TABLE OF CONTENTS

A Special Thank You to Steve & Lesley Steck, Autumn Wells-Stewart and Kerri Panico for creating this curriculum and sharing the gift of Python with classrooms everywhere!

INSTALLING AND CONFIGURING PYTHON:

Step 1:
Download and install Python 3.11.6 (this is the newest version of Python that has been verified to work with the PythonFlyer):

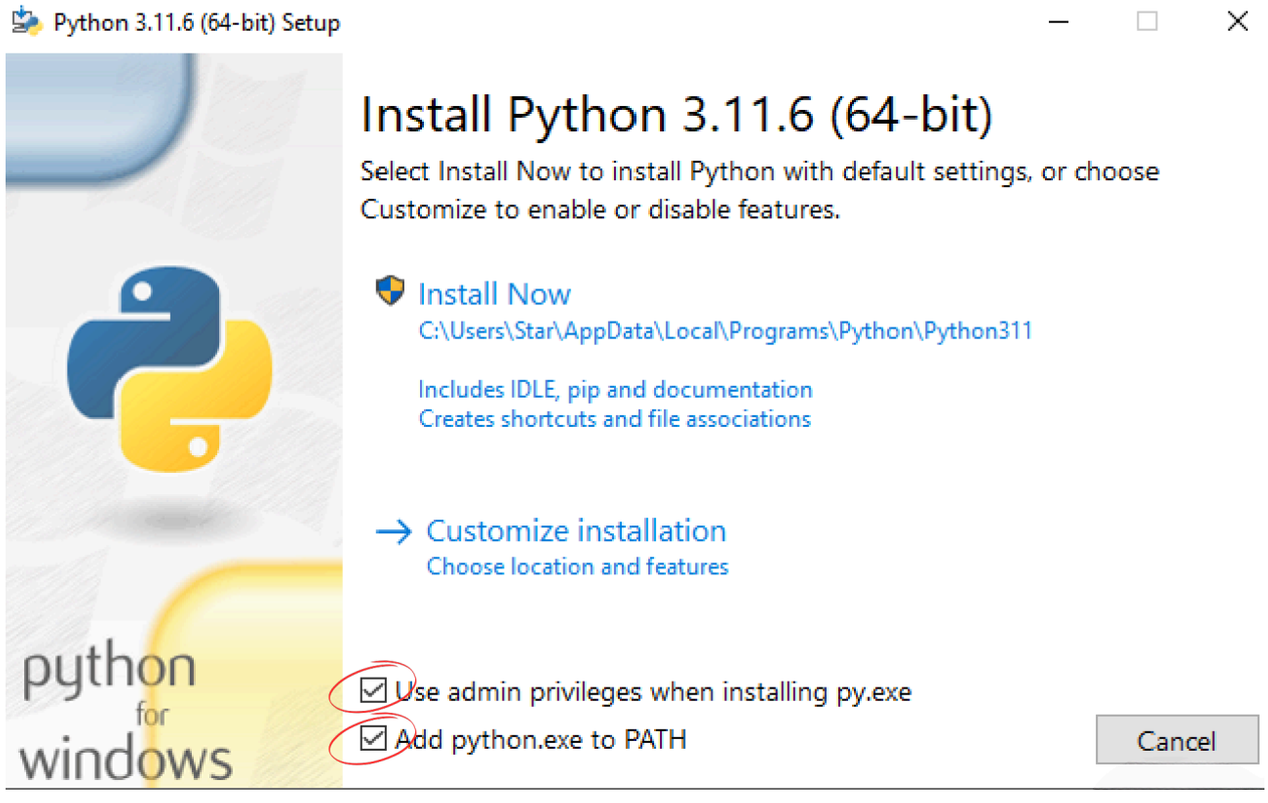
https://www.python.org/downloads/release/python-3116/

Scroll to the bottom of this page and download the Windows installer (64-bit) version (or if you have an older laptop, you may need to install the 32bit version).

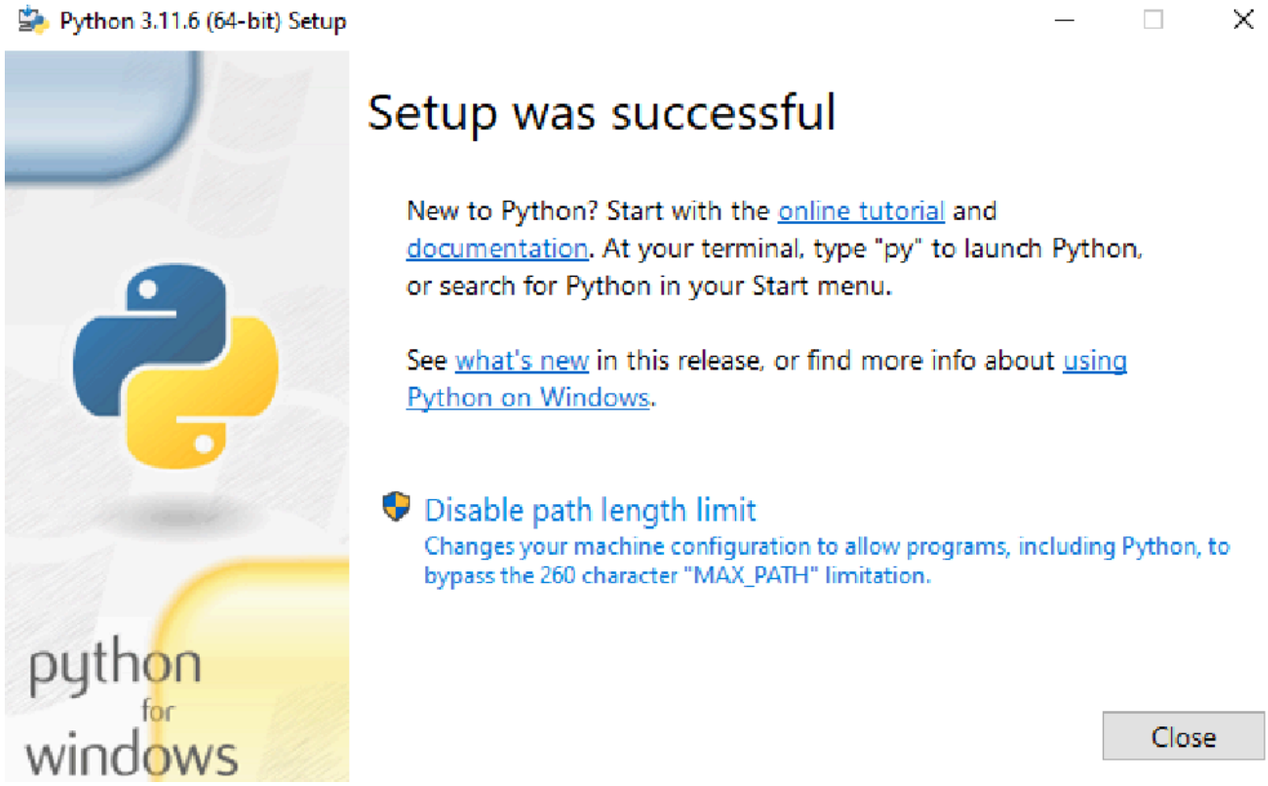
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
Gzipped source tarball	Source release		ed23dadb9f1b9fd2e4e7d78619685c79	25.4 MB	SIG	.sigstore
XZ compressed source tarball	Source release		d0c5a1a31efe879723e51addf56dd206	19.1 MB	SIG	.sigstore
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	3052a3dd9f61a5bad1ff16c46cfaa491	42.2 MB	SIG	.sigstore
Windows installer (64-bit)	Windows	Recommended	4a501c073d0d688c033d43f85e22d77e	24.8 MB	SIG	.sigstore
Windows installer (ARM64)	Windows	Experimental	34333bf5eb5fbd7a5eba5aa272b4e0ac	24.1 MB	SIG	.sigstore
Windows embeddable package (64-bit)	Windows		ff5f34b8d2504c49fc94ffc29998b8a0	10.7 MB	SIG	.sigstore
Windows embeddable package (32-bit)	Windows		d035d12d72e2d62b6e5219ea8f3bda39	9.6 MB	SIG	.sigstore
Windows embeddable package (ARM64)	Windows		314e56d6f35508570eca9c3407395b01	10.0 MB	SIG	.sigstore
Windows installer (32-bit)	Windows		041b1030be54ef78fd4c3a01ccb26267	23.5 MB	SIG	.sigstore

Once the installer is run, a window will popup with two checkboxes on the bottom of the window. Make sure both are checked before clicking "Install Now"



Click the "Close" button on the next screen.

