usability. While these updates drive innovation, they can sometimes require adjustments to existing code. Fortunately, Python offers tools to facilitate the migration process. For this course, we will be using Python 3.7.To stay up to date on the most current release of Python or find a specific release, you can visit [Python's website](https://www.python.org/downloads/).  
  Additionally, if you would like to learn more about Python and its history you can visit the following websites:
  + [The Incredible Growth of Python](https://stackoverflow.blog/2017/09/06/incredible-growth-python/)
* (Stack Overflow)
* [Why is Python Growing So Quickly - Future Trends](https://www.netguru.com/blog/why-python-is-growing-so-quickly-future-trends)
* (Netguru)
* [Python Trends](https://technobrains.io/watch-out-for-python-trends-2023/)
* (TechnoBrains)
* [Developer Survey Results 2018](https://insights.stackoverflow.com/survey/2018#technology)

(Stack Overflow)

### A Note on Syntax and Code Blocks

When writing code, using correct syntax is critical. Even a small typo, like a missing parenthesis bracket or an extra comma, can cause a syntax error and the code won't execute at all. If your code results in an error or an exception, pay close attention to syntax and watch out for minor mistakes. A single wrong character could take hours to identify in long code so it is important to be mindful of syntax when writing code.

#### Common syntax errors:

* Misspellings
* Incorrect indentations
* Missing or incorrect key characters:
  + Bracket types - ( curved ), [ square ], { curly }
  + Quote types - "straight-double" or 'straight-single', “curly-double” or ‘curly-single’
  + Block introduction characters, like colons - :
* Data type mismatches
* Missing, incorrectly used, or misplaced Python reserved words
* Using the wrong case (uppercase/lowercase) - Python is a case-sensitive language

If your syntax is correct, but the script has unexpected behavior or output, this may be due to a semantic problem. Syntax is like the vocabulary, grammar, spelling, and punctuation of code. Semantics are the meaning and logic of coded statements. It is possible to have syntactically correct code that runs successfully, but doesn't do what we want it to do.

#### Common semantic errors:

* Creating functional code, but getting unintentional output
* Poor logic structures in the design of the code

When working with the code blocks in exercises for this course, be mindful of syntax and semantic (logic) errors, along with the overall result of your code. Just because you fixed an error doesn't mean that the code will have the desired effect when it runs! Once you’ve fixed an error in your code, don't forget to click Run to check your work.

## Why is Python relevant to IT?

Remember how we mentioned that Python is simple and easy to use? Python makes it easy to express the fundamental concepts of programming like data structures and algorithms with easy to read syntax. This makes Python a great language to use to learn programming. And there are other reasons to pick Python, too.

Python is super popular in the IT industry, making it one of the most common programming languages used today.

Python isn't new. Its first version was released by Guido van Rossum back in 1991. Since then, the community that develops it has grown and the language has advanced a lot. Whenever there's a significant change to the semantics or syntax of the language, a new major version is released.

In 2000, Python 2 was released. In 2008, we got Python 3. In this course, we'll use Python 3.7, which came out in 2018. For many years, Python was considered a beginner's language and was mostly used for teaching concepts or writing very small simple scripts, like in this course. But in recent years, the adoption of Python has grown dramatically. One reason for this is that the language has become more powerful. It's also because there's more tools available in Python for a growing range of applications. You can use Python to calculate statistics, run your e-commerce site, process images, interact with web services, and do a whole host of other tasks.

Python is perfect for automation. It lets you automate everyday tasks by writing simple scripts that are easy to understand and easy to maintain. That's why Python is the language of choice for lots of people working in IT support, system administration, and web development. Not only that, but it's also used in fast-growing areas of IT, like machine learning or data analytics.

Last but not least, Python is available for download on a wide variety of operating systems, like Windows, Linux, and Mac OS. And what's more, Python is so popular in the workplace that if you are currently working in IT, you've most likely encountered it already. And if you're planning a career in IT, chances are you'll interact with Python quite a bit.

So there's a whole lot of reasons for why Python is relevant to today's IT industry.

Question

Let's check whether you soaked all that in with a quick question! Select all options that explain why Python is relevant to today’s IT industry.

Python scripts are easy to write, understand, and maintain.

Correct

Woohoo! Python is a language that tries to mimic our natural language and so Python scripts are generally easy to write, understand and maintain.

There are many system administration tools built with Python.

Correct

You got it! Over the years, the Python community has developed a lot of additional tools that can be used by system administrators to get their job done.

Python was written by Guido van Rossum in 1991.

Python is available on a wide variety of platforms.

Correct

Well done, you! Python is available on Windows, Linux, MacOS and even on mobile devices, making it a great tool for IT specialists looking to create scripts that can work across platforms.

There have been multiple major version releases over the years which incorporate significant changes to the language.

A large part of programming is learning through trial and error and asking questions. So if at any point you get stuck, don't get discouraged. Making mistakes helps you improve. The more you see failure or broken code as an opportunity to learn, the quicker you'll master programming. I remember the first Python script I ever wrote. It took a lot of refactoring, debugging, and testing to get it to work. I relied on a lot of my teammates for help and mentorship and wound up spending more time on Stack Overflow than actually writing the code. Thankfully, you don't have to reinvent the wheel. There's almost always someone on the Internet who's trying to do what you're doing and can help point you in the right direction when you're stuck. Sometimes it takes a village. It's really important to keep in mind that even experienced programmers may need to ask a colleague a question from time to time or look something up on the Internet. Whether you're a programming novice or have some experience in software development, remember, the best programmers overcome challenges by seeking help or using other resources. Once you've completed this program, you'll be well on your way to confidently programming in basic Python. There's lots of information online that will help you continue to develop your programming skills. For example, there are lots of online courses for specific programming languages. You'll find answers to your Python coding questions in the official Python documentation. You can use sites like Stack Overflow to discuss and share with other developers. And you can ask questions in our discussion forums. You can even subscribe to some of the Python mailing lists to keep in the know on the latest updates to the language. You're opening the door to the whole world of programming, and it's super exciting to be joining the development community. The most important thing to remember is that you're never alone. Any questions you may have, any time in your career, there are resources out there to help you find the answers you need. Wow, that was a lot of information. Feel free to take a quick break, grab something to drink, and then head on over to the supplemental reading to learn more about Python and the resources out there to help you learn.

### Other Languages

Although we picked Python for this course, it's important to note that it's just one of the many coding languages out there. Think of a given programming language as just one of the many powerful tools in your IT toolbox. Each language has its unique set of pros and cons. Some run faster than others. Some are better suited for enterprise applications. Others are particularly good at crunching numbers. There are platform-specific scripting languages like PowerShell which is used on Windows, and Bash which is used on Linux. Both are widely used by system administrators on those platforms. There are also general-purpose scripting languages similar to Python, like Perl or Ruby, which are also widely used for scripting and automation. JavaScript, which was originally developed as a client-side scripting language for the web, is increasingly used server-side for a broader set of tasks. And the list doesn't stop there. There's a vast array of traditional languages to explore like C, C++, Java, or Go. As you progress in your career in IT, you'll probably encounter a number of different languages and learn when to use each of them. But let's not get ahead of ourselves. First, we have Python to get our heads around. A nice feature of learning the basics of programming in one language is that you can generally apply the same concepts you learn to other languages. This means that once you're familiar with Python, you'll find it easier to pick up new coding languages as you'll spot and understand similarities and differences between them. After all, every language needs to do some common things like create variables, control the flow of a program, read input, and display output, even if they do these tasks using different approaches. As we called out earlier, learning a programming language is somewhat similar to learning a foreign language. You'll need to grasp the syntax and semantics for that language. Luckily for us, once you know the fundamentals of programming, learning another language is much easier than learning a second foreign language. There are a lot more similarities between programming languages than differences. To explore some of the similarities and differences between various scripting languages, let's take a look at a simple program that prints the words hello world ten times in three different languages, Python, Bash, and PowerShell.



As you can see, each language uses a different approach to printing hello world. But look closer and you'll see similarities too. Each language must somehow put text onto the screen. The command for Python is print, for Bash it's Echo, and for PowerShell it's Write-Host. Also notice that each language has to count to ten in some way. While Python does this by specifying range(10), Bash uses a sequence notation to count from 1 to 10. PowerShell has the most complex syntax in this example, but it also boils down to starting at 1 and counting up to 10. So as we've just seen there's a whole lot of programming languages out there, but don't let that scare you. In this course, you will only need to focus on learning Python. Once you can speak Python you can go on to learn any other language you want. Up next, we've got another quiz to help you practice what you've just learned.

### Study Guide: Introduction to Python

* This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.   
  In this segment, you learned that Python is a general purpose programming language that is commonly used for scripting and automation, as well as to develop a wide variety of applications. Python is compatible with most operating systems, including Windows, Linux, and Mac OS, and is updated every few years. Python can also run on a variety of machines, such as servers, workstations, PCs, mobile devices, IoT, and more.  
  Python is widely used in the IT field, including IT support, system administration, web development, machine learning, data analytics, and more. Python can be used to calculate statistics, run your e-commerce site, process images, interact with web services, and do a whole host of other tasks. Python instructions resemble the English language, which is what makes it easier to learn and understand when compared to other programming languages.  
   **Python is:**
  + a general purpose scripting language;
  + a popular language used to code a variety of applications;
  + a frequently used tool for automation;
  + a cross-platform compatible language;
  + a beginner-friendly language.
* **Python is not:** 
  + a platform-specific / OS-specific scripting language;
  + a client-side scripting language;
  + a purely object-oriented programming language.
* **Code comparison with Python**You will be learning about both Python and Bash scripting in this program. The following code illustrates a syntax difference between the two languages:

| **Print to screen in Python** | **Print to screen in Bash** |
| --- | --- |
| *>> print("Hello, how are you?")*  *Hello, how are you?* | *>> echo Hello, how are you?*  *Hello, how are you?* |

* Key Terms
  + **Platform-specific / OS specific scripting language** - Platform-specific scripting languages, like PowerShell (for Windows) and Bash (for Linux), are used by system administrators on those platforms.
  + **Client-side scripting language** - Client-side scripting languages, like JavaScript, are used mostly for web programming. The scripts are transferred from a web server to the end-user’s internet browser, then executed in the browser.
  + **Machine language** - Machine language is the lowest-level computer language. It communicates directly with computing machines in binary code (ones and zeros). In binary code, one equals a pulse of electricity and zero equals no electrical pulse. Machine language instructions are made from translating languages like Python into complex patterns of ones and zeros.
  + **Cross-platform** **language** - Programming language that is compatible with one or more platforms / operating systems (e.g., Windows, Linux, Mac, iOS, Android).
  + **Object-oriented programming language** - In object-oriented programming languages, most coding elements are considered to be objects with configurable properties. For example, a form field is an object that can be configured to accept only dates as input in the mm/dd/yy format, and can be configured to read from and write to a specific database.
  + **Python interpreter -** An interpreter is the program that reads and executes Python code by translating Python code into computer instructions.

### Practice Quiz: Introduction to Python

Question 1

In the code block below, replace \_\_\_ with the Python command to display “My first Python program” on the screen after the command is run.

print("My first Python program")

Correct

Way to go! You've given the computer the correct instruction in Python to do what you’ve asked of it.

### 

Question 2

Python is an example of what type of programming language?

* ~~Machine language~~
* ~~Platform-specific scripting language~~
* Client-side scripting language
* ~~General purpose scripting language~~

Incorrect

Not quite. Client-side scripting languages like JavaScript are used mostly for web programming. Python is used for a wide variety of applications.

### 

Question 3

The code block below contains a Bash command to display “Have a nice day” on the screen.Convert this Bash command to a Python command.

print("Have a nice day")

Correct

Great job! You used the correct syntax, with proper punctuation, to make this code work in Python.

### 

Question 4

In the code block below, replace \_\_\_ with the Python command that will display “This is fun!” on the screen 5 times.

for i in range(5):

print("This is fun!")

Correct

Well done! You remembered how to use the "print" command to do the work.

Question 5

Why is Python relevant to IT? Select all that apply.

