Python Coding for Teens

Intro Unit • Grades 7–10 • 3 Weeks

Variables • Input/Output • Conditionals • Loops • Functions

15 Lesson Plans • 5 Mini-Projects • 15 Debug Challenges

Student Coding Journal • Vocabulary Reference • Final Project Rubric

No installation required — runs in any web browser!

Replit • Google Colab • Python Anywhere

UM Printables

Table of Contents

Reference & Assessment

- Vocabulary Reference Sheet
- Final Project Rubric

Review Request

Teacher Guide

How to Use This Resource

This curriculum is designed as a three-week introductory Python unit for students in grades 7–10. No prior coding experience is required. Every lesson is built to run entirely in a web browser using Replit or Google Colab—no software installation needed. The unit progresses from basic print statements through variables, input/output, conditionals, loops, and functions, giving students a solid foundation in procedural programming.

Each lesson follows a consistent structure: Warm-Up, Direct Instruction, Guided Practice, Debug This! Challenge, and Exit Ticket. Mini-Projects at the end of each week integrate cumulative skills in a creative context. The Student Coding Journal provides structured space for note-taking, code sketching, and reflection.

Technology Setup

Option A: Replit (Recommended) — Visit replit.com and create a free account. Students click "+ Create Repl," choose Python, and start coding immediately. The editor, console, and output are all visible on one screen. Replit also supports real-time collaboration for pair programming.

Option B: Google Colab — Visit colab.research.google.com and sign in with a Google account. Create a new notebook and type code into code cells. Click the play button to run each cell. Colab notebooks auto-save to Google Drive. Note: Colab uses cells rather than a script-style editor, so some instructions may need minor adaptation.

Option C: Python Anywhere — Visit pythonanywhere.com for a simple browser-based Python console. This is best for quick demonstrations but lacks a full editor for longer programs.

Unit Overview & Pacing Guide

Week	Lessons	Focus	Mini-Project
1	1–5	Python Basics: print, variables, input/output, strings	Interactive Greeting Card

2	6–10	Logic & Loops: booleans, conditionals, while loops, for loops	Choice Adventure
3	11–15	Functions & Integration: defining functions, parameters, returns, cumulative project	Personal Assistant Bot

Assessment & Differentiation

Formative Assessment: Each lesson includes an Exit Ticket to check understanding. Debug This! challenges serve as formative checks on syntax and logic. The Student Coding Journal provides written evidence of thinking and reflection.

Summative Assessment: The Final Project (Lesson 15: Personal Assistant Bot) is assessed using the included rubric. Students must demonstrate mastery of variables, input/output, conditionals, loops, and functions in a cohesive program.

Supporting Struggling Learners: Provide partially completed code templates. Pair students for guided practice. Offer debug challenges with hints visible. Allow verbal explanations of logic before coding. Use the vocabulary sheet as a reference during work time.

Extending Advanced Learners: Challenge students to add features to mini-projects (e.g., input validation, score tracking, additional menu options). Encourage independent research on topics like lists, dictionaries, or importing modules. Provide open-ended "What if?" extension prompts at the end of each lesson.

Week 1: Python Basics

In Week 1, students set up their coding environment and learn the building blocks of Python: printing output, creating variables, getting user input, and working with strings. The week culminates in Mini-Project 1: an Interactive Greeting Card that combines all new skills.

Lesson 1: Welcome to Python!

Duration: 45–60 minutes

■ Learning Objectives

- Understand what Python is and why it is widely used
- Set up a coding environment in Replit or Google Colab
- Write and run a first Python program using print()
- Explain the concepts of syntax and output

■ Key Vocabulary

program — A set of instructions that tells a computer what to do

Python — A popular, beginner-friendly programming language

syntax — The rules for how code must be written so the computer understands it

output — The result that a program displays to the user

interpreter — Software that reads and runs Python code line by line

Warm-Up (5 min)

Ask students: *"What do you think coding is? Have you ever seen code before?"* Allow 2–3 minutes for pair discussion, then share out. Record responses on the board. Explain that today they will write their very first line of code.