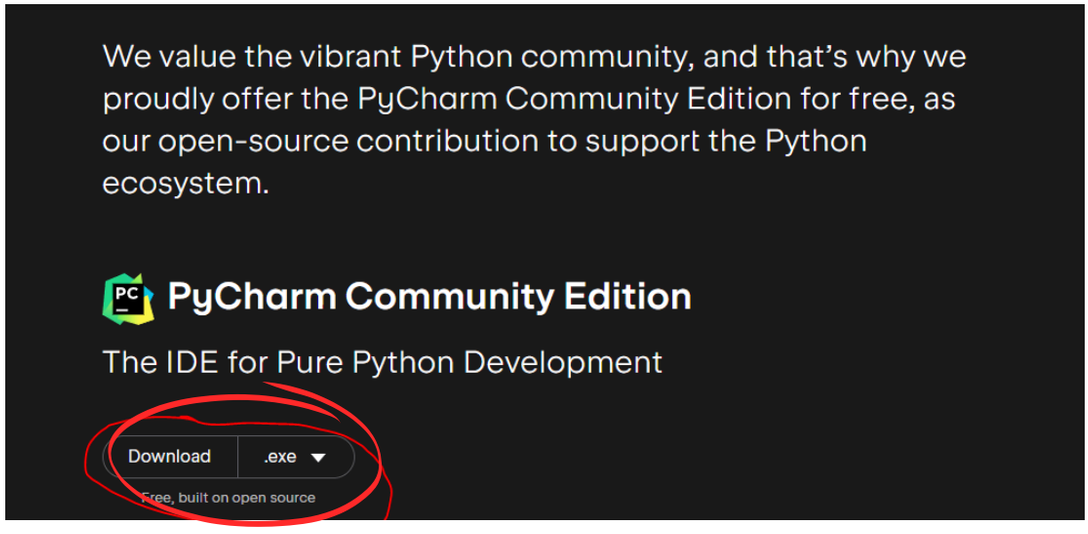


It is recommended that you reboot your computer before proceeding.

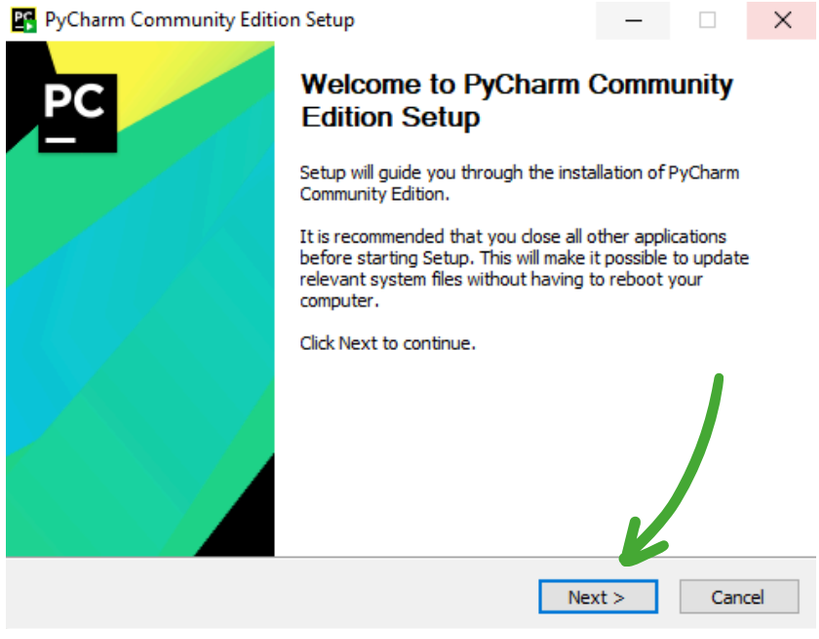
Step 2:
Download and install PyCharm:

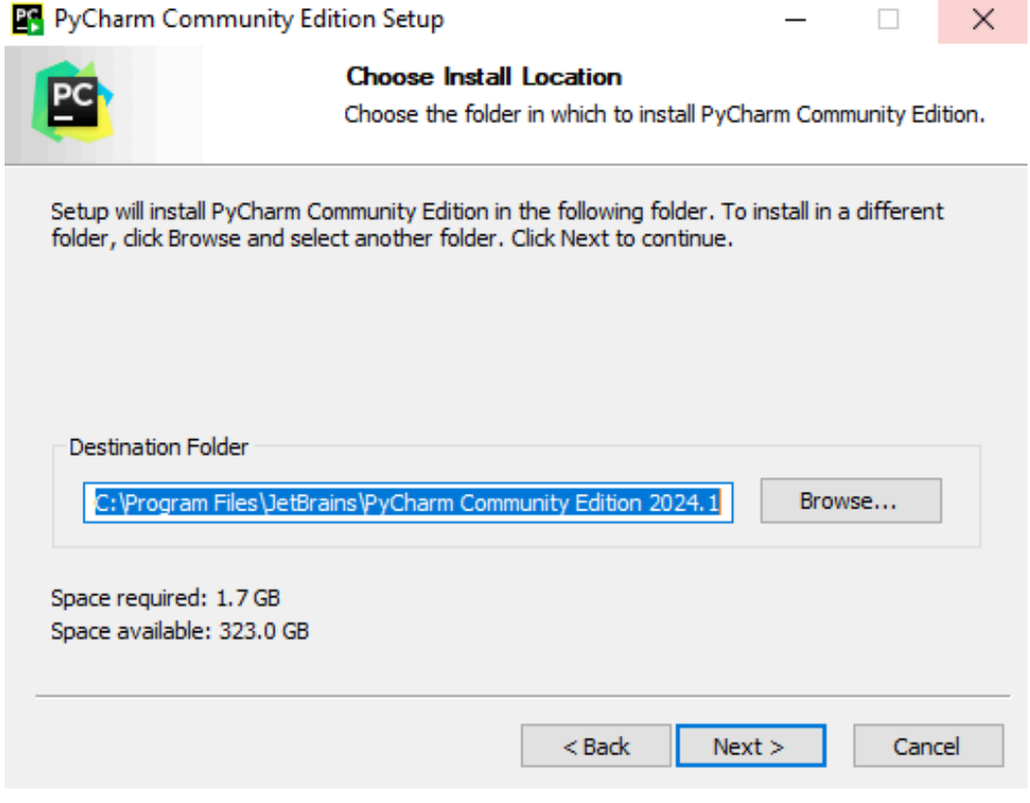
https://www.jetbrains.com/pycharm/download/

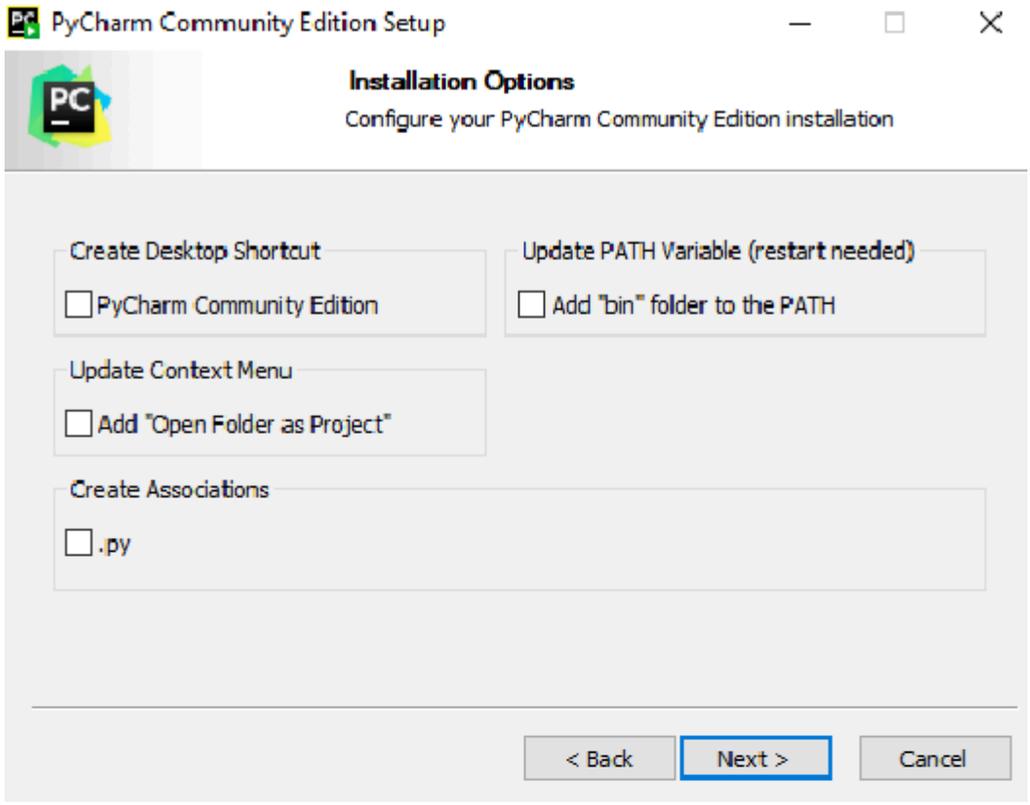
Scroll to the bottom of this page and download the PyCharm Community Edition:

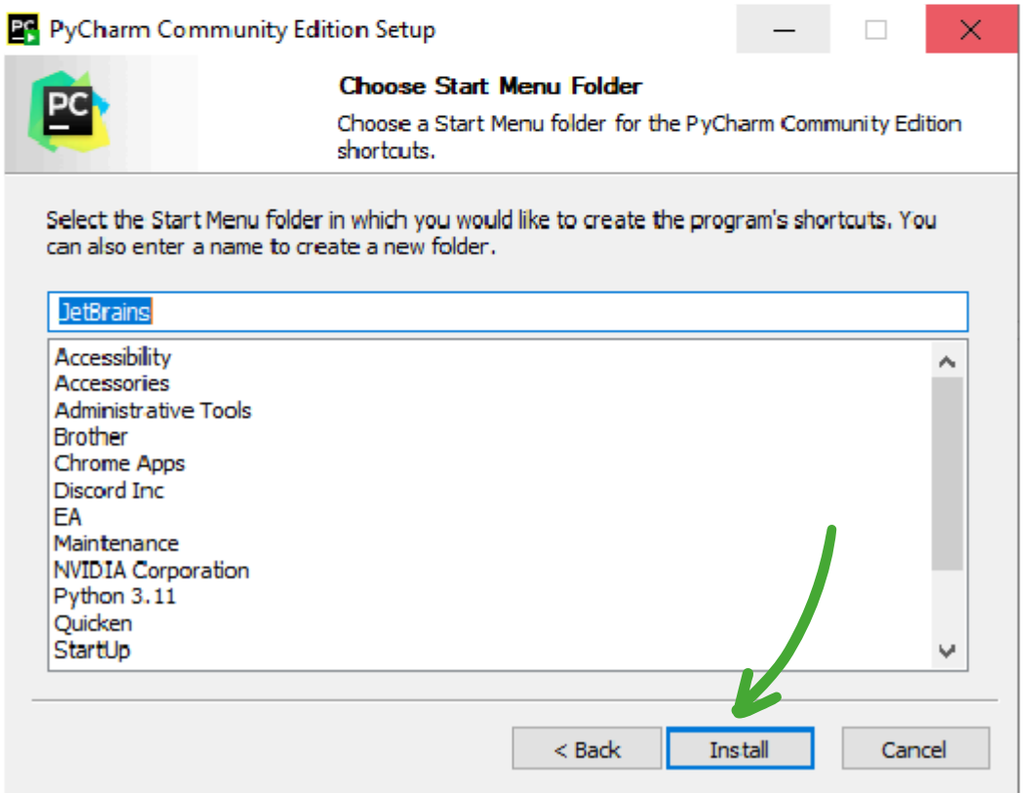


Click the “Next” button on all of the installation prompts, until it reaches a window that has “Install”, then click the “Install” button.

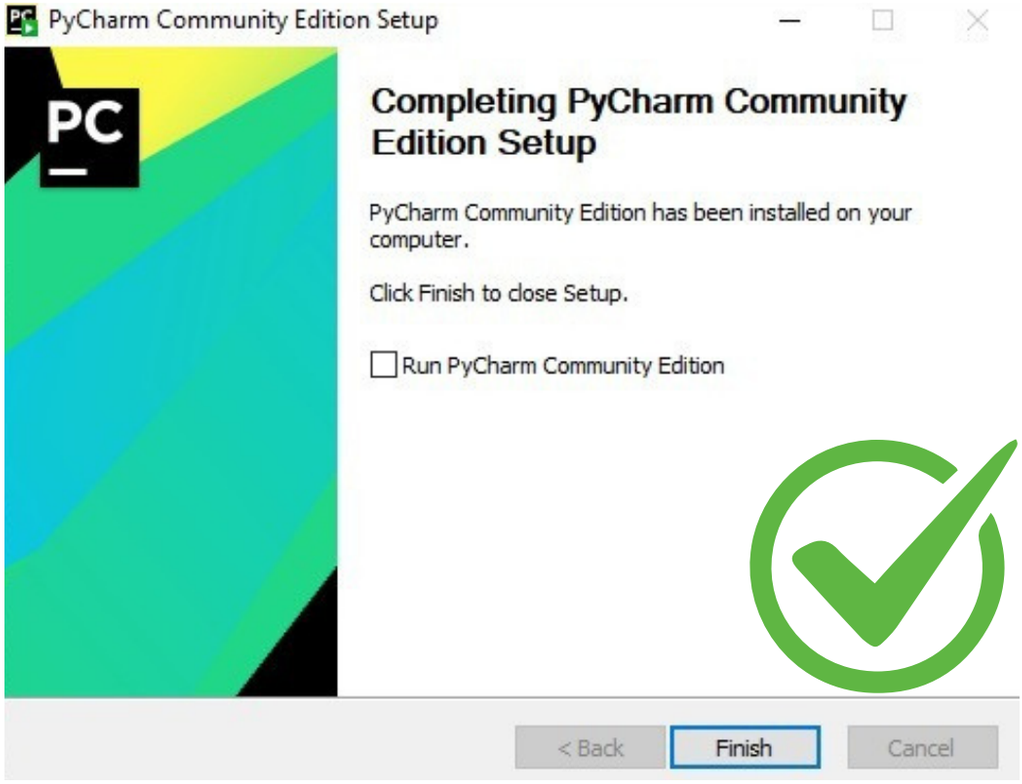








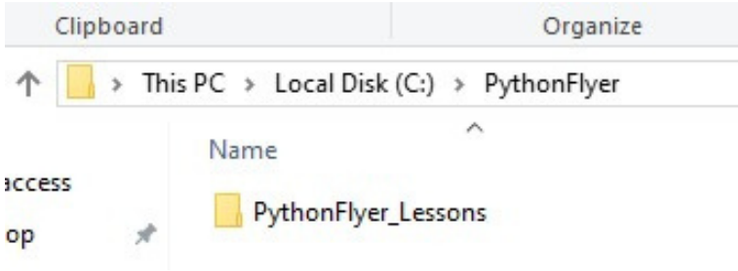
Click the "Finish" button and PyCharm should now be installed on your machine:



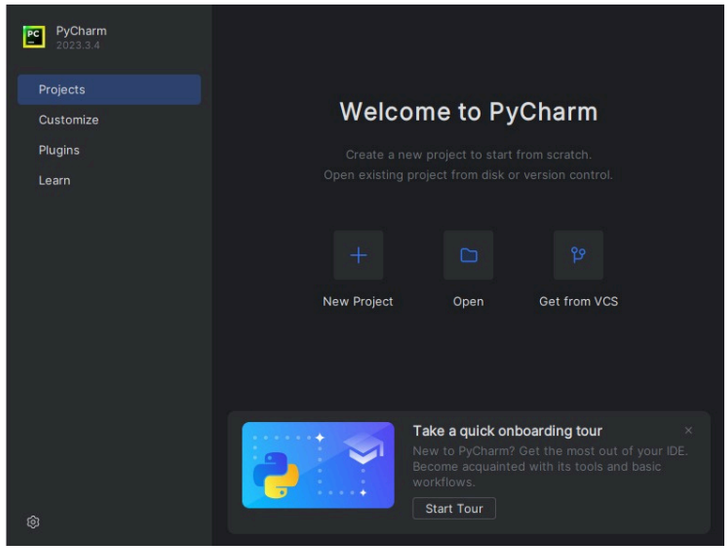
Step 3:
Setup the Python Flyer Lesson Files:

- Extract the PythonFlyer_Lessons.zip file to a folder on your computer.

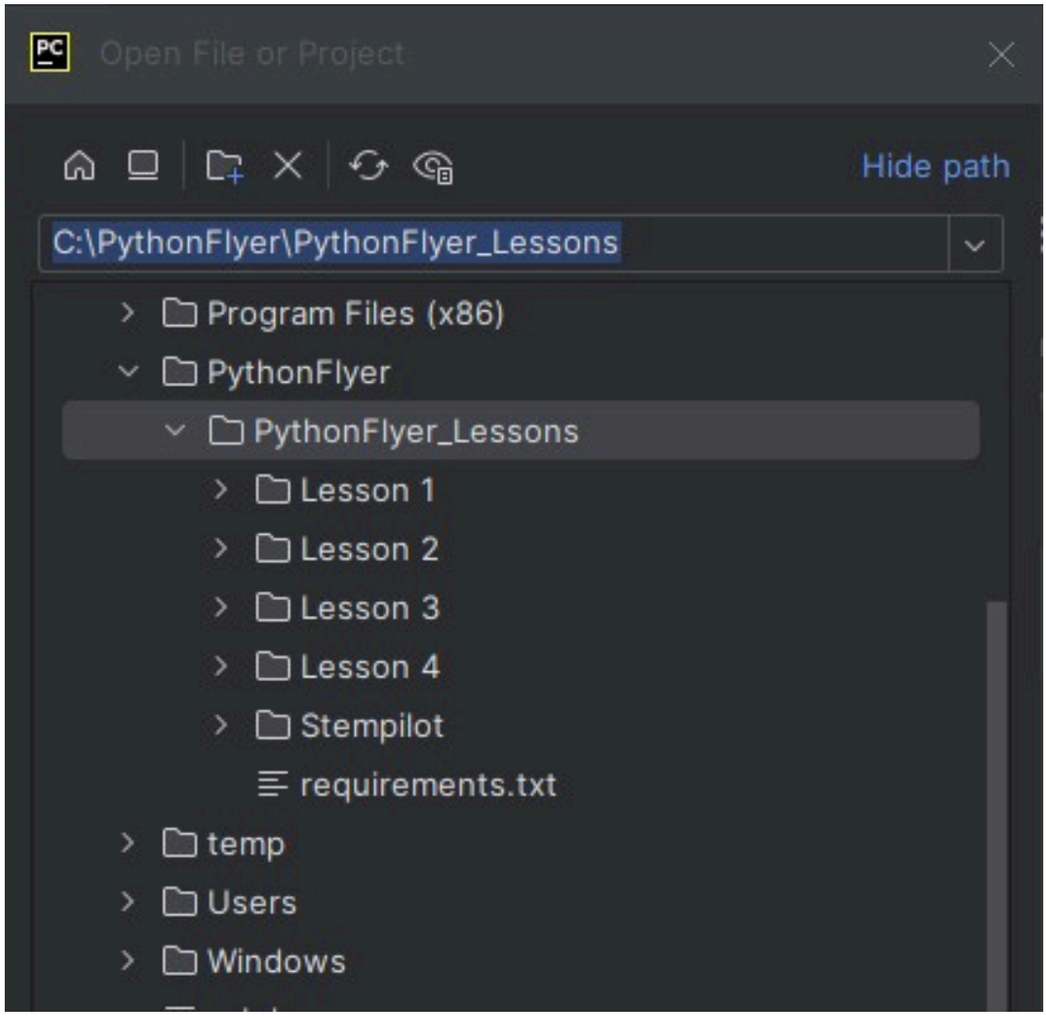
(in the example below we extracted the files to c:\PythonFlyer):