* Python is used in fast-growing areas of IT, like machine learning and data analytics.

Correct

Correct! Python is a popular tool used in the IT fields of machine learning and data analytics.

* Python works well as a scripting language for IT automation.

Correct! Python lets you automate repetitive IT tasks by writing simple scripts that are easy to understand and easy to maintain.

* ~~Python scripts run on IT servers only.~~
* Python can be used to calculate statistics, run e-commerce sites, process images, interact with web services, and more.

Correct

Correct! Python is very versatile and can be used for a wide variety of IT-related tasks.

## Hello World

### Review: Hello, World!

This reading contains the code used in the instructional videos from [**Hello, World!**](https://www.coursera.org/learn/python-crash-course/lecture/C5koy/hello-world)

## **Introduction**

This follow-along reading is organized to match the content in the video that follows. It contains the same code shown in the next video. You will find additional information throughout the guide to explain the purpose of each concept covered, why the code is written in a certain way, and tips for running the code. These code blocks will provide you with the opportunity to see how the code is written, allow you to practice running it, and can be used as a reference to refer back to.

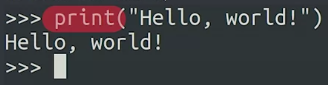
You can follow along in the reading as the instructor discusses the code or review the code after watching the video.



### Hello World!

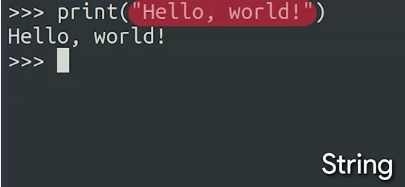
Now that you've got an idea of what Python code looks like, let's check out one of the most basic examples and dive deeper into what's going on. Get ready. We're going to use the Python interpreter to make our computer say hello to the world.

When we run this code either locally on our machine or on a web interpreter, the words hello world appear on the screen, just like magic. Actually it's not magic. It's because Print is a Python function that writes what we tell it to on the screen.

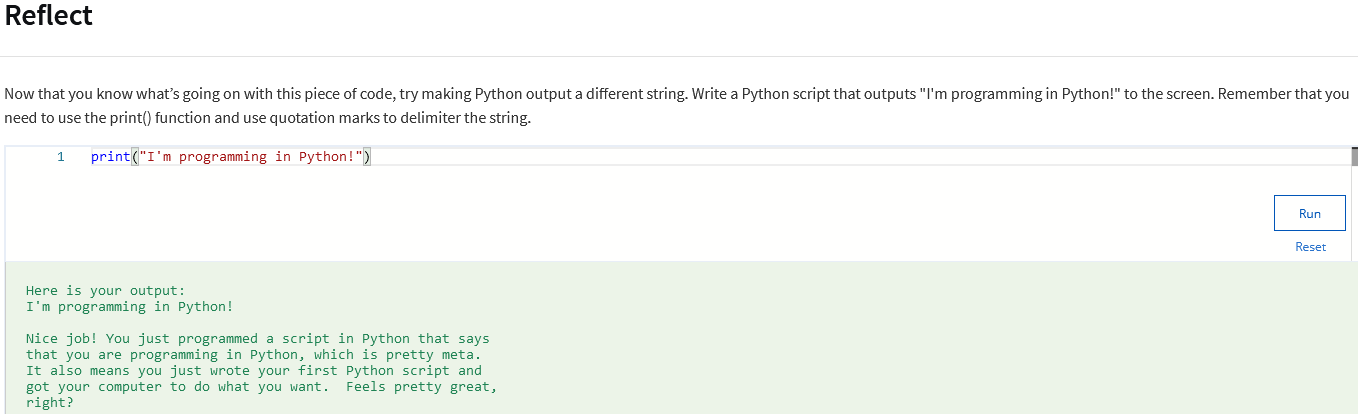


Like the statement hello world for example, the print function is part of the basic Python language, whenever we use keywords or functions that are part of the language, we're using the programming language's syntax to tell the computer what to do. So what are functions and keywords? Functions are pieces of code that perform a unit of work. We'll talk a lot more about functions later on, and you'll even learn how to write your own. Keywords are reserved words that are used to construct instructions. These words are the core part of the language and can only be used in specific ways. Some examples include if, while, and for. We'll explain all of those and a bunch more later in the course. As we called out, the keywords and functions used in Python are what makes up the syntax of the language. Once we understand how they work, we can use them to construct more complex expressions that get the computer to do what we want it to do.

Last off, notice how hello world is written between double quotation marks. Wrapping text in quotation marks indicates that the text is considered a string, which means it's text that will be manipulated by our script. In programming, any text that isn't inside quotation marks is considered part of the code.



Now, for a bit of trivia, do you know why we printed the whole world in our example?



Well, printing hello world has been the traditional way to start learning a programming language since way back in the '70s when it was used as the first example in a famous programming book called the C programming language. That example looked like this. In Python, the hello world example is just one line, in C, it's three lines, in other languages, it can be even more. While learning to write hello world won't teach you the whole language, it gives you a first impression of how functions are used, and how a program written in that language looks.

All right, now that we've written our first piece of Python code, I think you're ready for something a bit more challenging than hello world. Ready? Let's do it.

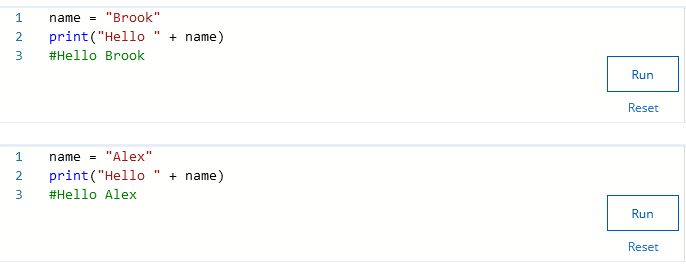
## Review: Getting Information from the User

This reading contains the code used in the instructional videos from [**Getting Information from the User**](https://www.coursera.org/learn/python-crash-course/lecture/2yv1R/getting-information-from-the-user)

## **Introduction**

This follow-along reading is organized to match the content in the video that follows. It contains the same code shown in the next video. These code blocks will provide you with the opportunity to see how the code is written, allow you to practice running it, and can be used as a reference to refer back to.

You can follow along in the reading as the instructor discusses the code or review the code after watching the video.



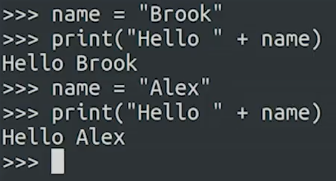
### Getting information from the User

On the whole, for a program to be useful it needs to get at least some information from the user. With this data, the program can take actions that are relevant to the user, instead of generic actions, like printing hello world.

Data can be provided to a computer in a bunch of different ways. For example, on a website you might input data by entering text into text fields or clicking links. If you're using a mobile application, maybe you'll click on buttons or select preferences from a drop-down menu. In a command line program, you might provide additional data by passing strings as parameters to the program, or you could have the program ask you for data interactively. All of these various platforms, programs, and apps process data differently. Some might take the contents of a file as data to be processed, others gather data from other sources and process it in the background.

Remember our earlier example, when we automated the process of identifying and removing duplicate emails? There, the data provided to the program was the list of emails, which would usually be given in a file that lists the emails one per line. Whichever way your application gets the data, it will need to come from somewhere. For our first examples in this course, we'll just have the data as its own line in our block of code. This is limited, but straightforward. Later in this course, and in upcoming courses, we'll introduce you to better ways of feeding data into your code.

For now though, let's see this idea in action in a very simple example. By having the name separate from the call to the print function, we're making the line of code that calls the print function generic, while still personalizing the greeting. If we then want to say hello to a different person, we only need to change the name, but the call to the print function will remain the same.



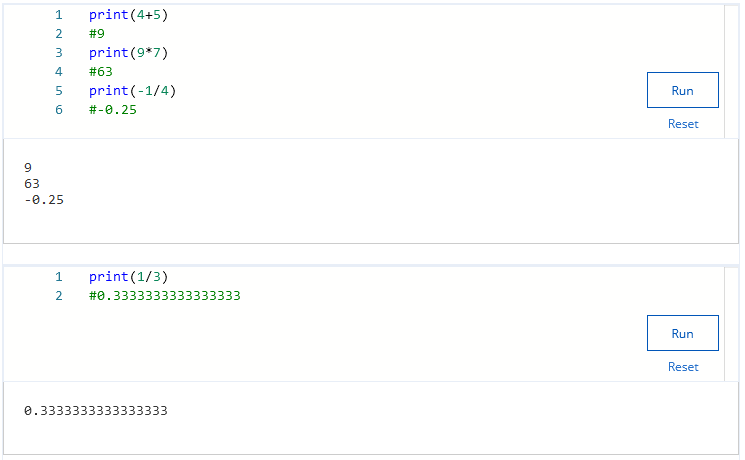
Pretty simple, right? Next up, we'll learn a few other easy things that you can get Python to do for you.

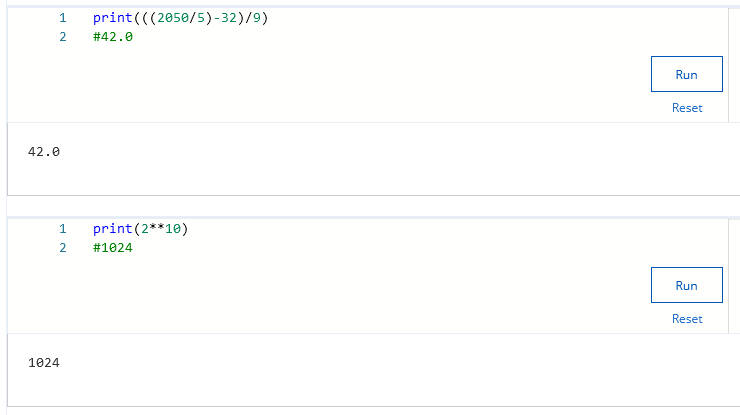
### Review: Python can be your calculator

## **Introduction**

This follow-along reading is organized to match the content in the video that follows. It contains the same code shown in the next video. These code blocks will provide you with the opportunity to see how the code is written, allow you to practice running it, and can be used as a reference to refer back to.

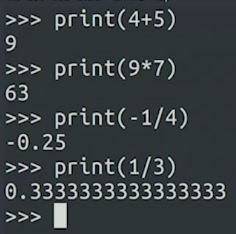
You can follow along in the reading as the instructor discusses the code or review the code after watching the video.





### Python Can Be Your Calculator

There's a ton of things that you can do with Python and you'll learn many of them in this course. But before we dive into complex subjects, let's have some fun with another simple task that you can do with Python. We are going to make Python our calculator. Let's start with something easy. So 4 plus 5 is 9, 9 times 7 is 63, minus 1 divided by 4 is minus 0.25. Easy. Repeating or periodic numbers are printed in a longer format. Let's try 1 divided by 3.



In math theory, when 1 is divided by 3, the digit 3 repeats forever after the decimal point. Of course, it's hard to display something that repeats forever. So instead, we have a representation showing lots of decimal places.

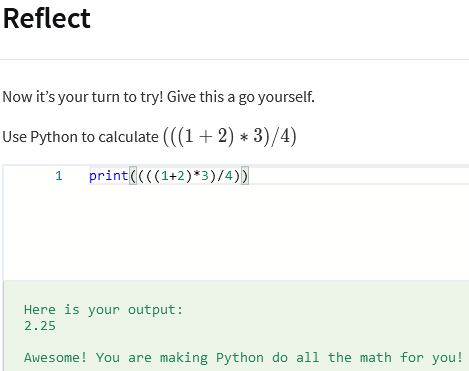
Let's get the computer something a bit trickier. Let's say we want to divide 2050 by 5, then subtract 32 and then divide the result by 9. To do this, we'll need to use parentheses, just as we do in typical math problems.



You can also use Python to get squares, cubes, or any power of n of a number. For example, let's say we want to find out what 2 to the power of 10 is. To get Python to give us the answer, we use the double star operator.



If you're starting to worry that this is becoming an algebra course, relax. We're not going to do anything more complex than what we've just seen.