Direct Instruction (15 min)

What is Python? Python is one of the most popular programming languages in the world. It is used for web development, data science, artificial intelligence, game design, and more. Python is known for being readable—its code looks a lot like English, making it a great first language to learn.

Setting Up: Walk students through creating an account on Replit or opening Google Colab. Demonstrate how to create a new Python file/notebook and how to run code. Point out the editor area (where you type) and the console (where output appears).

The print() Function: The first command every programmer learns:

```
print("Hello, World!")
print("Welcome to Python!")
print("My name is ___")
```

Explain each part: `print` is a function name, the parentheses hold the input (called an **argument**), and the quotation marks wrap the text (called a **string**). Python displays whatever is inside the quotes on the screen. The quotation marks themselves do NOT appear in the output. Emphasize: every opening parenthesis and quote must have a matching closing one.

Guided Practice (10 min)

Students type and run the following programs. After each, ask them to predict the output before running:

```
# Program A
print("Python is awesome!")
print("Let's learn to code!")

# Program B
print("Line 1")
print("Line 2")
print("Line 3")
```

Challenge: Can you make Python print your full name on one line and your school on the next?

■ Debug This! — Challenge 1

```
print("Hello World"
```

Hint: Look closely at the parentheses.

Exit Ticket (5 min)

Write a program that prints your name on one line and your favorite food on the next. Show your running program to the teacher before leaving.

Extension: *Can you make your program print a decorative border using symbols like = or * ?*

Lesson 2: Variables — Storing Information

Duration: 45–60 minutes

■ Learning Objectives

- Create and assign variables in Python
- Identify and use four data types: int, float, str, bool
- Follow Python variable naming rules and conventions
- Display variable values using print()

■ Key Vocabulary

variable — A named container that stores a value in a program

assignment — Storing a value in a variable using the = sign

data type — The kind of value a variable holds (int, float, str, bool)

integer (int) — A whole number without a decimal point (e.g., 14, -3, 0)

float — A number with a decimal point (e.g., 3.14, -0.5, 2.0)

string (str) — Text enclosed in quotation marks (e.g., 'hello', '123')

boolean (bool) — A value that is either True or False

Warm-Up (5 min)

Display a real-world analogy: *"A variable is like a labeled box. You write a name on the box and put something inside. When you need it later, you look for the label."* Ask: If you had boxes labeled 'name', 'age', and 'happy', what would you put inside each?

Direct Instruction (15 min)

Creating Variables:

```
# Quiz Generator
# Starter Code

def ask_question(question, correct_answer):
"""Asks a question, returns 1 if correct, 0 if wrong."""
guess = input(question + " ")
if guess.lower() == correct_answer.lower():
print("Correct!")
return 1
else:
print(f"Wrong! The answer was {correct_answer}.")
return 0

# Main program
print("=== Science Quiz ===")
score = 0

# Ask three questions and add to the score
score += ask_question("What planet is known as the Red Planet?", "Mars")
score += ask_question("What is the chemical symbol for water?", "H2O")
score += ask_question("How many legs does an insect have?", "6")

# Print final score
print(f"\nFinal Score: {score}/3")

# --- YOUR CUSTOMIZATIONS BELOW ---
# Can you add at least 5 more questions?
# Can you give harder questions more points?
# Can you add an A/B/C/D grading scale at the end?
```

Mini-Project 5: Personal Assistant Bot (Final Project)