Open PyCharm and navigate to Projects on the left panel:

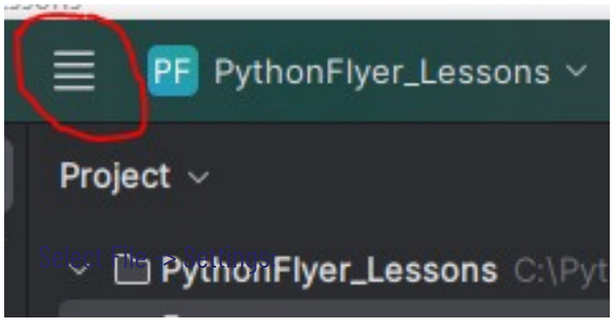


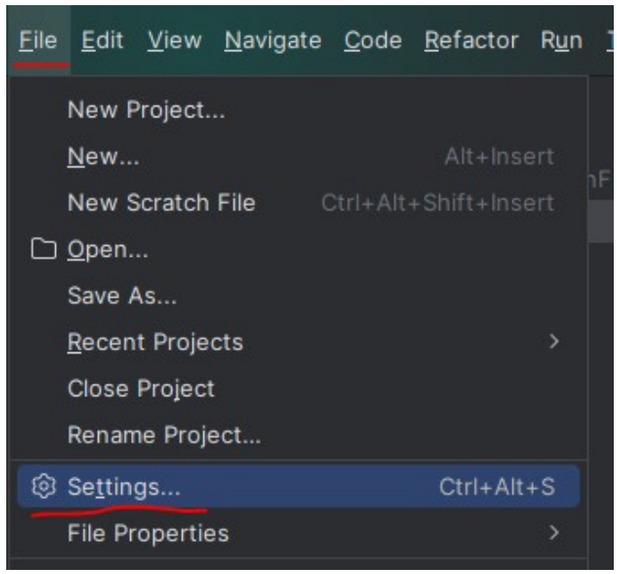
- Select Open and locate the PythonFlyer_Lessons directory that you extracted from the .zip file:



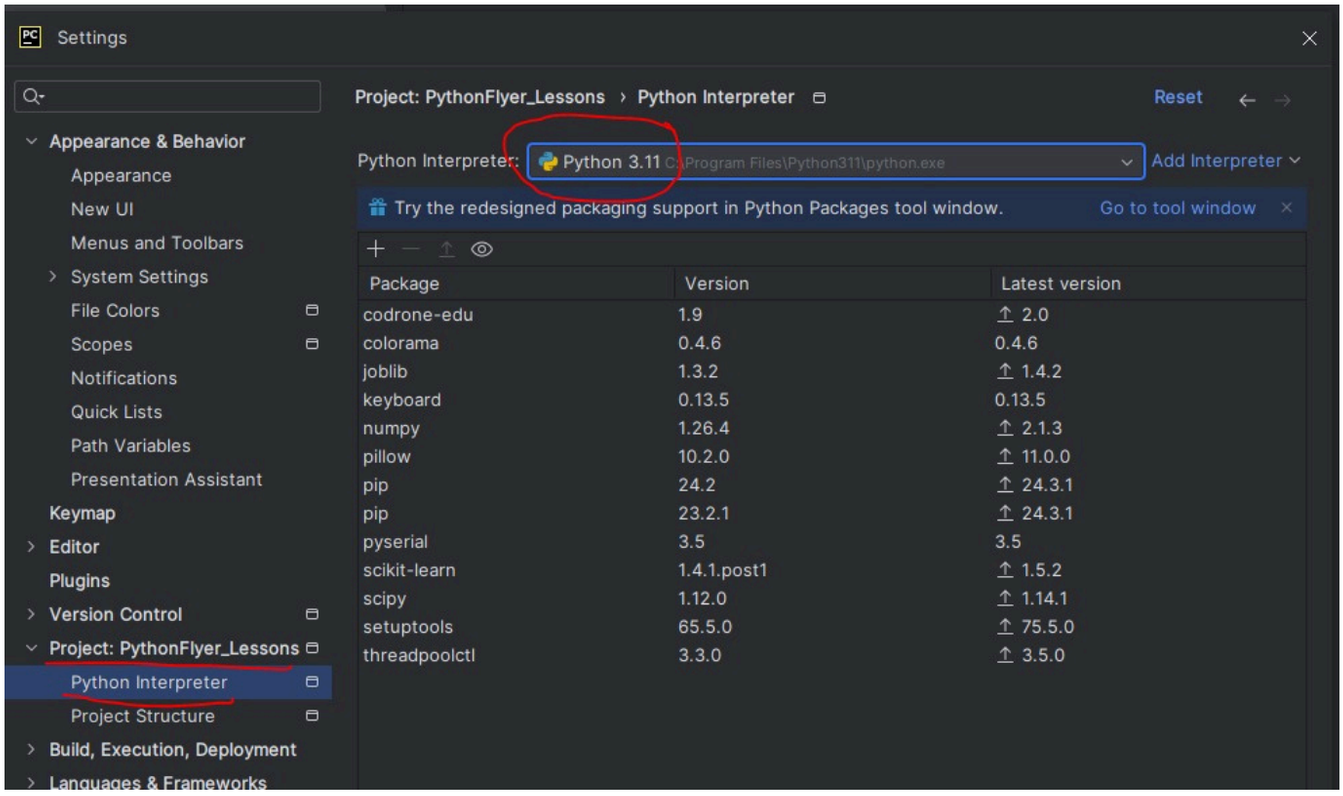
- Press the OK button:

- Click the menu button on the top left of the PyCharm screen:

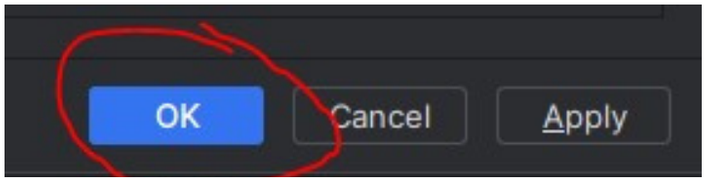




- Under Project: PythonFlyer_Lessons, select Python Interpreter. Ensure that the Python Interpreter is set to Python 3.11. If it is not, select it from the drop down:

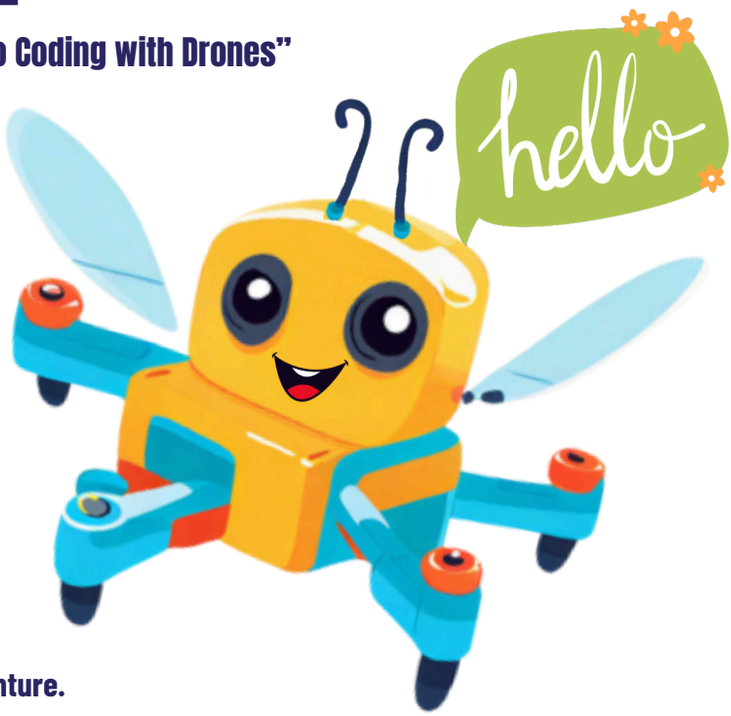


- Click the OK button on the bottom of this window:



Chapter 1

"Python Adventures: Your First Steps into Coding with Drones"



Introduction: Welcome, future Python coders! In Chapter 1, we embark on an exciting journey into the world of programming, where you'll learn the building blocks of Python and discover how to apply them to control drones. Each lesson will unlock new skills, guiding you from basic coding concepts to controlling your PythonFlyer. Let's dive into this thrilling adventure and start coding our way to the skies!

What we'll cover:

- **Lesson 1a: Kicking Off Your Python Adventure.**

Learn the fundamental principles of Python, including basic syntax and how to use the print function.

- **Lesson 1b: Unveiling the Power of Comments**

Discover the importance of comments in coding and how they help organize and clarify your code.

- **Lesson 1c: Coding Adventures**

Discovering the Power of Variables. Explore how variables work in Python and how they act as containers for storing and manipulating data in your programs.