If you're thinking, "Why would I use Python instead of just a normal calculator?" That's a valid question.

Experimenting in this way, you get familiar with the language's math capabilities. In IT jobs, there are many tasks that require you to use math calculations. You might need to count how many times a certain word appears in a text, or work out the average time it takes for an operation to complete, or how much you have to compress an image to fit in certain size constraints. Whatever you need to calculate, writing a script can help you do it faster and with more accuracy. So you need to know what mathematical operations are available to you.

Python actually has a lot more advanced numeric capabilities that are used for data analysis, statistics, machine learning, and other scientific applications. We won't get into these in this course. But if you want to learn more about them on your own, there's a wealth of online resources available.

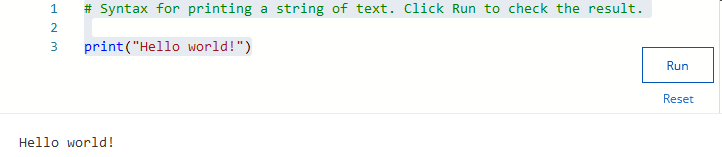
Next up, a cheat sheet to help you with programming concepts that we've just covered.

### Study Guide: First Programming Concepts

This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.

### Functions

A function is a piece of code that performs a unit of work. In the examples you've seen so far, you have only encountered the **print()** function, which outputs a message to the screen. You will use this function frequently in this course to check the results of your code. The syntax of the print() function is modeled in the example below.



* **Keywords**A keyword is a reserved word in a programming language that performs a specific purpose. In your first Python example, you briefly encountered the keywords **for** and **in**. Note that keywords will often appear in **bold** in this course.   
  In the next few weeks, you will also learn the following keywords:  
  Values: **True**, **False**, **None** Conditions: **if**, **elif**, **else** Logical operators: **and**, **or**, **not** Loops: **for**, **in**, **while**, **break**, **continue** Functions: **def**, **return** You don't need to learn this whole list now. We'll dive into each keyword as we encounter them. There are additional reserved keywords in Python. If you would like to read about them, please visit the linked “Python Keywords” article in the Resources section at the end of this study guide.   
  **Arithmetic operators**Python can calculate numbers using common mathematical operators, along with some special operators, too:
* **x + y**  Addition + operator returns the sum of x plus y
* **x - y** Subtraction - operator returns the difference of x minus y
* **x \* y** Multiplication \* operator returns the product of x times y
* **x / y** Division / operator returns the quotient of x divided by y
* **x\*\*y** Exponent \*\* operator returns the result of raising x to the power of y
* **x\*\*2** Square expression returns x squared
* **x\*\*3** Cube expression returns x cubed
* **x\*\*(1/2)**  Square root (½) or (0.5) fractional exponent operator returns the square root of x
* **x // y** Floor division operator returns the integer part of the integer division of x by y
* **x % y** Modulo operator returns the remainder part of the integer division of x by y
* **Order of operations**The order of operations are to be calculated from left to right in the following order:
  + **P**arentheses ( ), { }, [ ]
  + **E**xponents xy (x\*\*y)
  + **M**ultiplication \* and **D**ivision /
  + **A**ddition + and **S**ubtraction -
* You might find the **PEMDAS** mnemonic device to be helpful in remembering the order.
* **Resources for more information**For more information about the concepts covered in this reading, please visit:
  + [Built-in Functions](https://docs.python.org/3/library/functions.html)
* - Lists and summarizes Python’s built-in functions.
* [Python Keywords](https://www.w3schools.com/python/python_ref_keywords.asp)
* - Lists Python’s reserved keywords and a brief description of what each keyword does.
* [Different Arithmetic operators in Python](https://flexiple.com/python/arithmetic-operators-in-python/)
  + - Provides more examples of the proper syntax for using arithmetic operators in Python.
* For additional Python practice, the following links will take you to several popular online interpreters and codepads:
  + [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)

### Practice Quiz: Hello World

Practice Assignment • 10 min

## **Your grade: 100%**

Your latest: 100%•

Your highest: 100%•

To pass you need at least 80%. We keep your highest score.

### 

Question 1

What are functions in Python?

Functions let us use Python as a calculator.

Functions are pieces of code that perform a unit of work.

Functions are only used to print messages to the screen.

Functions are how we tell if our program is functioning or not.

Correct

Right on! Python functions encapsulate a certain action, like outputting a message to the screen in the case of print().

Status: [object Object]

1 / 1 point

### 

Question 2

What are keywords in Python?

Keywords are reserved words that are used to construct instructions.

Keywords are used to calculate mathematical operations.

Keywords are used to print messages like "Hello World!" to the screen.

Keywords are the words that we need to memorize to program in Python.

Correct

You got it! Using the reserved words provided by the language we can construct complex instructions that will make our scripts.

Status: [object Object]

1 / 1 point

Question 3

What does the print function do in Python?

The print function generates PDFs and sends it to the nearest printer.

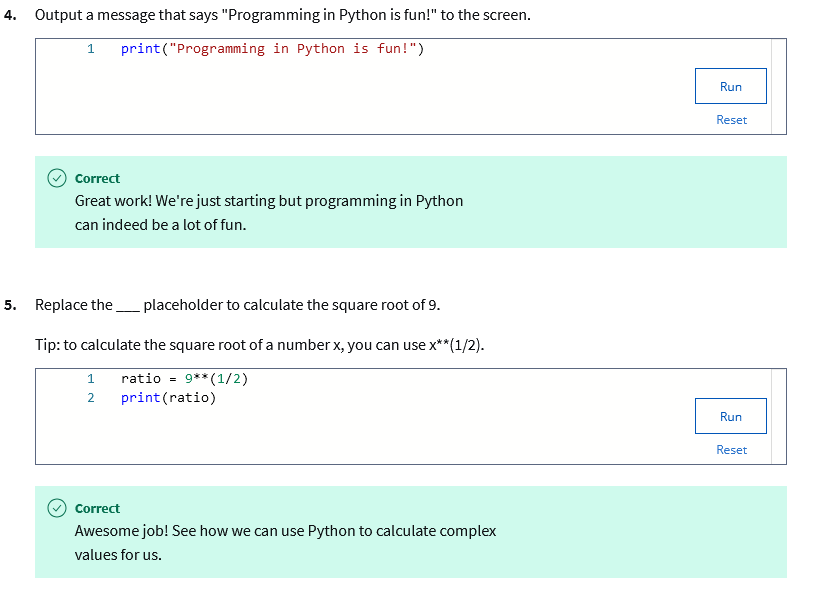
The print function stores values provided by the user.

The print function outputs messages to the screen

The print function calculates mathematical operations.

Correct

You nailed it! Using the print() we can generate output for the user of our programs.



## 

### Code editors and IDEs overview

Now that we've covered some Python coding basics, we'll introduce you to some of the coding tools. Coding tools can help you develop, write, debug and visualize your code. In this video, we'll cover the basics of code editors and Integrated Development Environments or IDEs.

Code editors are tools used to make writing code easier. With a code editor, you can write, debug and execute programs in Python.

Code editors provide features, including syntax highlighting, automatic indentation, error checking, and auto-completion.

Overall, code editors help define the code structure and function. This allows you to write code more efficiently. They also make it easier to understand variables, commands, functions, and loops. VS Code, Jupyter Notebooks and CoLab are some of the code editors we will cover in the next few videos. IDEs have more features than code editors.

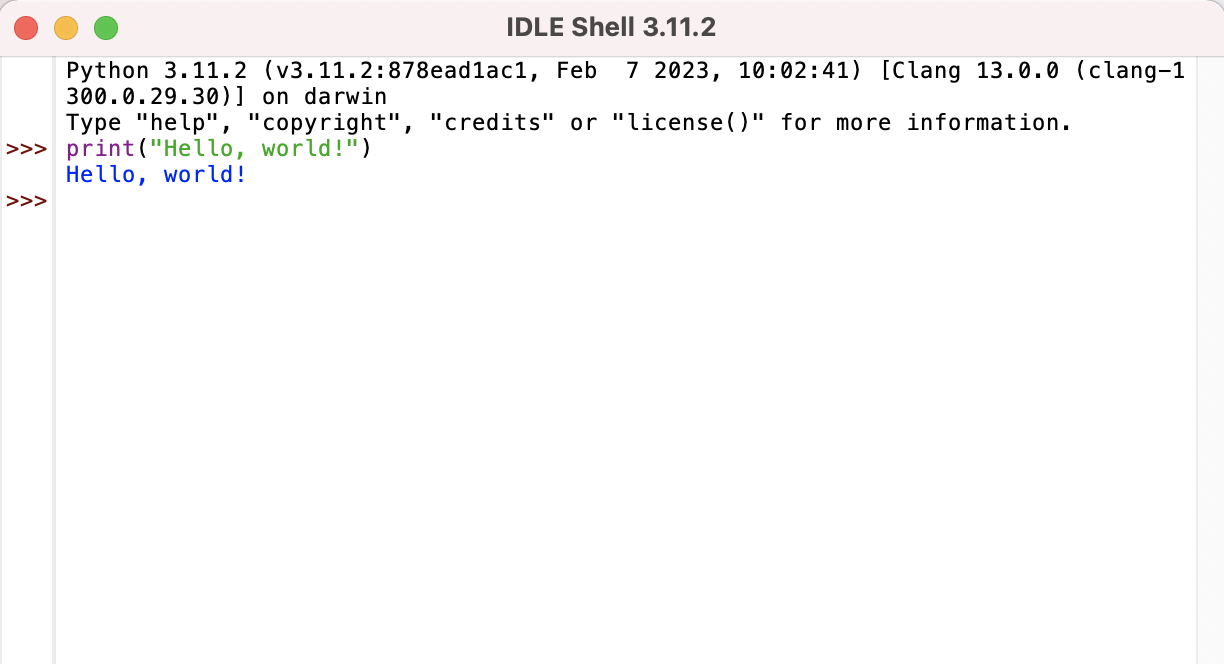
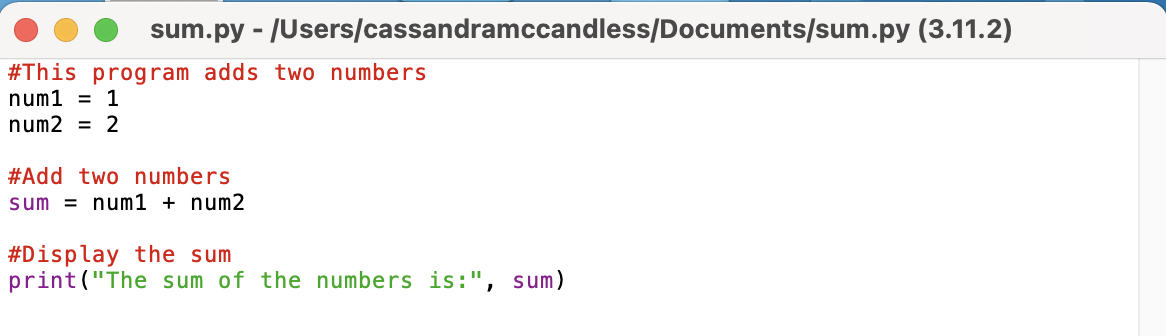
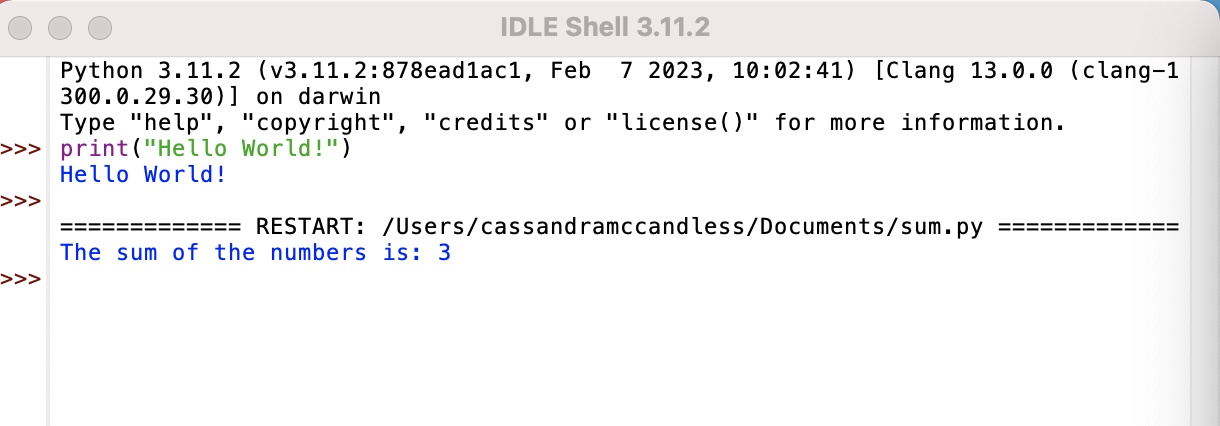
The difference between a code editor and an IDE is that a code editor is like a text editor. IDEs provide direction in writing code and allow you to see how the code executes.

