BEGINNER'S TOOLKIT - BASIC PYTHON CALCULATOR PROJECT

Prompt-Powered Kickstart: Building a Beginner's Toolkit for Python Programming

Title & Objective

Title:

Getting Started With Python - Building a Basic Calculator (Beginner Toolkit)

Objective:

The objective of this toolkit is to guide beginners through the process of installing Python, understanding its basic syntax, and running a functional calculator program. The project demonstrates how Generative AI prompts can support learning, reduce development time, and improve code clarity and debugging efficiency.

Technology Overview

What is Python?

Python is a high-level, general-purpose programming language widely used for web development, automation, data science, machine learning, and application development. It is known for its simple syntax and strong community support.

Why Python?

- Beginner-friendly
- Highly versatile
- Cross-platform support
- Large library ecosystem

- Heavy use in industry

Real-World Application Example

YouTube uses Python to power video recommendation, classification, and metadata processing services, enabling efficient content delivery at scale.

System Requirements

Supported Operating Systems:

- Windows
- Linux
- macOS

Required Tools:

- Python 3.9 or higher
- Text editor (VS Code recommended)
- Terminal or command prompt

Installation & Setup Instructions

✔️ Step 1: Verify Python Installation

Open a terminal and run:

```
python --version
```

If Python is installed, the version number will display.

✔️ Step 2: Install Python (If required)

Download from:
https://www.python.org/downloads/

Follow the installer prompts and ensure the option **“Add Python to PATH”** is checked.

✔️ Step 3: Create the Project File

Open your preferred code editor and create a file named:

```
calculator.py
```

✔️ Step 4: Paste the Code Below

```
def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b

print("=== BASIC CALCULATOR ===")
print("Select an operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

choice = input("Enter choice (1/2/3/4): ")

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if choice == '1':
    print("Result:", add(num1, num2))
elif choice == '2':
    print("Result:", subtract(num1, num2))
elif choice == '3':
    print("Result:", multiply(num1, num2))
elif choice == '4':
    print("Result:", divide(num1, num2))
else:
```

```
    print("Invalid input")
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b
```

Step 5: Run the Program

In the terminal:

```
python calculator.py
```

Minimal Working Example

📌 Expected Functionality

The program should:

- Prompt the user to choose a math operation
- Accept two numbers
- Return the operation result

📌 Sample Output

```
=== BASIC CALCULATOR ===
Select an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 3
Enter first number: 6
```

```
Enter second number: 7
Result: 42
```

AI Prompt Journal - Learning With AI

Below are the prompts used and their impact on learning and development.

Prompt #1

Prompt:
"Explain how to build a simple calculator in Python using functions."

AI Response Summary:
The model outlined function-based code structure for add, subtract, multiply, and divide.

How it helped:
It provided direction on function organization, reducing guesswork and improving modularity.

Prompt #2

Prompt:
"I'm getting TypeError when performing calculations. What could be the issue?"

AI Response Summary:
The model explained that input values were strings, and conversion to float or int was required.

Fix Applied:
Added:

```
float(input())
```

Prompt #3

Prompt:
"Rewrite my calculator code to prevent division-by-zero errors."

AI Response Summary:
It provided conditional validation logic used in the divide() function.

How it helped:
Improved error handling and made program output more robust.

Reflection on AI Assistance

Using generative AI significantly accelerated my learning curve. It helped me:

- Debug faster
- Understand syntax errors
- Produce cleaner code
- Learn step-by-step

Common Issues & Fixes

Issue	Cause	Fix
TypeError on calculations	Input treated as string	Convert to float/int
Division crash	User enters 0	Added validation
Incorrect output	Operation choice mismatch	Conditional checks

References

- Official Python Docs — https://docs.python.org
- W3Schools Python Guide
- StackOverflow Community Threads
- AI prompts and responses