- **Lesson 1d: Unlocking Data Magic**

Master the use of mathematical operators to perform calculations and manipulate data efficiently in your Python programs.

Lesson 1a "Kicking Off Your Python Adventure"

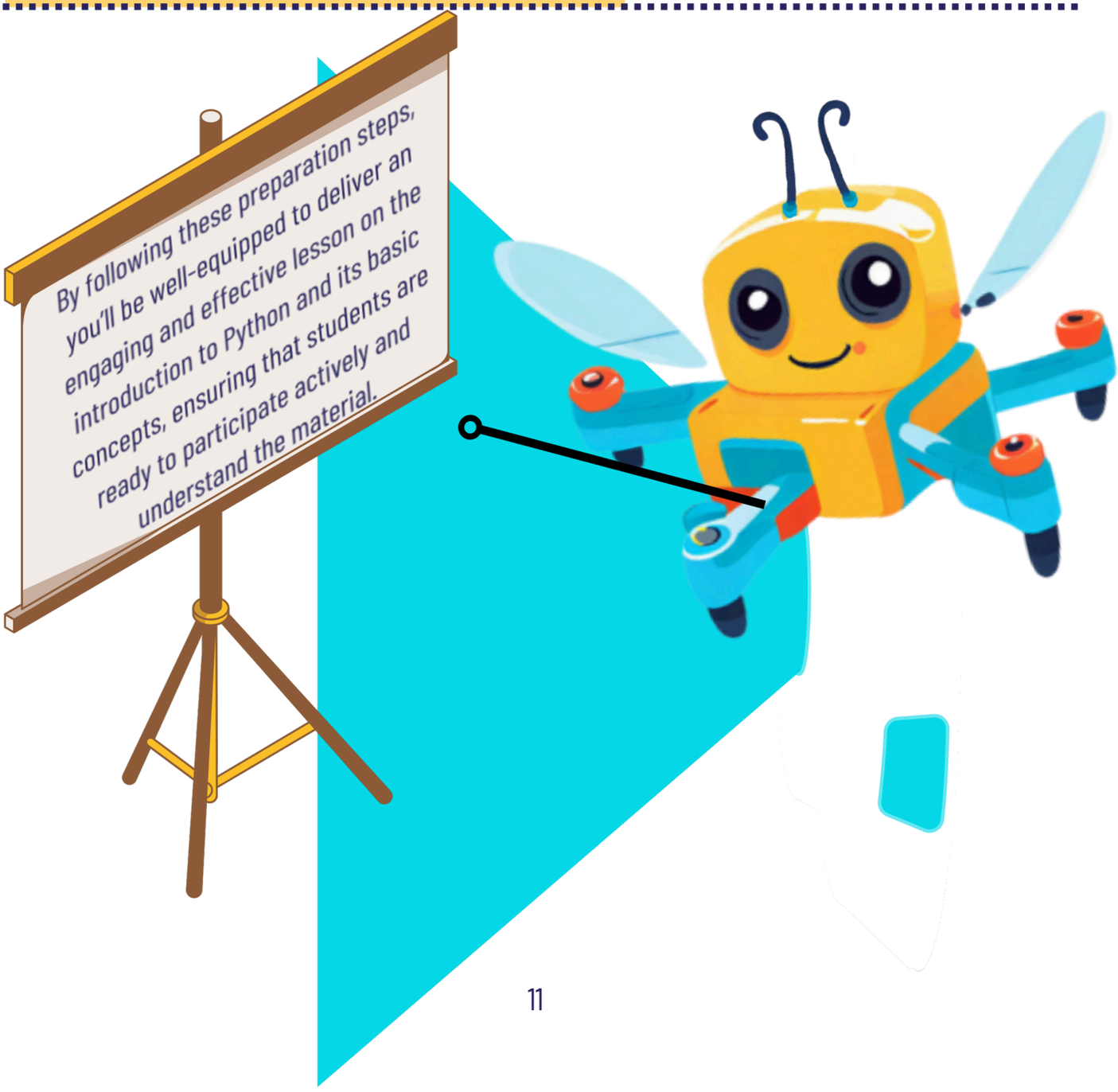
Welcome to Lesson 1a of our coding journey! Today, we're diving into the exciting world of Python, one of the most popular and beginner-friendly programming languages out there. Think of Python as the language that will help you talk to computers, and today we're going to start learning how to speak it. We're going to explore the basic building blocks of Python, starting with its "grammar"—the syntax. Just like you need to know the rules of a language to write a good story, you need to know Python's syntax to write awesome code. We'll also get hands-on with the print() function, your first tool in telling the computer what to do—whether it's showing messages on the screen or displaying the results of calculations. And the best part? You'll be doing all of this in PyCharm, a super cool coding environment that will help you write, test, and debug your Python programs. By the end of this lesson, you'll be ready to start coding on your own, with the skills to create simple programs and the knowledge to keep growing as a Python programmer.

By the end of Lesson 1a, you will be able to:

- **Understand Python's Basic Syntax:** Learn the rules of Python's "grammar" so you can write code that the computer understands.
- **Master the print() Function:** Use the **print()** function to make your code "speak" by displaying text and variables on the screen.
- **Navigate PyCharm Like a Pro:** Get comfortable with PyCharm, the tool you'll use to write and run your Python code, so you can focus on bringing your ideas to life.

Learning Aides

Laptop or PC with Python and PyCharm installed on it	Classroom
Smartboard or Projector	Classroom
Lesson Files (PythonFlyer - Lesson 1a)	USB drive/file
Printed Python Code Examples	USB drive/file



Glossary

Basic Syntax	The set of rules that defines the structure of a programming language. In Python, syntax rules govern how code must be written to be correctly interpreted and executed
print( ) function	A built-in Python function used to display text or other outputs in the console
Python Editor	A software tool specifically designed for writing and editing Python code. Provides features like syntax highlighting, code completion and debugging tools.
PyCharm	An Integrated Development Environment (IDE) specifically designed for Python development. Provides tools for writing, testing, debugging, and managing Python code.
Execution	The process of running a program or script. In Python, execution refers to the Python interpreter processing the code line by line to perform tasks.
Syntax Highlighting	A feature in code editors and IDEs that color-codes elements of the code according to their function, improving readability and helping identify errors.
Syntax Errors	Errors that occur when Python code doesn't follow the correct rules, preventing the program from running. Common causes include missing punctuation, improper indentation, or incorrect commands.
Code Completion	A feature in IDEs that suggests possible completions for partially typed code, such as keywords, function names, or variables.
Debugging Tools	Features in IDEs that help programmers find and fix errors (bugs) in their code. Includes breakpoints, step-through execution, and variable inspection.
Folders and Files	Organizational structures used to store and manage code and related resources. Folders contain files, and files contain the actual code or data.

Teacher Preparation for Lesson 1a: Introduction to Python and Basic Concepts.
Review Lesson Materials:

Familiarize Yourself with the Content: Thoroughly review the "PythonFlyer - Lesson 1a" file. Focus on the foundational concepts of Python, including syntax rules, the print() function, and basic navigation within the PyCharm IDE. Ensure you understand how these concepts will be introduced and practiced in the lesson.
Practice Writing and Executing Python Code: Write and run several simple Python scripts using the print() function. Experiment with different messages and syntax variations to anticipate potential student questions or challenges.
Prepare Visual Aids:

- **Create or Review Visual Aids:** Prepare visual aids that highlight the basic syntax of Python, the structure of a print() statement, and the layout of the PyCharm IDE. Use diagrams or screenshots to illustrate key concepts like syntax highlighting.
- **Plan for Displaying Code Examples:** Ensure you have clear, step-by-step code examples that you can display using the projector or Smartboard. These examples should start simple and gradually increase in complexity as students grasp the basics.

Set Up Classroom Technology:

- Test the Projector or Smartboard: Make sure the projector or Smartboard is functioning properly. Test the connection to your computer and verify that lesson materials and live coding demonstrations can be displayed clearly to all students.
- Prepare Student Laptops/PCs: Confirm that all students have access to laptops or PCs with Python and PyCharm installed. Ensure that the "PythonFlyer - Lesson 1a" files are accessible to all students and that they can open and run Python scripts without issues.

Plan Interactive and Hands-On Activities:

- Design Interactive Exercises: Plan exercises where students can practice writing and executing simple Python scripts using the print() function. Consider activities that allow them to experiment with different messages and observe how changes in code affect the output.

Prepare Handouts and Resources:

- Print Handouts: Ensure that any handouts required for the lesson, such as syntax reference sheets or step-by-step guides for using PyCharm, are printed and ready for distribution to students.
- Display Handout Files on Smartboard: Have the handout file displayed on the Smartboard for easy reference during the lesson, ensuring all students can follow along.

Glossary Review for Lesson 1a:

- Display the Glossary of Terms: Show the glossary of terms for Lesson 1A on the board or projector. Review each term with the class, ensuring they understand the definitions and how to use these terms when controlling the PythonFlyer. Encourage students to ask questions if they need clarification.

Review of Key Terms:

Basic Syntax: Explanation:

- **Explanation:** Explain that basic syntax refers to the set of rules that determine how code must be structured for it to be correctly interpreted by Python. Understanding these rules is crucial for writing code that runs without errors.
- **Example:** Show how Python code is structured with indentation and without semicolons at the end of statements.

print() Function:

- **Explanation:** Highlight that the print() function is used to display text or other outputs on the screen. It is one of the most basic and commonly used functions in Python.
- Syntax: print("XXXXX")
- **Replace "XXXXX" with any text you want to display on the screen.**
- **Make sure everything inside the parentheses is surrounded by quotation marks " ". (We'll learn about some exceptions to this in a later lesson.)**
- **Example:** Demonstrate with print("Hello, World!"), explaining how this command will display the text "Hello, World!" in the console.