You can think of a code editor like a landline phone. You can dial numbers and make phone calls with it. An IDE is more like a smartphone. You can still make phone calls, but it has additional features that allow you to make video calls, send text messages, browse the Internet and use lots of other applications.

An IDE is a software tool that simplifies the process of creating a new software application. An IDE is a software application that provides comprehensive facilities for software development. IDEs always include a code editor. IDEs allow you to develop code more efficiently by bringing multiple tools into one environment. You'll be able to edit, build, test and package all in one application. They also allow you to program applications more quickly without having to add manual integration and configuration.

Coming up, we will cover IDLE. IDLE is an IDE that comes with Python automatically. It's an excellent beginner level IDE because it's very simple to use. It may not be great for larger projects, but it's excellent for getting to know how to use an IDE. Now, let's dive into some of the code editors and IDEs that you can use as you continue in your Python learning journey.

### Review: Use the command-line

* The command-line is used to tell your computer what to do. You can use it to access servers, move files, change directories, and write scripts. In this reading, you will learn how to write Python scripts at the command-line along with the Python GUI IDLE. We will also cover the different ways you can access the command-line based on your operating system.   
  **Using the command line on MacOS**Using the spotlight search, type in “terminal.” Select the terminal application. You should see your username followed by the $ sign. MacOS comes with Python 2.7 pre-installed. You can install Python 3 from [python.org](http://python.org). Just remember, this means that you will have 2 versions of Python installed on your Mac, and you will need to pay special attention to your paths.  
  To check which version of Python you have installed on your Mac, use the following command.  
  *python --version*To check for Python3, use the following command.   
  *python3 --version***Using the command line on Windows OS**In Windows, open the start menu. In the search box type cmd. Right-click on cmd and select Run as administrator. This will open up the command-line. Windows OS does not come with Python installed. Visit the official Python [download page](https://www.python.org/downloads/windows/) for Windows. Select the Windows installer (64-bit) or (32-bit). After the installer is downloaded, double–click the file. Select **Install launcher for all users**. Follow the prompts during installation. Make sure to select the **Add python.exe PATH** checkbox. This will allow you to launch Python from the command line. Once installation is complete, you can check for Python from the command line.  
  To check for Python, use the following command. The version of Python you installed will appear.   
  *python --version***Using the command line on Linux OS**Access the Linux terminal using Ctrl + Alt + T. This will allow you to check for Python. Type python. Python comes preinstalled on most Linux systems. If the command is not found, you can install Python by writing sudo apt install python3.   
  You can begin writing Python code from the terminal. Simply type python to use the interactive mode. You can also write Python scripts using Linux with IDLE which we will cover next.   
  **Using IDLE**Python IDLE is included with Python installations on Windows and MacOS. You can download IDLE using your package manager on Linux. Python IDLE is an interactive interpreter or file editor that allows you to easily write Python scripts and programs. IDLE provides syntax highlighting, code completion, and automatic indentation.   
  Double click on the IDLE icon to open it on your computer. This will open a blank Python interpreter window. You can begin writing code right away.   
    
  You can also open a new file. Go to File → Open → New File from the menu bar. Here you can write a Python file. Once you have completed writing your Python code in the file, go to File → Save As. Give your Python file a name. Hit Save. To run the Python code in the file you saved click Run → Run Module from the menu.  
  [**Sum.py**](http://sum.py) **File  
  **[**Sum.py**](http://sum.py) **Output  
    
  Key takeaways**Whichever operating system you are using, you will be able to run Python from the command-line. Using a text editor like IDLE and running python from the command-line is best for executing and debugging individual scripts or .py files.   
  **Resources for more information**Here is a list of additional resources for writing and running Python on your local machine.
  + A [guide](https://towardsdatascience.com/a-quick-guide-to-using-command-line-terminal-96815b97b955)
* to use terminal (command-line) on Mac, Windows, and Linux operating systems.
* [IDLE documentation and instructions](https://docs.python.org/3/library/idle.html)
* [Python Scripts vs. Jupyter Notebooks](https://learnpython.com/blog/python-scripts-vs-jupyter-notebooks/)

### Use the command-line

Let's get started with our first coding tool. In this video, you'll learn how to write Python scripts at the command line, along with the Python GUI, Idle. The command line is used to tell your computer what to do. You can use it to access servers, move files, change directories, and write scripts.

In this video, we'll show you how to use the command line and Idle on the Linux operating system. If you're using another operating system, you can refer to the reading titled, "Use the command line." Now, let's go over the command line on Linux OS. Access the Linux terminal by clicking on it or using CTRL, ALT, T. This will allow you to check for Python. Type Python. Python comes pre-installed on most Linux systems. If the command is not found, you can install Python by writing sudo apt install python3. Next, it will ask you for your password. Type your password and press enter. Python version three will begin installation. Once installation is complete, you can begin writing Python code from the terminal. Simply type python3 to use the interactive mode.

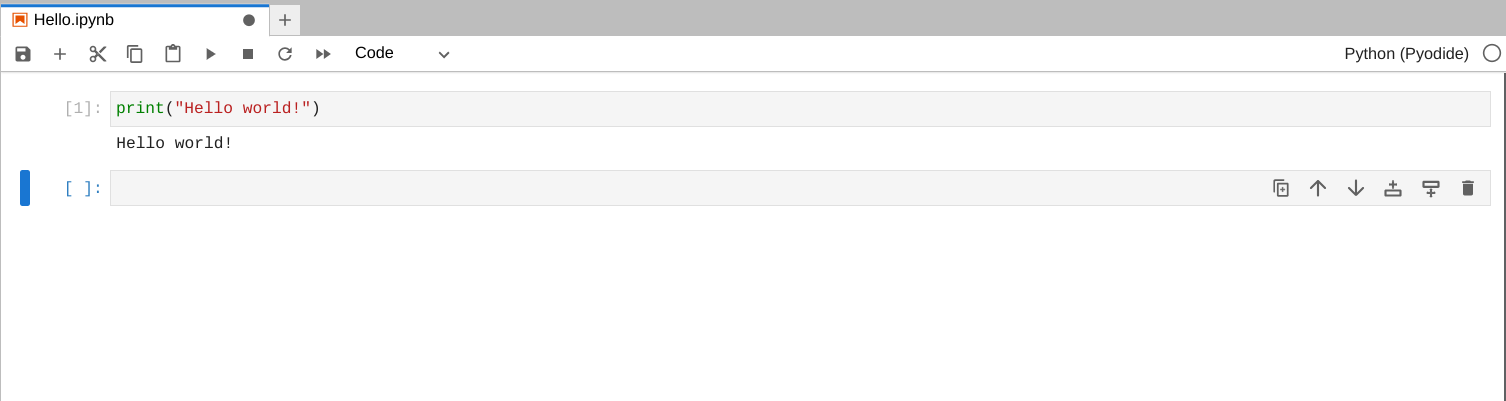
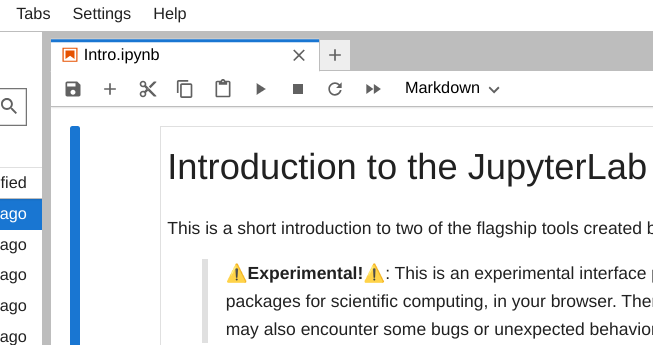
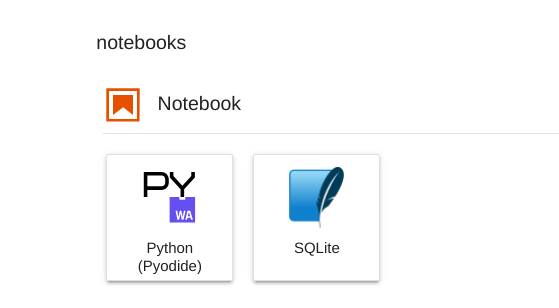
Let's try out interactive mode using a print statement. Type print Hello World, and press enter. You will see the output, "Hello World." To exit interactive mode, we will type exit().

You can also write Python scripts using Linux with Idle. Idle allows you to write Python scripts with ease on your local machine. Idle is an integrated development and learning environment for the Python programming language. Python Idle is included with Python installations on Windows and Mac OS. You can download Idle using your package manager on Linux. Python Idle is an interactive interpreter, or file editor, that allows you to easily write Python scripts and programs. Idle provides syntax highlighting, code completion, and automatic indentation.

To install Idle on Linux, type sudo apt install idle. Once installation is complete, type idle to open the interpreter and begin coding. Now we can write a Python file! Once you've completed writing your Python code in the file, go to file save as. Give your Python file a name. In this example, we will name our file sum.py. Hit save. To run the Python code in the file you saved, click run, run module from the menu, or press enter. Now you know how to use the command line to begin writing Python code. Remember, whichever operating system you're using, you'll be able to run Python from the command line. Using a text editor like Idle and running Python from the command line is best for executing and debugging individual scripts or .py files.

## 

### Review: Use JupyterLab and Jupyter Notebooks

* There are many different places and environments that you can use to write and run code. In this video, you will learn more about JupyterLab and Jupyter Notebooks to write, run, and debug Python code. JupyterLab and Jupyter Notebooks are part of an open source project called [Project Jupyter](https://jupyter.org/) and are free to use.   
  **JupyterLab**JupyterLab is a web-based interface that allows you to use Jupyter Notebooks to write, run, and debug Python code. JupyterLab is an online environment that allows you to run your code in the cloud.   
  **Jupyter Notebook**Jupyter Notebook can be used and run in the web-based interface through [JupyterLab](https://jupyter.org/try-jupyter/lab/) or on your local machine. Jupyter Notebook allows you to create documents that contain live code. You can write Python programs and scripts using Jupyter Notebooks and see how they execute all in one place. It is a great tool for creating and understanding the code you are writing because you can see your input and output all in one spot.   
    
     
  **Using Jupyter Notebook on JupyterLab**To use Jupyter Notebook in the cloud-based environment click [here](https://jupyter.org/try-jupyter/lab/?path=notebooks%2FIntro.ipynb). To create a new notebook in the lab environment, + a new tab, and select Python under  
     
  Notebook. A new Notebook will open up, and you can start writing and running Python code.   
    