Concepts Used: Functions, parameters, return values, while loops, if/elif/else, input/output, variables

```
# Personal Assistant Bot
# Starter Code

def greet():
"""Greets the user by name."""
name = input("What is your name? ")
print(f"Hello, {name}! I am your assistant bot.")

def math_helper():
"""Adds two numbers together."""
print("--- Math Helper ---")
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
total = num1 + num2
print(f"The sum of {num1} and {num2} is {total}.")

def main():
"""Main program loop."""
print("=== Personal Assistant Bot ===")
choice = ""
while choice != "3":
print("\nMenu:")
print("1. Greet me")
print("2. Math Helper")
print("3. Quit")
choice = input("Choose an option (1-3): ")

if choice == "1":
greet()
elif choice == "2":
math_helper()
elif choice == "3":
print("Goodbye! Have a great day!")
else:
print("Invalid choice. Please enter 1, 2, or 3.")

# Run the program
main()

# --- YOUR CUSTOMIZATIONS BELOW ---
# Add at least one custom feature function (e.g., coin flipper, temperature
converter).
# Add it to the menu!
# Make sure your menu loop updates correctly.
```

Week 3 • Lesson 15: Final Project — Personal Assistant Bot • Notes & Code Sketches

■ Key Notes

Write down the key concepts and vocabulary you learned today...

■ Code Sketches

Draft your code ideas here before typing them on the computer:

```
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
# .....................................................
```

Week 3 • Lesson 15: Final Project — Personal Assistant Bot • Reflection & Exit Ticket

■ Debug Reflection

Debug Challenge 15 was an infinite loop. Why did the menu keep printing forever, and how did you fix it?

■ Exit Ticket / Reflection

Describe the custom feature you added to your bot. How does it work (what inputs does it need, what does it return)?

■ Questions I Still Have

Is there anything confusing? Write it down to ask the teacher or look up later.

Vocabulary Reference Sheet

Keep this reference sheet handy while coding! It contains all the key vocabulary terms from the unit, organized by week.

Week 1: Python Basics

Term	Definition
program	A set of instructions that tells a computer what to do
Python	A popular, beginner-friendly programming language
syntax	The rules for how code must be written so the computer understands it
output	The result that a program displays to the user
interpreter	Software that reads and runs Python code line by line
variable	A named container that stores a value in a program
assignment	Storing a value in a variable using the = sign
data type	The kind of value a variable holds (int, float, str, bool)
integer (int)	A whole number without a decimal point (e.g., 14, -3, 0)
float	A number with a decimal point (e.g., 3.14, -0.5, 2.0)
string (str)	Text enclosed in quotation marks (e.g., 'hello', '123')
boolean (bool)	A value that is either True or False
input	A function that pauses the program and waits for the user to type something
type conversion	Changing a value from one data type to another (e.g., str to int)
concatenation	Joining strings together using the + operator
prompt	The message displayed to the user by the input() function
f-string	A string prefixed with f that embeds variables inside curly braces: f"Hello, {name}!"
method	A function that belongs to an object, called with dot notation (e.g., name.upper())
index	The numbered position of a character in a string, starting from 0

string repetition	Using * to repeat a string (e.g., '=' * 20)
debugging	The process of finding and fixing errors in code

Week 2: Logic & Loops

Term	Definition
boolean expression	An expression that evaluates to either True or False
comparison operator	A symbol that compares two values (==, !=, <, >, <=, >=)
logical operator	A keyword that combines or modifies boolean expressions (and, or, not)
conditional	A statement that runs code only when a condition is True
if statement	The first condition checked in a conditional block
elif statement	An additional condition checked if previous conditions were False
else statement	A fallback that runs if all previous conditions were False
indentation	The spaces at the beginning of a line that group code in Python
code block	A group of statements that run together, defined by indentation
branching	A program structure where different choices lead to different outcomes
nested conditional	An if statement placed inside another if statement
path	A specific sequence of choices and outcomes in a branching program
while loop	A loop that repeats as long as its condition is True
infinite loop	A loop that never ends because its condition never becomes False
counter	A variable that tracks the number of times a loop has run
iteration	One complete execution of the code inside a loop body
increment	Increasing a variable's value (e.g., count = count + 1)
for loop	A loop that iterates over a sequence (like a range of numbers)
range()	A function that generates a sequence of numbers for a for loop
iterate	To repeat a process for each item in a sequence
step	The amount by which a loop counter increases or decreases each iteration