Python Editor:

- **Explanation:** Describe a Python editor as a software tool specifically designed for writing and editing Python code. It helps in making the coding process smoother and more efficient.
- **Example:** Show how to use an editor like PyCharm or IDE, pointing out features like syntax highlighting and code completion.

Execution:

- **Explanation:** Explain that execution is the process of running a program or script, where the Python interpreter processes each line of code to perform the tasks written in the script.
- **Example:** Demonstrate the execution process by running a Python script in PyCharm.

PyCharm:

- **Explanation:** Introduce PyCharm as a powerful Integrated Development Environment (IDE) used for Python development, which includes many tools to help write, test, and debug code.
- **Example:** Show students the PyCharm interface and highlight features like the project explorer, editor window, and run/debug tools.

Syntax Highlighting:

- **Explanation:** Discuss syntax highlighting as a feature that color-codes different elements of the code to improve readability and help quickly identify errors.
- **Example:** Show how PyCharm or another Python editor highlights different elements of the code, such as keywords in one color and strings in another.

Code Completion:

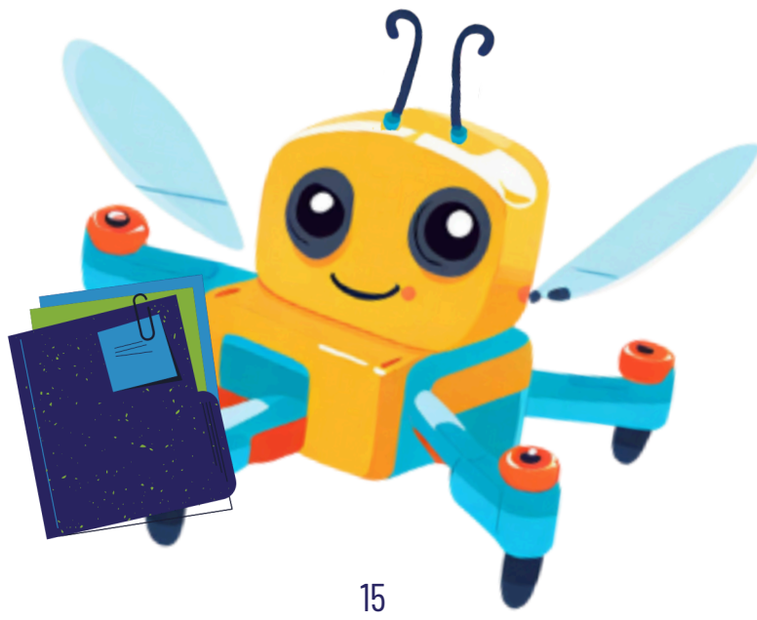
- **Explanation:** Explain that code completion is a feature in IDEs that suggests possible completions for partially typed code, helping to speed up coding and reduce errors**.**
- **Example:** Demonstrate how typing `print` in PyCharm might suggest `print()` as an auto-completion option.

Debugging Tools:

- **Explanation:** Describe debugging tools as features that help identify and fix errors in the code. These tools allow you to pause the code, inspect variables, and step through code execution to understand what's going wrong.
- **Example:** Show how to set a breakpoint in PyCharm and explain how it allows you to pause the program at a specific line to check the program's state.

Folders and Files:

- **Explanation:** Explain that folders and files are used to organize and manage code and resources. Folders group related files together, making it easier to manage large projects**.**



Lesson 1a: Kicking Off Your Python Adventure

Introduction to Python and Basic Concepts

Lesson Objectives Review:

Display the Objectives: Start by reviewing the lesson objectives with your students. This ensures they have a clear understanding of what they'll be learning and accomplishing by the end of the lesson**.**

Detailed Breakdown of Objectives:

Understand Python's Basic Syntax:

- **Explanation:** Introduce students to the fundamental rules of Python syntax, which is the set of rules that dictates how Python code is written. Understanding these basics is essential for writing code that the computer can understand and execute.
- **Example:** Show simple examples of Python syntax, such as using parentheses for functions, proper indentation, and how to structure basic code statements.

Master the print( ) Function:

- **Syntax Overview:** Teach students how to use the print() function, which is their first tool for communicating with the computer. The print() function allows them to display text and results on the screen.
- **Example:** Provide examples of how to use print() to display messages, numbers, and variables, showing how this function can be used to test and debug code.

Navigate and Use PyCharm:

- **Explanation:** Guide students through the basics of using a Pycharm. Explain how an Integrated Development Environment (IDE) like PyCharm helps them write, run, and debug their Python code efficiently.
- **Example:** Walk through the PyCharm interface, demonstrating how to create a new project, write Python code, and execute it within the editor.

Apply Basic Concepts to Write and Execute Simple Python Programs:

- **Hands-On Exercise:** Engage students in writing their first simple Python program using the print() function. Guide them through saving their script in a properly named file and executing it in PyCharm.
- **Practice:** Encourage students to experiment with modifying their print() statements to display different messages, helping them understand how changes in code affect the program's output.

This structured approach will ensure students not only understand the objectives of Lesson 1a but also see practical examples and engage in hands-on practice, reinforcing the concepts in a way that is both engaging and educational.

Flow into the Lesson Content for Lesson 1a

Now that we've covered the introduction to Python glossary, let's dive deeper into how to write and understand basic Python code. We'll start by understanding why Python's syntax is important—it's like the rules of a language that allow you to communicate with the computer. Next, we'll learn about the print() function, your first tool for making Python "speak." Finally, we'll bring it all together in a hands-on activity where you'll write simple Python code in PyCharm to display messages on the screen. **REMEMBER TO STAY FOCUSED AND FOLLOW EACH STEP CAREFULLY!**

Lesson 1a: Kicking Off Your Python Adventure: Introduction to Python Programming

Understanding Python Basics

Welcome, everyone! Today, we're going to start our journey into the world of Python programming. Python is one of the most popular and beginner-friendly programming languages, and learning it will open up a world of possibilities for you. In this lesson, we'll focus on learning the basic building blocks of Python so you can begin writing code.

Why is this important?

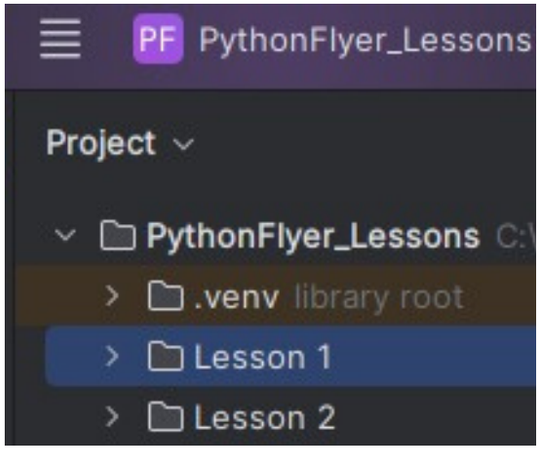
Imagine learning a new language like Spanish or French. You need to know how to form sentences and use the right words. Python is the same—you need to know how to form the correct syntax to "talk" to the computer and tell it what to do. By learning Python's basics, you'll be able to write simple programs and take the first steps toward becoming a skilled programmer.

Let's learn to print!

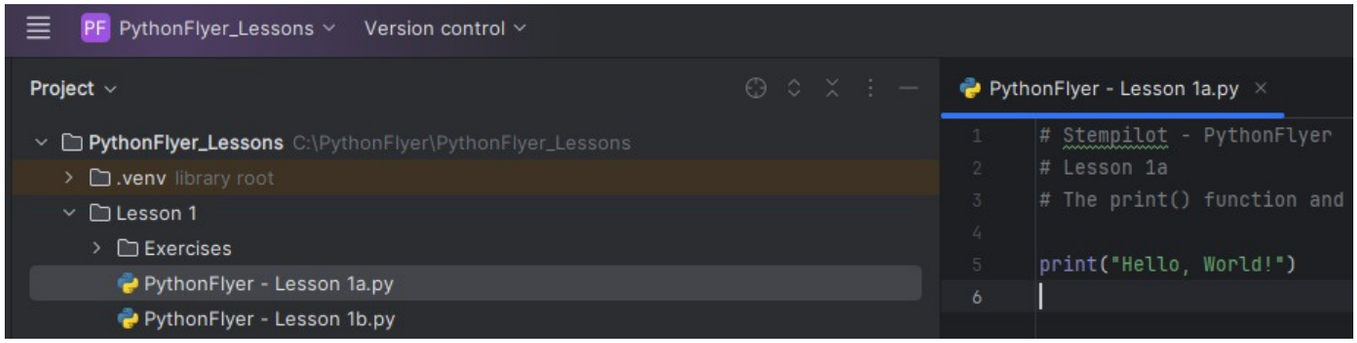
Step-by-Step Python Code Breakdown of the print() Function and Syntax Errors

Step 1: Open PythonFlyer - Lesson 1a

- **Open PyCharm: Launch PyCharm from your computer.**
- **Navigate to the "Lesson 1" folder:**



- Open the file named "PythonFlyer - Lesson 1a.py":



Now, you're ready to start coding!