    
  **Installing Jupyter**If you would like to install JupyterLab and Jupyter Notebooks on your local machine you can do this by using the pip command from the Python terminal command-line.  
  **JupyterLab**
  + Install JupyterLab *pip install jupyterlab*
  + Once installed, launch JupyterLab *jupyter-lab*
* **Jupyter Notebook**
  + Install the classic Jupyter Notebook *pip install notebook*
  + Run the notebook *jupyter notebook*
* **Sharing Jupyter Notebooks**JupyterLab and Jupyter Notebooks is a great resource for writing Python code. Notebooks can be shared and saved easily using email, GitHub, and [Jupyter Notebook Viewer](https://nbviewer.org/).   
  **Jupyter Resources**There are tons of Jupyter resources available. Here is a list of some great ones to help you get started!
  + Try [Jupyter Notebooks](https://docs.jupyter.org/en/latest/start/index.html)
* in your browser.
* Install and use [JupyterLab and Jupyter Notebooks](https://docs.jupyter.org/en/latest/install.html)
* .
* More information on [getting started](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html)
* and using Jupyter Notebooks.
* [Official documentation](https://docs.jupyter.org/en/latest/)

for Project Jupyter.

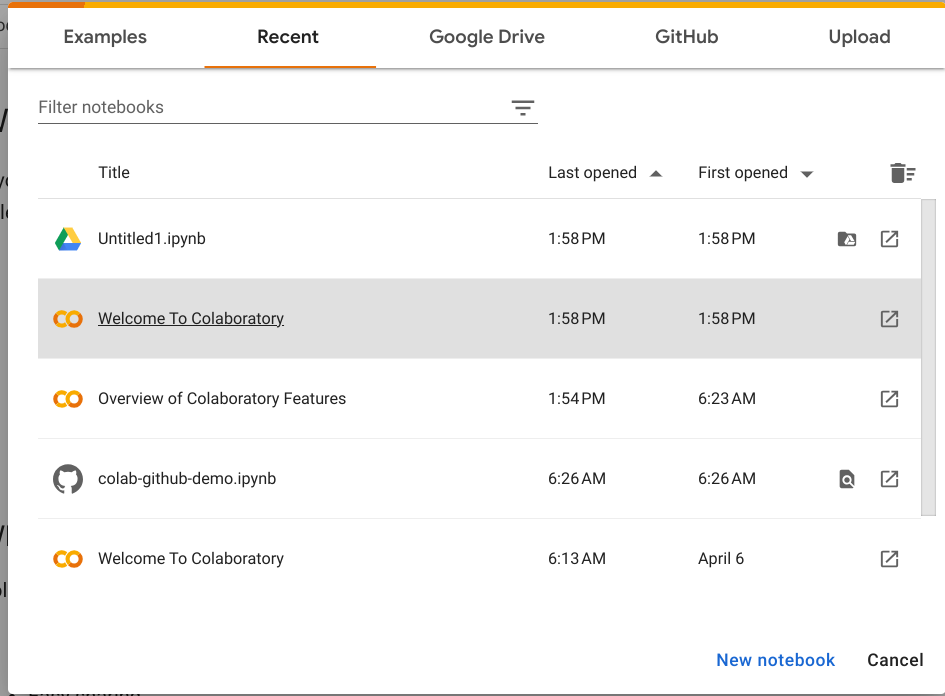
## Use JupyterLab and Jupyter Notebooks

There are many different environments that you can use to write and run code. In this video, you'll learn more about JupyterLab and Jupyter Notebooks for writing, running and debugging Python code.

Jupyter Lab and Jupyter Notebooks are part of an open source project called Project Jupyter and are free to use. You can learn more about Project Jupyter at jupyter.org.

Jupyter Lab is a web-based interface that allows you to use Jupyter Notebooks to write, run and debug Python code. Jupyter Lab provides an online environment that allows you to run your code in the cloud. Jupyter Notebook can be used in the web-based interface through Jupyter Lab or on your local machine. Jupyter Notebook allows you to create text-based documents that contain live code blocks. You can write Python programs and scripts using Jupyter Notebooks and see how they execute all in one place. It's a great tool for creating and understanding the code you're writing because you can see your input and output all in one spot. This is what a basic Jupyter Notebook looks like. Next, we will cover how to use Jupyter Notebooks. One of the most popular ways to use Jupyter Notebooks is in the cloud-based environment. You can use this environment by visiting jupyter.org. Once you're at jupyter.org, you can get started. First, click try. Then select JupyterLab. To create a new notebook in the lab environment, add a new tab and select Python under notebook. A new notebook will open up and you can start writing and running Python code. Let's give it a try. You will write your code in the cell with the blue border. Let's use a print statement. Type print "Hello world" in the cell. Next, we will run the code in the cell. To do this, we'll use the run button located at the top of the toolbar. Click on run and your cell will execute. Your output should be Hello World. You can save your Jupyter Notebook by using the save button that's also located in the top toolbar. Click save and give your notebook file a name. For this example, I will name my file Hello.ipynb. Also, if you would like to download your notebook, you can do that using the file menu located in the top left-hand corner. You can also use Jupyter Lab and Jupyter Notebooks on your local machine. You can do this by using the pip command from the Python terminal command line. To install Jupyter Lab, type pip install Jupyter Lab at the command line. Once it's installed, you can launch it by typing Jupyter-Lab. To install Jupyter Notebooks, type pip install notebook at the command line. Once it's installed, you can launch it by typing Jupyter Notebook. Jupyter Lab and Jupyter Notebooks are great resources for writing Python code. Notebooks can also be shared and saved easily using email, GitHub and Jupyter Notebook Viewer. There are also tons of resources on Jupyter Notebooks out there, so if you need more support the Jupyter website is a great resource to dive deeper into. In the next video, we will go over CoLab.

### Review: Use Colab

* Colab Notebooks are Jupyter Notebooks that are hosted by Google’s Colaboratory. In Colab, you can write and run Python code. In this reading, you will learn more about how to use Colab and its features.  
  **About Colab**Colab is a web-based platform that allows you to write and run Python code really quickly in Google Drive. It is free and ready to use with zero configuration required. You can simply visit [Colab](https://colab.sandbox.google.com/notebooks/basic_features_overview.ipynb) to get started!
* **Colab’s Features**Colab provides all of the functionality that Python offers. The cells within Colab can include code, text, and images. The code cells include executable code and rich text which make it easy to write and run code. Colab also makes it easy to include markdown in your notebooks. This is a great feature for sharing notebooks because you can add headings, paragraphs, lists, mathematical formulas, and more. You can install Python packages using the pip command within the code cell. Colab Notebooks can easily be shared with other collaborators, too.   
  Note: When you create a Colab Notebook it is stored in your Google Drive. You can easily share using the share button in the upper right hand corner of the notebook.   
  **Create a Colab Notebook**
  + Open [Google Colab](https://colab.sandbox.google.com/notebooks/welcome.ipynb#recent=true)
  + .
  + Click on New Notebook in the bottom right corner.
* 
  + Begin writing your Python code.
  + To run your code and see its execution use the
* Image of the run button in Google Colab  
  **Pro tip:** To install a Python library use the pip command with an exclamation point.  
  *!pip install <library name>***Resources for more information**Review the following resources to find out more about Colab:
  + This [resource](https://colab.sandbox.google.com/notebooks/basic_features_overview.ipynb)
* provides an overview of Colab’s features with examples.
* This is a [guide](https://colab.sandbox.google.com/notebooks/snippets/importing_libraries.ipynb)
* to installing and using Python libraries in Colab.
* This [resource](https://colab.sandbox.google.com/github/googlecolab/colabtools/blob/main/notebooks/colab-github-demo.ipynb)

provides directions on how to save and load Colab Notebooks in GitHub.

### Use Colab

Recently, we covered how to use Jupyter Notebooks. In this video, we will go over Colab notebooks. Colab Notebooks are Jupyter Notebooks that are hosted by Google's Colaboratory.