Week 3: Functions & Beyond

Term	Definition
module	A pre-written Python file containing functions you can use in your program
import	A keyword used to load a module into your program
random module	A Python module that generates random numbers and choices
random.randint(a, b)	A function that returns a random integer between a and b (inclusive)
function	A named, reusable block of code that performs a specific task
define	To create a function using the def keyword
call	To execute a function by writing its name followed by parentheses
docstring	A string literal on the first line of a function that describes what it does
parameter	A variable listed in the function definition that receives a value
argument	The actual value passed to a function when it is called
return	A keyword that sends a value back from a function to the caller
return value	The data that a function sends back after executing
score	A variable that tracks the number of correct answers
case-insensitive	Treating uppercase and lowercase letters as equivalent
modular design	Breaking a program into smaller, reusable functions
menu-driven	A program that displays options and lets the user choose what to do
application	A complete, usable software program

Final Project Rubric: Personal Assistant Bot

Use this rubric to assess the Lesson 15 Final Project. Students must demonstrate mastery of variables, input/output, conditionals, loops, and functions in a cohesive, interactive program.

Code Functionality	Program runs flawlessly with zero errors. All features work as described. Handles unexpected input gracefully.	Program runs with no major errors. All required features work. May have minor issues with edge cases.	Program runs but has noticeable bugs or crashes on some inputs. Some features are incomplete.	Program fails to run or crashes immediately. Core features are non-functional.
Variables & Input/Output	Uses clear, descriptive variable names. Flawlessly uses input(), type conversion, and formatted f-strings for output.	Uses appropriate variables. Uses input() and output correctly. Minor formatting or conversion issues.	Variable names are unclear. Struggles with input type conversion or string formatting.	Missing variables or input/output. Code relies on hardcoded values only.
Conditionals (if/elif/else)	Uses if/elif/else logic perfectly. Menu choices route correctly. Includes input validation for invalid choices.	Uses conditionals correctly for the most part. Menu works but may lack input validation.	Flawed conditional logic (e.g., using if instead of elif, missing else). Menu choices overlap or fail.	Missing conditionals entirely, or logic is fundamentally broken.
Loops (while/for)	While loop runs the menu perfectly without infinite loops. Loop updates correctly. Uses for loops where appropriate.	Menu loop works generally well. May have a minor infinite loop issue or awkward loop structure.	Loop logic is flawed; causes infinite loops or exits too early. Does not return to the menu correctly.	Missing loop, or loop prevents the program from functioning.
Functions (def, params, return)	Creates modular functions for all features. Uses parameters and return values effectively. Includes docstrings.	Uses functions to separate features. Uses parameters and returns, but may have minor structural issues.	Functions exist but are poorly structured (e.g., missing returns, using print instead of return, no parameters).	No functions used, or functions are defined but never called.
Creativity & Extensions	Added 2+ original custom features beyond requirements. Highly creative and personalized.	Added 1 original custom feature. Shows effort in personalizing the bot.	Only implemented the starter code requirements with no custom features.	Did not complete the starter code requirements; no custom features.
Code Readability & Comments	Code is beautifully organized. Clear comments and docstrings explain all logic. Consistent indentation.	Code is readable. Basic comments are present. Indentation is mostly consistent.	Code is messy or confusing. Few comments. Inconsistent indentation.	No comments. Code is unreadable. Indentation errors present.

Grading Scale

A (90-100%): 25-28 points total

B (80-89%): 22-24 points total

C (70-79%): 19-21 points total

D (60-69%): 17-18 points total

F (Below 60%): Below 17 points

Thank You!

Thank you for choosing this curriculum resource! I pour countless hours into designing high-quality, classroom-tested materials that save you time and engage your students.

■ Your Feedback Matters! ■

If you found this resource helpful, please consider taking a minute to leave a rating and review on Teachers Pay Teachers.

Your feedback not only helps me improve but also helps other educators find quality resources for their classrooms!

Thank you for supporting UM Printables!

UM Printables

Empowering Educators, Engaging Students.