The print() Function: Making Python Speak

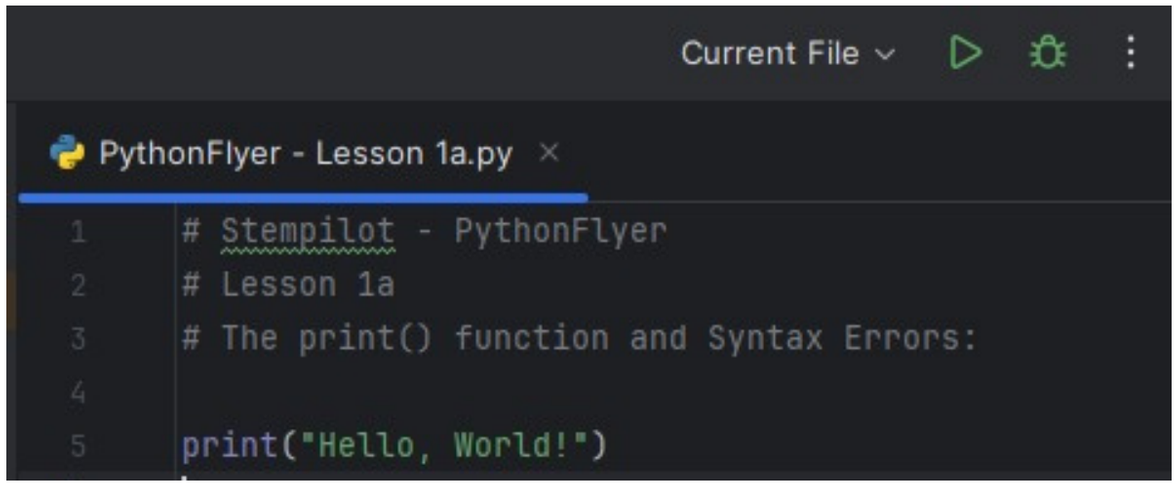
The first tool you'll learn to use in Python is the print() function. It's like having a conversation with your computer, where you can tell it to display text or numbers on the screen. The print() function is essential for showing the output of your code, whether it's a message or the result of a calculation.

Before we move forward, lets go over how to use the print() function and learn about syntax:
Syntax:
print("XXXXX")

1. Replace "XXXXX" with the text you want to show on the screen.
2. Make sure everything inside the parentheses is surrounded by quotation marks " ". (We'll learn more about exceptions in a future lesson.)

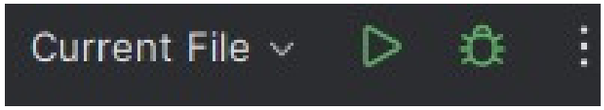
Execution:
Let's run the python script that we just opened.

- Make sure you have the correct file open:



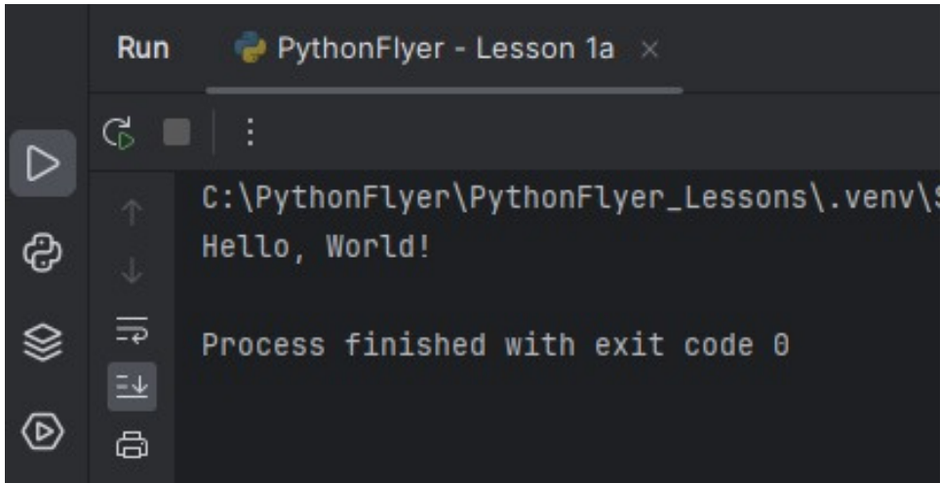
Note: Ignore the first 3 lines of the code in this file, we will be going over those lines in a different lesson. We only care about line 5 in this lesson.

- **Verify that PyCharm is set to execute on the "Current File":**



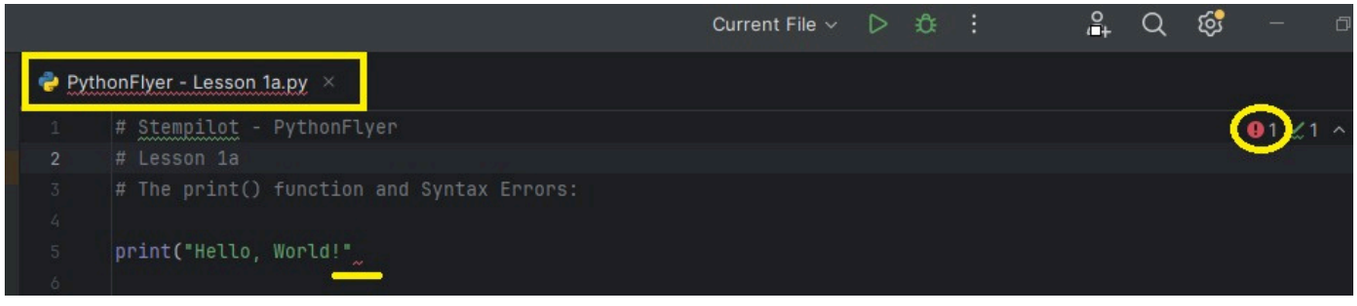
- **Click the run button:**

Observations:
You should have noticed a section appear on the bottom of your PyCharm window similar to below:

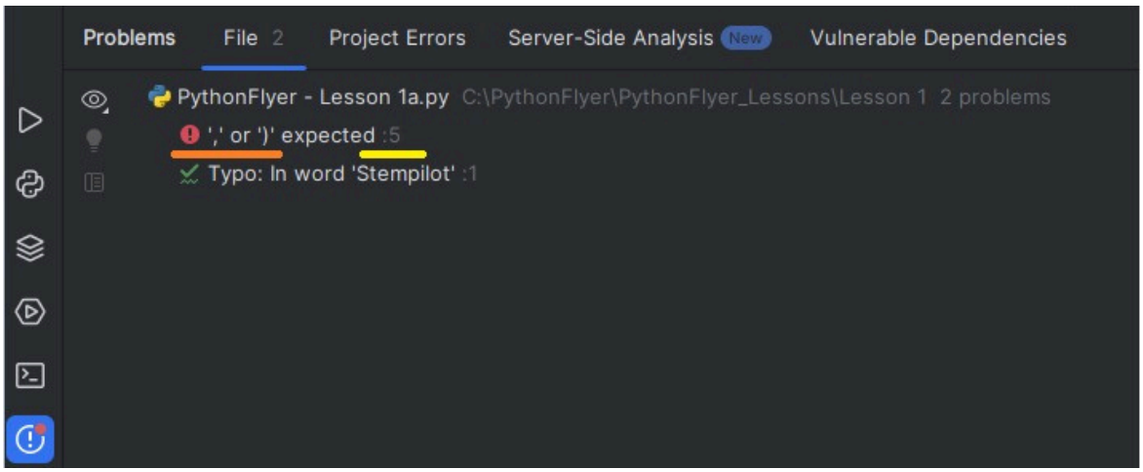


Notice the sentence "Hello, World!" was **printed** out to the screen.
Now let's edit the line with the print function and delete one of the parentheses. Your new line should look like this:

You should have noticed a red error symbol on the top right of your screen as well as a red squiggly line just after the second quotation mark and under the file name at the top of your code window, as shown below:



Please click on the red error circle on the top right of the screen. You should see the console window on the bottom of the screen change to the following (if you do not, click on the red error circle again. If you click on it too many times the error section will disappear, don't panic, just click it once more and it will return):



The section underlined in orange indicates what may be missing from your code, in this case we are missing a parenthesis ")". The section underlined in yellow indicates the line that this syntax error can be found, in this case it should be on line 5. You can ignore the second line indicating that there's a typo in the code. This is just PyCharm's spell checker not recognizing the word and has no impact on the execution of the code.

Let's put the parenthesis back, so your line should look like this:

```
5    print("Hello, World!")
```

Interactive Review:

- **Q&A Session:** Ask students to explain each term in their own words. For example, "What is basic syntax, and why is it important?" or "How does the print() function work in Python?"

Why Understanding Python's Syntax is Important

Python's syntax is like the grammar of a language. If you don't follow the rules, the computer won't understand your instructions, just like you wouldn't understand a sentence with bad grammar. Learning how to write code correctly will help you avoid errors and make sure your programs run smoothly.

Practicing Python Basics

After you understand the basics of Python's syntax, you'll start practicing writing code using the print() function. This will help you understand how Python works and get you comfortable writing and running your own programs.

Why Practice is Important

The more you practice writing Python code, the better you'll get at it. You'll begin to see how small changes in your code affect what happens when you run your program. This practice is essential for building confidence and becoming more comfortable with coding.

Real-Life Example:

Think of writing Python code like sending a text message to a friend. If you don't spell the words correctly, your friend might not understand your message. In Python, if your code isn't correct, the computer won't understand it either, and your program won't work.