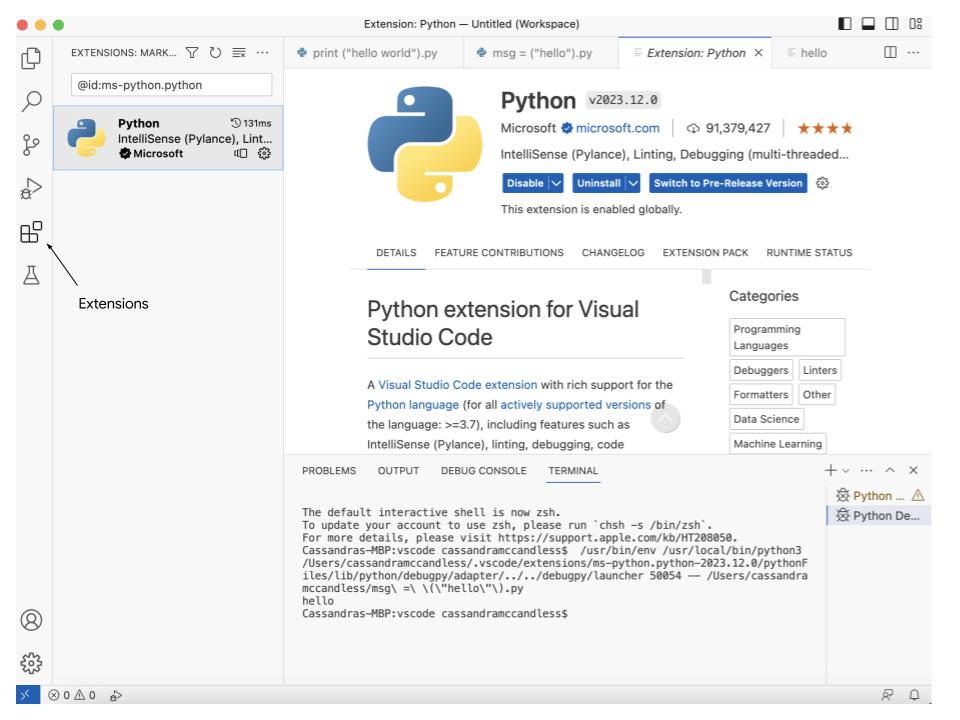
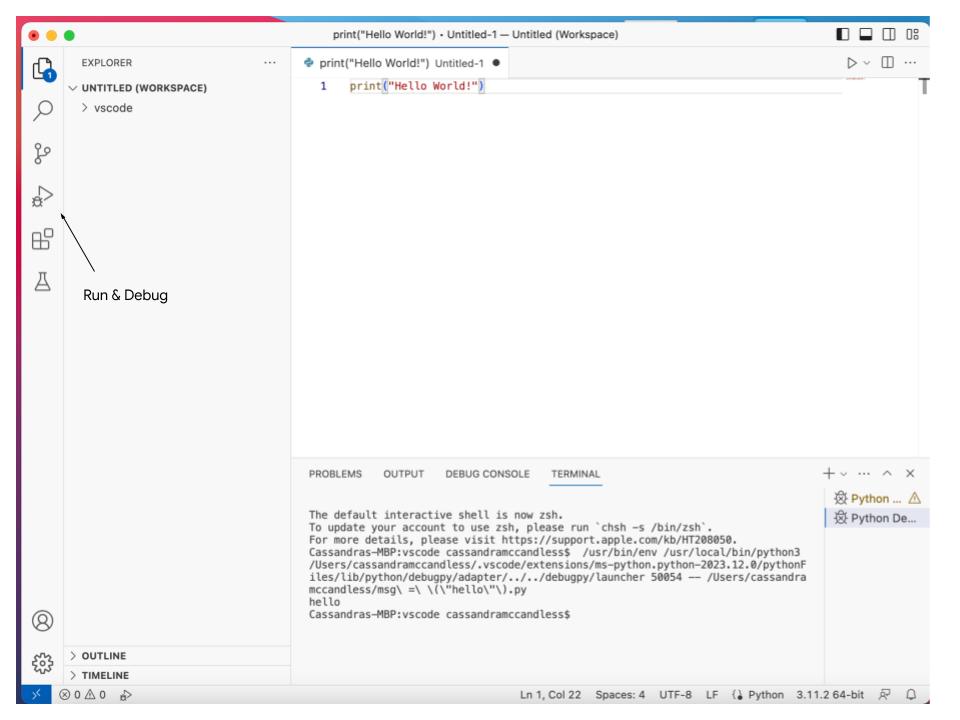
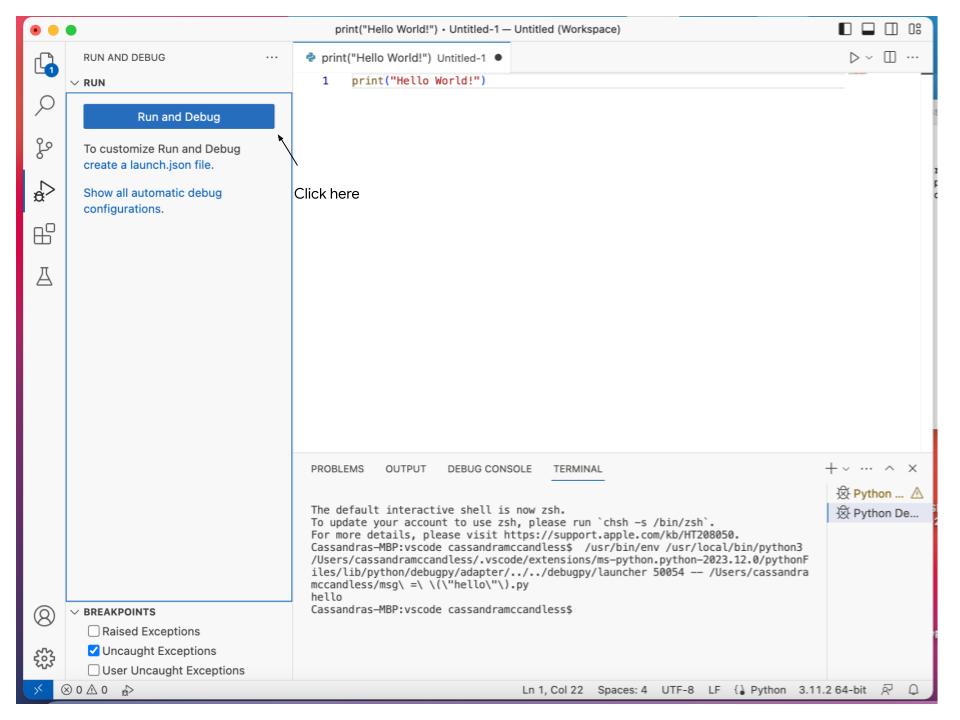
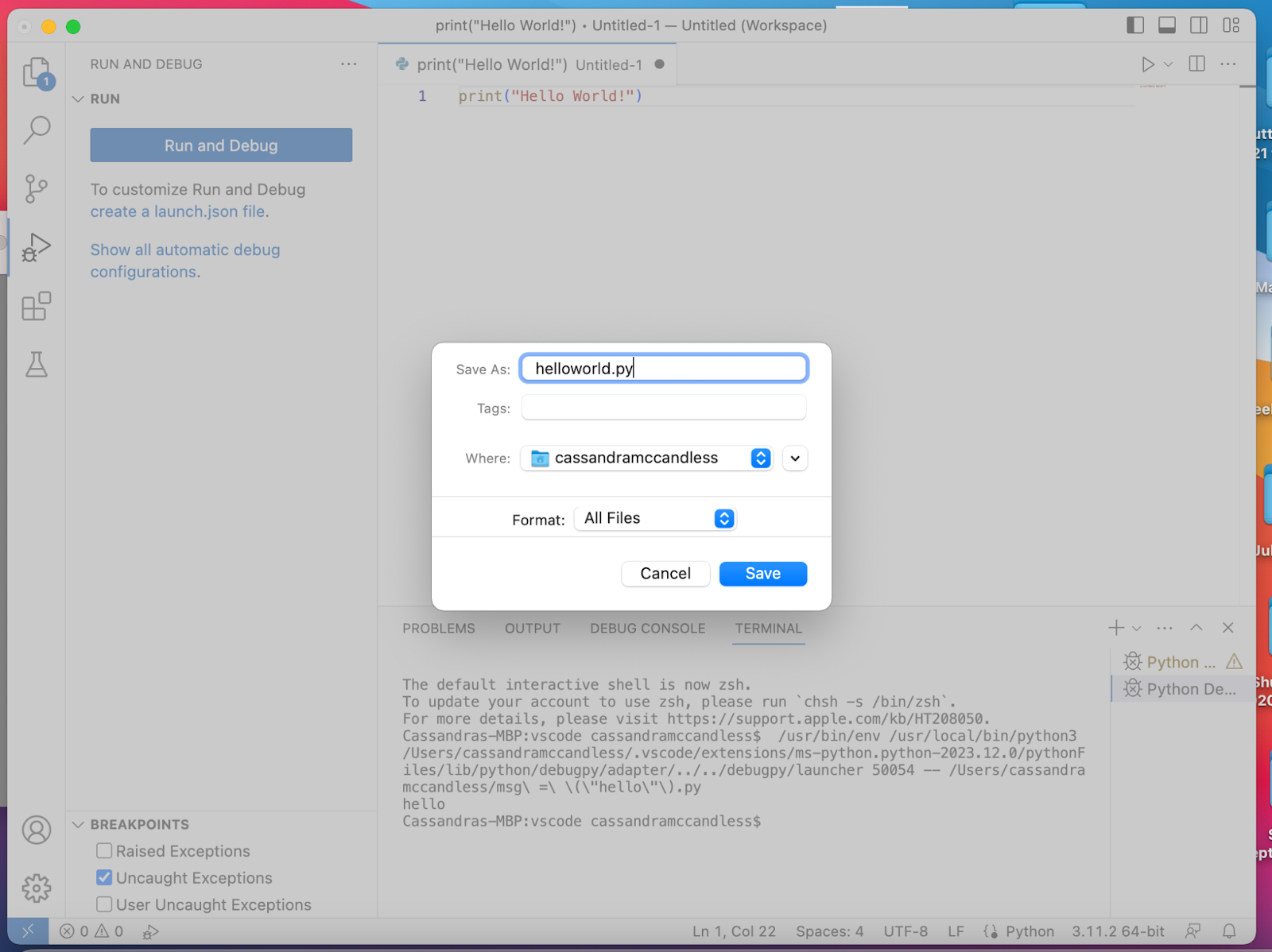
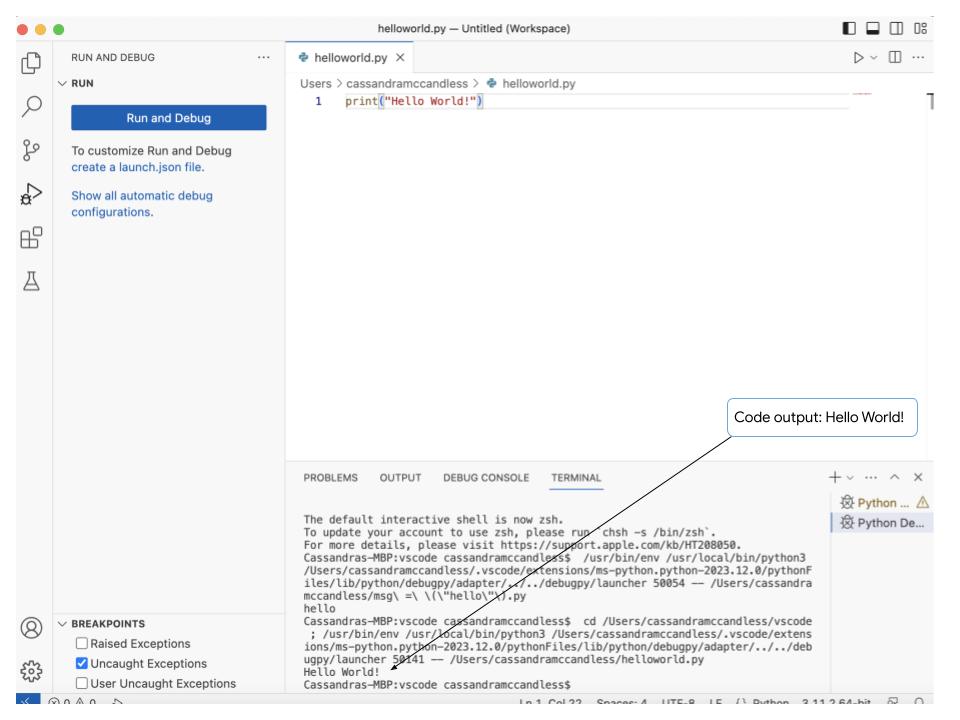
in Colab, you can write and run Python code. In this video, you'll learn more about how to use Colab and its features.

Colab is a web-based platform that allows you to write and run Python code really quickly in Google Drive. It's free and ready to use with zero configuration required. You can simply visit colab.sandbox.google.com or search Google Colab to get started.

Colab provides all of the functionality that Python offers. The cells within Colab can include code, text, and images. The code cells include executable code and rich text, which makes it easy to write and run code. Colab also makes it easy to include markdown in your notebooks. This is a great feature for sharing notebooks because you can add headings, paragraphs, lists, mathematical formulas and more. You can install Python packages using the PIP command within the code cell.

Colab Notebooks can easily be shared with other Collaborators too. Now, let's go over how to use Colab. First, open Google Colab. Again, you can do this by visiting colab.sandbox.google.com. Go to file and click on New notebook. Now you can begin writing your Python code. In the code cell, let's write a print statement that outputs your name. To run your code and see its execution, use the run button to the left of the code cell. You can also give your coding project a name. At the top, you can change Untitled.ipynb to something like Name.ipynb. This will make it easier to find your code in the future. The great thing about Colab is that it automatically stores your notebooks in Google Drive. To find your notebooks, you can simply visit your Google Drive, type in Colab Notebooks in the search bar and open up the folder titled Colab Notebooks. Here you will see all of the notebooks you've created. And that's how you use Colab. Pretty simple, right? I'm sure by now you're getting the hang of using these code editors. Let's move on to our final one, VS code.

### Review: Use VS Code

* VS code is an open source code editor that includes developer tooling. It is similar to Jupyter Notebooks and Colab, but it includes more features. VS code provides built-in support with [Intellisense](https://code.visualstudio.com/docs/editor/intellisense#:~:text=IntelliSense%20is%20a%20general%20term,%2C%20and%20%22code%20hinting.%22) code completion, an interactive debugger, and other build and scripting tools. VS code has a simple design that is easy to use. Its intuitive features make it a great choice for coding in Python!  
  **Using VS Code with Python**To use VS code for Python, you will need to have Python3, VS Code, and the VS Code Python extension installed. Make sure that you have Python3 installed on your computer by typing the following command in the terminal on your computer.   
  Linux/macOS: open a Terminal Window and type the following command:  
  *python3 --version*Windows: open a command prompt and run the following command:  
  *py -3 --version*If Python3 is installed the output should look like this:  
  *$python3 -- version  
  Python 3.11.3*If it is not installed the output will look like this:  
  *$python3 -- version  
  command not found: python***Download VS Code and install the Python extension**You can download VS code [here](https://code.visualstudio.com/Download). You can use VS code on Windows, Mac, and Linux operating systems. Download the version that is compatible with your operating system. Follow the download prompts. Once the download and installation is complete you will be able to open VS Code and begin using it.   
  Next, you will install the Python extension. You can do this by visiting the [VS Code Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-python.python). Follow the download instructions there. You can check to make sure the Python extension was added successfully by clicking on this icon in VS Code.  
    
  **Note:** You cannot use VS Code Marketplace to install the Python extension on MacOS.   
  For MacOS, open the command palette in VS code. You can do this by hitting Cmd+Shift+P. Type ‘**shell command**’. Find the Shell Command: **Install ‘code’ command in PATH**.   
  Once this is completed you can begin writing, running, and debugging Python code in VS Code!  
  **Pro tip:** To try VS Code without downloading or installation click [here](https://vscode.dev/).  
  **Create a Python File**To create a Python file in VS Code go to File → New File… → Python File. A new workspace will appear, and you can begin writing your Python code. Let’s test it out with a simple statement. Type the following statement into the new workspace.  
  *print(“Hello World!”)*Next, click on the Run and Debug icon in the left hand toolbar.   
    
  Then, click Run and Debug here:  
    
  You will be prompted to name and save your file. Give your Python file a name and click save.  
    
  Your code will run and you will see its output here.   
    