Mastering Basic Python Commands

Now that we've covered the basics of the print() function, it's time to see how everything works together. Understanding how to use this function on its own is important, but combining it with other concepts in future lessons will help you create more complex programs.

Staying in Control:

When you're coding, it's important to pay attention to every detail in your syntax, whether it's a missing quotation mark or an extra space. Following Python's rules will help you avoid errors and make your code work exactly as you intend.

Safety Tips for Coding:
Just like we need to be careful with a drone, we need to be careful when writing code. Always double-check your code before running it. It's easy to make small mistakes like leaving out a quotation mark or misspelling a word, and these can cause your program to fail.

Putting It All Together
With these new skills in Python programming, you'll be able to write basic programs that display messages and interact with the computer. Whether you're writing simple code or building something more complex, these basics are the foundation for everything you'll learn in the future.

Keep Practicing:
The more you practice writing Python code, the better you'll get at it. These skills will help you as you move on to more advanced concepts, like using variables, doing calculations, and even creating interactive programs.

Remember: Mastering the basics of Python is key to becoming a skilled coder—start by learning how to use the print() function and write simple programs, and soon you'll be coding like a pro!

Wrap-Up Discussion:

- What Did You Learn Today?
  - Discuss the importance of understanding Python's basic syntax, how to use the print() function effectively, and how to fix common errors.
- **Questions to Ask the Class:**
  - Why is it important to follow Python's syntax rules closely?
  - What happens if you forget to include a parenthesis in your code?
  - How does using the print() function help you understand what your code is doing?

Challenge Activity:

Objective:

- As a final challenge, ask students to find any errors that may exist in the following snippets of code:

- Example A:
  - print(It is a nice day today)
- Example B:
  - print("Hello World"!)
- Example C:
  - print "My Name is BeeBot"
- Example D:
  - print("I hope it rains")
- Example E:
  - print("What is your name?')
- Example F:
  - print("How old are you?"
- Example G:
  - print (I Like Drones")

Answers:

- Example A: The print function is missing quotations " ".
  - It should be: print(**"**It is a nice day today**"**)
- Example B: The exclamation mark is outside of the quotations " ".
  - It should be: print("Hello World**!**")
- Example C: The print function is missing parentheses ( ).
  - It should be: print**(**"My Name is BeeBot"**)**
- Example D:
  - There is nothing wrong with this example.
- Example E: The second quotation " is a single quote '.
  - It should be: print("What is your name?**"**)
- Example F: The print function is missing the second parentheses).
  - It should be: print("How old are you?")
- Example G: The print function is the first quotation ":
  - It should be: print("I Like Drones")

Questions and Discussion for Lesson 1a: Kicking Off Your Python Adventure

Question 1: What does the print() function do in Python?
Answer: The print() function tells the computer to display information on the screen. It can be used to show text, numbers, or results from calculations. It's one of the first tools you'll use to interact with the computer in Python.

Question 2: How does adding quotation marks affect the print() function?
Answer: Quotation marks are used to tell Python that you are dealing with a string, which is just a series of text characters. Without them, Python will think you're trying to refer to something else, like a variable or command. For example, print("Hello") will work, but leaving out the quotation marks, like print(Hello), will cause an error because Python will not know what "Hello" is.

Question 3: Why do we need to use parentheses in the print() function?
Answer: Parentheses are needed in Python to tell the print() function what it should display. Everything inside the parentheses is what will show up on the screen when you run your program. Without the parentheses, Python doesn't know what you want it to do.

Question 4: What should you do if your print() statement doesn't work correctly?
Answer: First, check your syntax. Look for missing quotation marks, parentheses, or any typos. PyCharm will often highlight errors for you, and the error message can help you figure out what went wrong. Always double-check your code to see if you've followed Python's syntax rules correctly.

Question 5: How would you write a Python command to display the message "Hello, World!" on the screen?
Answer:

Explanation: This command will display "Hello, World!" on the screen when you run your code.

Question 6: After displaying "Hello, World!", you want the program to also display "Let's learn Python!" right after. How would you write the code for this?
Answer:

```
print("Hello, World!")
```

Explanation: This code will first print "Hello, World!", and then it will immediately print "Let's learn Python!" on the next line. Each print() function call displays its message in the order the code is written.

```
print("Hello, World!")
print("Let's learn Python!")
```

Suggested Discussions for Lesson 1a: Kicking Off Your Python Adventure

1. Discussion on the Importance of Syntax

Ask students:

- "What happens if you forget a quotation mark or a parenthesis when using the print() function?"
- Discussion Point: Highlight the importance of following Python's syntax rules closely. These rules are like grammar in a language—without them, the computer doesn't understand your code, and it won't work.

2. Discussion on Understanding Error Messages

Ask students:

- "How do error messages in PyCharm help you fix mistakes in your code?"
- Discussion Point: Talk about how error messages are tools to help you learn. They give hints about what went wrong in your code, like missing a parenthesis or using incorrect syntax.

3. Exploring Creative Uses of the print() Function

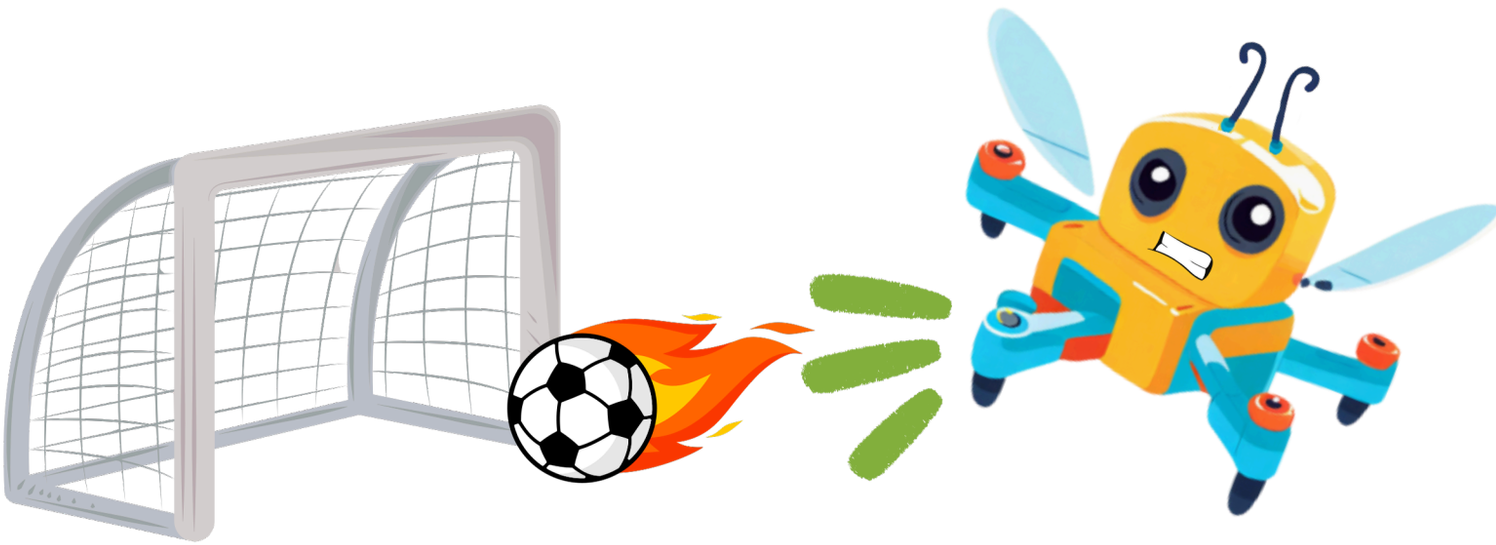
Ask students:

- "How do you think the print() function can be used in more advanced programs?"
- Discussion Point: Encourage students to think creatively. The print() function can be used to show the results of calculations, to display game scores, or even to give instructions to a user. It's one of the most versatile functions in Python and is used in almost every program.

4. Application to Real-World Scenarios

Ask students:

- "How might understanding Python's basic syntax help you with more advanced programming tasks?"
- Discussion Point: Explain how learning the basics, like using the print() function and understanding syntax, will help students as they move on to more complex topics, such as loops, functions, and variables. These basics are the foundation for all programming tasks.



Lesson 1a Recap:
Kicking Off Your Python Adventure

Today, we took our first steps into the world of Python programming, focusing on the essential building blocks that will guide us through our coding journey. Here's a quick recap:

- **Understanding Basic Syntax:** We explored the foundational rules of Python syntax, learning how to write clear and correct code that the computer can understand and execute.
- **Using the `print()` Function:** We practiced using the `print()` function, our first tool for communicating with the computer, allowing us to display text and results directly on the screen.
- **Navigating Python Editors:** We learned how to navigate and utilize Python editors, like PyCharm, to write, execute, and debug our Python code efficiently.
- **Executing Code in PyCharm:** We ran our first Python script in PyCharm, seeing firsthand how our code translates into actions on the computer.
- **Identifying Errors:** We experimented with introducing small errors, such as missing a parenthesis in the `print()` function, and learned how to recognize and fix these errors using PyCharm's debugging tools.

Keep practicing these foundational concepts to build a strong base for your programming skills. Remember, understanding the basics is key to mastering more advanced topics as we progress. Next, we'll dive deeper into coding with new concepts and challenges!

We are on our way to: Lesson 1b - Unveiling the Power of Comments!