  Now you know the basics of VS Code and how to use it!   
  **Key takeaways**VS Code is an extremely robust source code editor. It uses [Intellisense](https://code.visualstudio.com/docs/editor/intellisense#:~:text=IntelliSense%20is%20a%20general%20term,%2C%20and%20%22code%20hinting.%22) technology which provides syntax highlighting and autocomplete for coding. VS code allows you to debug right in the editor with its interactive console. Overall, VS code is extremely interactive and customizable. There is also a large library of extensions that are easily integrated!  
  **Resources for more information**Here are some resources on what VS Code has to offer!
  + This [resource](https://code.visualstudio.com/#powerful-debugging)
* provides an overview of VS Code.
* This [resource](https://code.visualstudio.com/docs/languages/python)
* provides more information on using Python with VS Code and also includes a tutorial you can follow along with.
* This is the [extension library](https://marketplace.visualstudio.com/VSCode)

for VS Code.

### Use VS Code

Now that you're familiar with a few different coding tools, let's take a look at VS Code. VS Code is an open source code editor that includes developer tooling. It's similar to Jupyter Notebooks and Colab, but it includes more features. VS Code provides built-in support with IntelliSense code-completion, an interactive debugger, and other build and scripting tools. VS Code has a simple layout that makes it easy to learn and use. Its intuitive features make it a great choice for coding in Python. To use VS code for Python, you will need to have Python 3, VS Code, and the VS Code Python extension installed.

You can use VS Code on Windows, Mac, and Linux operating systems. For this video, we'll be using the Linux operating system. If you're using another operating system, we have provided a reading on using VS Code with the other operating systems that you could take a look at later on.

Now let's get started. We'll start by opening a terminal window. Type the following command, python3 --version. If Python 3 is installed, the output will show the Python version that is installed on your computer. An example of what that output would look like is python 3-10-12. If it is not installed, the output will return: command not found. This means you'll need to start by installing Python 3. To do this, go to your terminal. At the command line, you will type sudo apt install python3, and press enter. It will prompt you to enter your password. Enter your password, and it will begin installing. As Python 3 is installing, it may prompt you to continue by asking, do you want to continue? Yes slash no. Select yes and press enter. Now Python 3 is installed and ready to go. Let's move on to VS Code. You can download VS Code at codevisualstudio.com/download. Here we're going to download the version that is compatible with Linux. Let's select the .deb version of VS Code. We will give it a second to finish downloading. Next, we're going to go back to the terminal. We will navigate to the downloads folder. We will do this by typing cd Downloads/. Then we'll use the LS command. This will list the Debian file we just downloaded. The name of the file you just downloaded will appear. Next, you will type sudo dpkg -i, and the name of the file listed. Once that command is executed, Visual Studio Code is installed. To open up Visual Studio Code, we type code at the command line, and Visual Studio Code will open right up for us. The final step is setting up Python in VS Code. To do this, we will browse for extensions. To view extensions in VS Code, we will type ctrl shift X. This will show you a list of the most popular VS Code extensions on the VS Code marketplace. You can search for Python using the search bar, and click install. Once it's done installing, it will give you the option to get started with Python development. We'll jump right in by selecting: create Python file. This will open up the code editor. Let's test it out with a simple statement. We're going to type print hello world into the new workspace. Next, click on run to run the code. You'll be prompted to name and save your file. Give your Python file a name and click save. In this example, we will give our file the name helloworld.py. Your code will run, and you will see its output here. Great job. Now, to summarize, VS Code is an extremely robust source code editor. It uses IntelliSense technology, which provides syntax highlighting and auto complete for coding. VS Code allows you to debug right in the editor with its interactive console. Overall, VS Code is extremely interactive and customizable. There's also a large library of extensions that are easily integrated. Great job. That was a lot to cover. Feel free to review these videos as many times as you need to. And remember, one of the best ways to figure out which tools work best for you are to explore all the features they have to offer and practice using them.

### More on IDEs and code editors

* There are lots of code editors and IDEs out there. In this reading, we will cover some other popular IDEs. Feel free to try them out and see which ones you like best!  
  **PyCharm**PyCharm is a very popular IDE for Python development. It is an open source IDE that offers many great features to help you write better code faster. It is an intelligent code editor that has useful features, including code completion, error detection, and on-the-fly code fixes. PyCharm also allows you to navigate easily through your code with code folding, code search, and code structure diagrams. The built-in debugger provides breakpoints, step-by-step execution, and variable inspection which allows you to find and fix bugs very quickly. You can test your code within the IDE which allows you to ensure your code is reliable and error free. Finally, PyCharm also offers a wide range of plug-ins created by its community of developers which has expanded its functionality.   
  Overall, PyCharm is a great choice when it comes to an IDE. It is great for beginners because it is easy to use and has a large community of users which means there are tons of tutorials and resources available online. PyCharm is also highly customizable. It is a popular IDE that many developers use and love!  
  **PyDev**PyDev is an open source IDE for Python Development that is based on [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)). PyDev includes an intelligent code editor, code navigation, debugging, testing, and refactoring. Its refactoring engine can help you improve the structure and readability of your code which makes maintenance and debugging easier. PyDev is also fully integrated with Eclipse which means you can use all of Eclipse’s features like project explorer, the perspective system, and its built-in debugger.   
  PyDev is a great IDE for developers at any stage! It's free to use, compatible with many versions of Python, and its plugins make it extensible. PyDev has a large community of users and developers which means there are lots of tutorials, resources, and forums online.  
  **Notepad++**Notepad++ is an open source text editor for Microsoft Windows environments. It supports many programming languages, including Python. Notepad++ can open multiple files and tabs. It also does syntax highlighting for known file types. Syntax highlighting helps you edit code because it displays text in different colors and fonts that help you keep track of the classes, methods, functions, and objects in your code. It also helps you identify errors in your code so you can spend less time debugging. Notepad++ has a number of other advanced features, like auto-completion and file management systems that can help you code more efficiently.   
  Notepad++ is a great text editor for developers who work in a Microsoft Windows environment! It has an active, collaborative community. Users can ask the community for help and even contribute to the project!  
  **Key Takeaways**PyCharm, PyDev, and Notepad++ are popular tools used for writing Python code. To try them out, you can download and install them here!
  + For more information on PyCharm and to download it, click [here](https://www.jetbrains.com/pycharm/)
* .
* For more information on PyDev and to download it, click [here](https://www.pydev.org/download.html)
* .
* For more information on Notepad++ and to download it, click [here](https://notepad-plus-plus.org/)
  + .
* **Resources for more information**Now, we have covered different coding tools and how to use them. Code editors and IDEs are essential for writing efficient Python code. Every programmer has their favorite tools, and you can find yours by comparing features and testing them out.   
  Here are some resources you might find useful as you dive deeper into Python programming.
  + This [blog](https://blog.jupyter.org/)
* includes the latest news and updates on Jupyter Lab and Jupyter Notebooks.
* This [resource](https://code.visualstudio.com/docs/getstarted/tips-and-tricks)
* includes tons of tips and tricks for using VS Code.
* This [reading](https://amitness.com/2020/06/google-colaboratory-tips/)
* includes more information on some of the features Colab has to offer, including GitHub integration, keyboard shortcuts, and more!
* This [resource](https://realpython.com/python-idle/)

has tons of information on IDLE. It covers everything from working with files to debugging and more!

## Module Review

### First Steps Wrap Up

Congrats. You made it to the end of the first module. Great job. You've taken the first steps to learning a new programming language, and growing your IT skill set. Getting there shows real determination and a will to learn. We've covered a lot of topics, and many might be new to you if you've never learned about programming before. You've discovered what scripting is, what the syntax and semantics of a programming language are all about, and how they relate to automation. We've got to grip some small blocks of Python code, talked about why Python is relevant to IT, and explored what other programming languages are available. We've had our first approach to how to input data, and write a script that puts this data to use, and we've seen how you can use Python to perform typical math calculations. Not bad for your first Python steps, right? This is just the beginning of an exciting journey, learning to code, and we hope you're eager to learn more. Coming up, get ready for your first graded assessment. These assessments help you check whether you've understood all the concepts and that you're ready to move on to the next stage. Now, don't worry. If at any point you're not sure about a question, you can always review the videos and readings to remind yourself of the answer. Remember, that everybody learns at different speeds. So take your time, really get familiar with the concepts. Once you feel ready, the assessment is there waiting for you.

### Glossary terms from course 1, module 1

Terms and definitions from Course 1, Module 1

**Automation:** The process of replacing a manual step with one that happens automatically

**Client-side scripting language:** Primarily for web programming; the scripts are transferred from a web server to the end-user’s internet browser, then executed in the browser

**Code editors:** Tools to provide features, including syntax highlighting, automatic indentation, error checking, and autocompletion

**Computer program:** A step-by-step list of instructions that a computer follows to reach an intended goal

**Functions:** A reusable block of code that performs a specific task

**IDE:** A software application that provides comprehensive facilities for software development

**Interpreter:** The program that reads and executes code

**Input:** Information that is provided to a program by the end user

**Logic errors:** Errors in code that prevent it from running correctly

**Machine language:** Lowest-level computer language. It communicates directly with computing machines in binary code (ones and zeros)

**Object-oriented programming language:** Most coding elements are considered to be objects with configurable properties

**Output:** the end result of a task performed by a function or computer program

**Platform-specific scripting language:** Language used by system administrators on those specific platforms

**Programming:** The process of writing a program to behave in different ways

**Programming code**: A set of written computer instructions, guided by rules, using a computer programming language

**Programming languages:** Language with syntax and semantics to write computer programs

**Python:** A general purpose programming language

**Python interpreter:** Program that reads and executes Python code by translating Python code into computer instructions

**Script:** Often used to automate specific tasks

**Semantics:** The intended meaning or effect of statements, or collections of words, in both human and computer languages

**Syntax:** The rules for how each statements are constructed in both human and computer languages

**Variables:** These are used to temporarily store changeable values in programming code

### Study Guide: Module 1 Graded Quiz

It is time to prepare for your first graded quiz! Please review the following items from this module before beginning the Module 1 Graded Quiz. If you would like to refresh your memory on these materials, please revisit the Study Guides located before each Practice Quiz in Module 1: [Study Guide: Introduction to Programming](https://www.coursera.org/learn/python-crash-course/supplement/JNRad/study-guide-introduction-to-programming), [Study Guide: Introduction to Python](https://www.coursera.org/learn/python-crash-course/supplement/BqgFu/study-guide-introduction-to-python), and [Study Guide: First Programming Concepts](https://www.coursera.org/learn/python-crash-course/supplement/e5FGg/study-guide-first-programming-concepts)

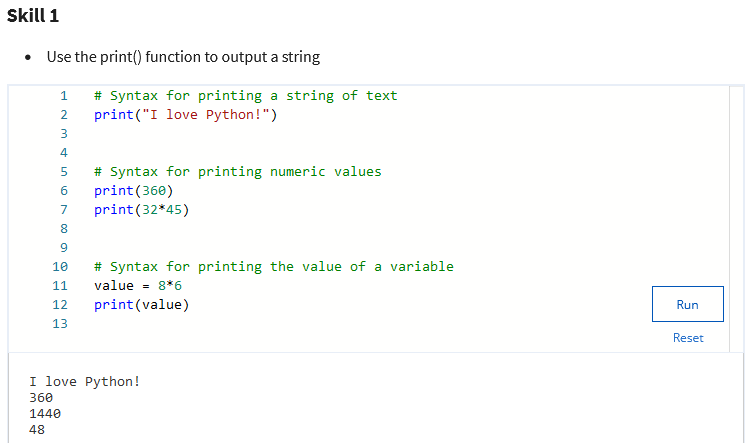
.

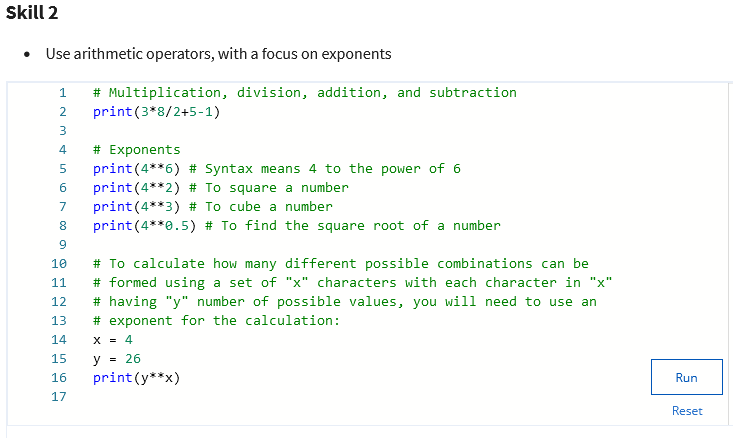
# **Knowledge**

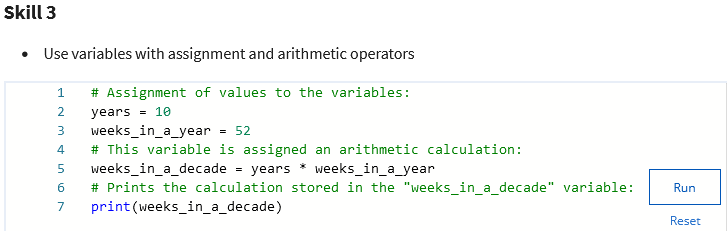
* Benefits of the Python programming language
* How Python compares to other programming languages
* How the knowledge of one programming language affects learning and using other programming languages
* How scripting applies to automation
* Proper syntax for arithmetic operations
* Functions and keywords used to display data
* Why precision is important when programming computer instructions

## **Terms**

* Computer programs
* Programming language
* Syntax
* Semantics
* Logic errors
* Script
* Automation
* Function

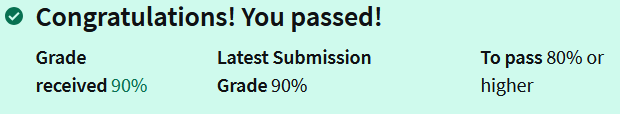


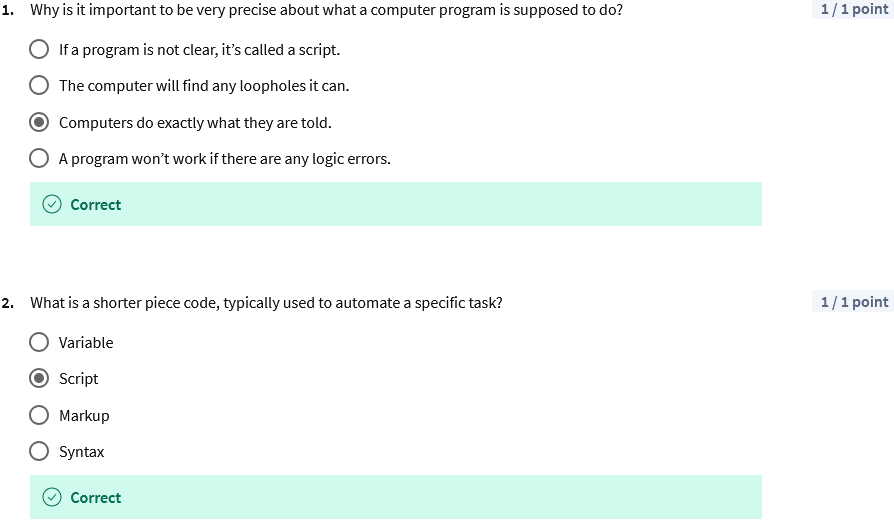


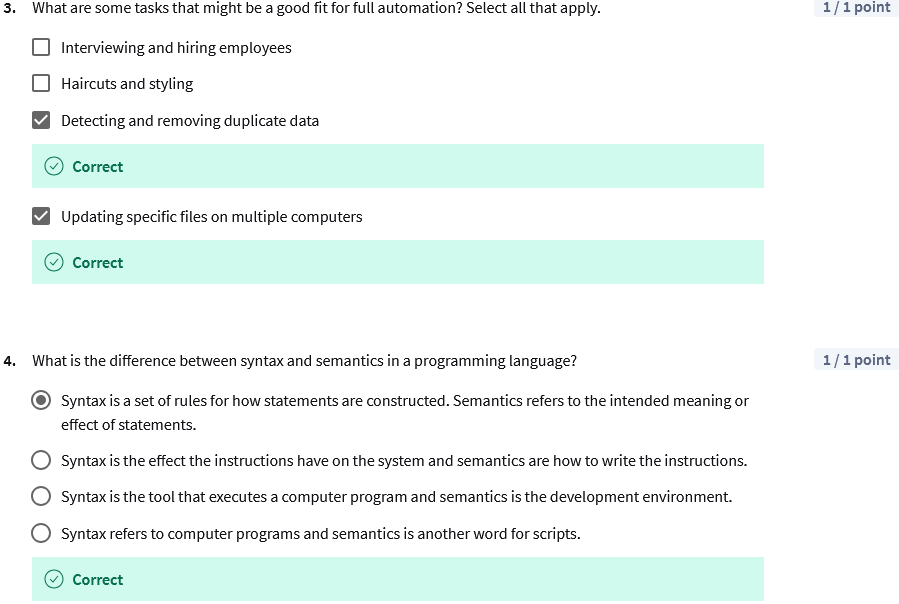


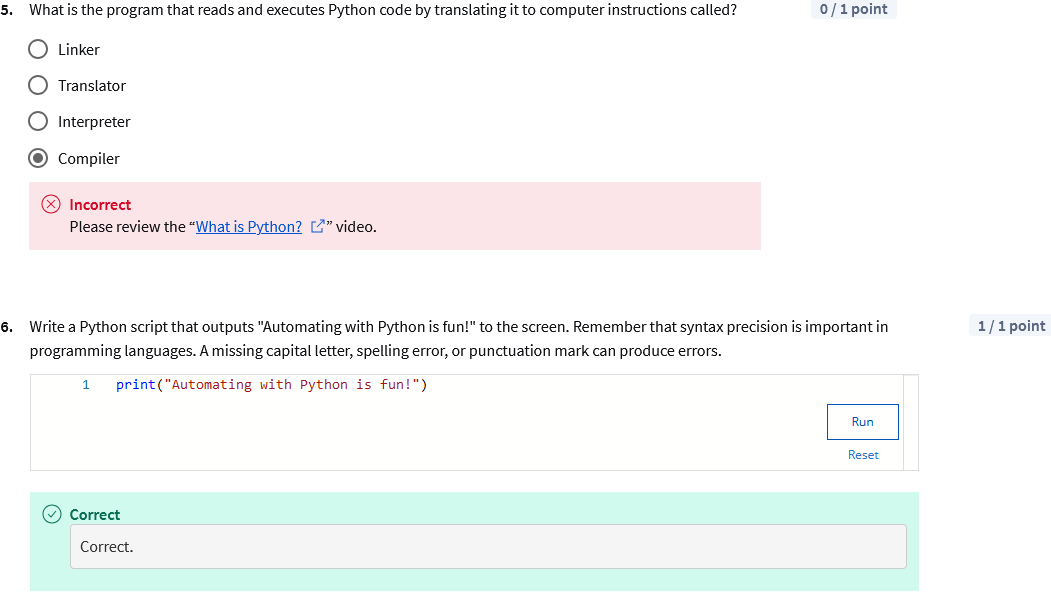
* **Reminder: Correct syntax is critical**Using precise syntax is critical when writing code in any programming language, including Python. Even a small typo can cause a syntax error and the automated Python-coded quiz grader will mark your code as incorrect. This reflects real life coding errors in the sense that a single error in spelling, case, punctuation, etc. can cause your code to fail. Coding problems caused by imprecise syntax will always be an issue whether you are learning a programming language or you are using programming skills on the job. So, it is critical to start the habit of being precise in your code now.   
  No credit will be given if there are any coding errors on the automated graded quizzes - including minor errors. Fortunately, you have 3 optional retake opportunities on the graded quizzes in this course. Additionally, you have unlimited retakes on practice quizzes and can review the videos and readings as many times as you need to master the concepts in this course.   
  Now, before starting the graded quiz, review this list of common syntax errors coders make when writing code.  
  **Common syntax errors:**
  + Misspellings
  + Incorrect indentations
  + Missing or incorrect key characters:
    - Parenthetical types - ( curved ), [ square ], { curly }
    - Quote types - "straight-double" or 'straight-single', “curly-double” or ‘curly-single’
    - Block introduction characters, like colons - :
  + Data type mismatches
  + Missing, incorrectly used, or misplaced Python reserved words
  + Using the wrong case (uppercase/lowercase) - Python is a case-sensitive language
* **Resources**For additional Python practice, the following links will take you to several popular online interpreters and codepads:
  + [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)

### Module 1 Graded Assessment









\*5 Answer:

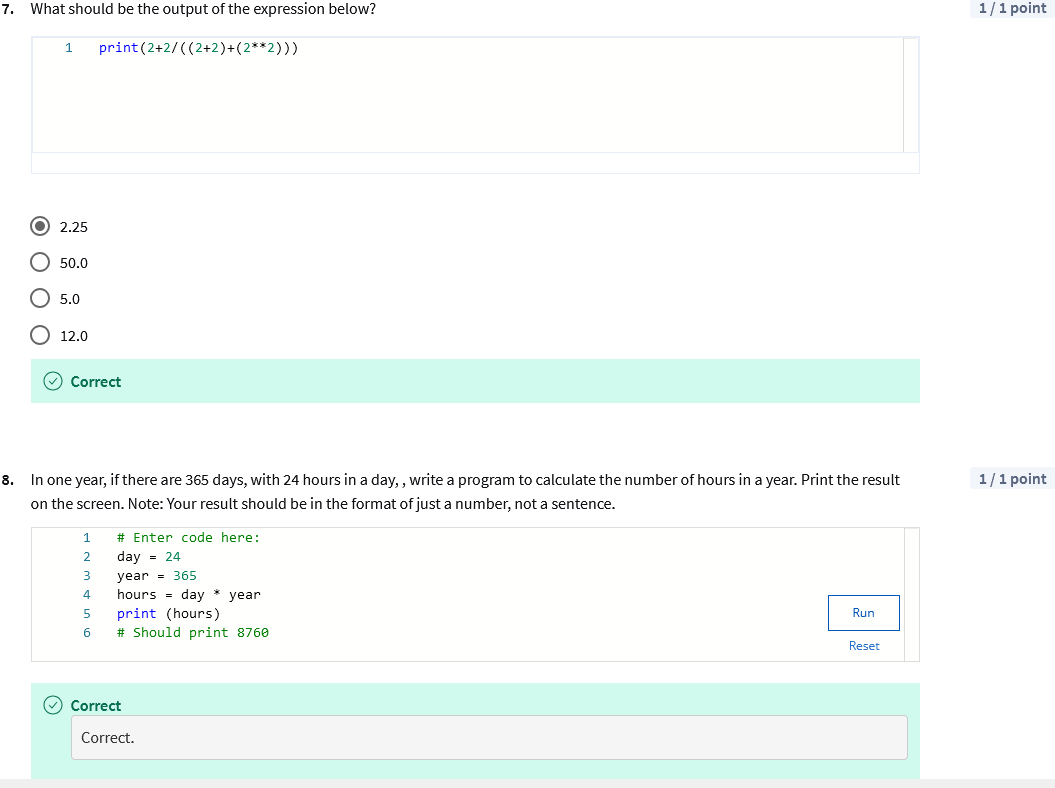
The correct answer is **Interpreter**.

### **Explanation:**

* An **Interpreter** reads and executes Python code directly, translating it into computer instructions line by line.
* A **Compiler**, on the other hand, translates the entire code into machine code before execution.
* A **Linker** combines object code files into a single executable.
* A **Translator** is a general term that could apply to any program that converts one language into another but is not specific to this context.

In Python, the **Python Interpreter** (like python or python3 command) is used to run Python code directly.

### **Correct Answer: Interpreter**



## 

## 