Ankara Medipol University

TYPE SPEED CALCULATOR

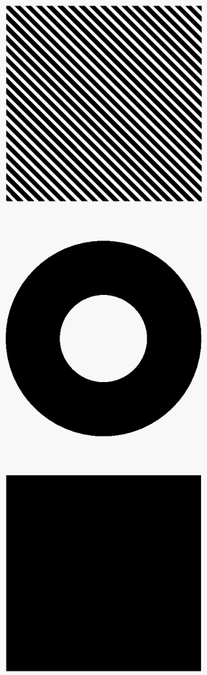
with python

Arife Zeynep Muratoğlu 64220048
Berre Sümeyye Tandoğan 64210036
Zeynep Çalapkulu 64210019
Nefise Buse Uzun 64220041

Content

Introduction

- Programming languages are the foundation of modern software development.
- Python stands out due to its simplicity, versatility, and rich ecosystem.
- This project focuses on Python through the development of a "Type Speed Calculator," showcasing its ability to create user-friendly and efficient applications.
- The project highlights Python's strengths, including its clean syntax, robust standard library, and GUI tools like Tkinter.



History and Evolution

Python was created by Guido van Rossum and first released in February 1991. Inspired by the ABC programming language, he aimed to develop a simple, readable, and extensible language. The name "Python" was derived from the British comedy show Monty Python’s Flying Circus.

1991

Python 0.9.0:
First public release with core features like exception handling, functions, and indentation-based syntax.

1994

Python 1.0:
Official release, introducing lambda functions, map(), filter(), and reduce()

2000

Python 2.0:
Added list comprehensions, garbage collection, and partial Unicode support but had design inconsistencies.

2008

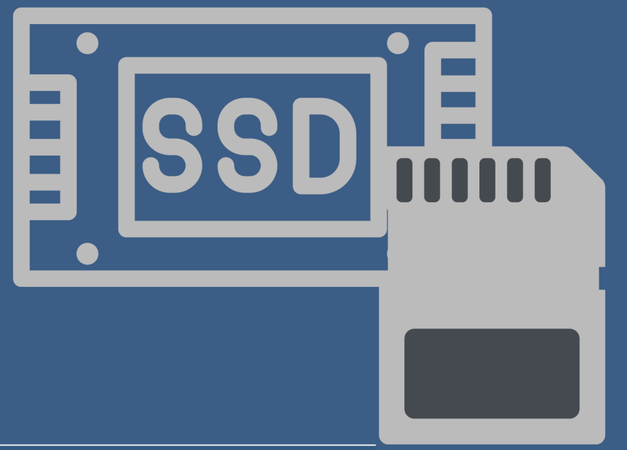
Python 3.0:
Backward-incompatible update with improved Unicode handling and modernized syntax (e.g., print() as a function).

2020

End of Python 2 Support: Developers fully transitioned to Python 3.

Core Features

Readability	• Python prioritizes readability, making code easy to understand and maintain. • It uses English-like syntax and mathematical influences. • This emphasis on simplicity reduces the learning curve for beginners.
Syntax	• Python has a clean and simple syntax. • It doesn’t require semicolons or parentheses to end statements; new lines are used instead. • Indentation defines code blocks, and incorrect indentation leads to syntax errors. • The standard indentation is four spaces.
Paradigms	• Procedural Programming: Suitable for scripting and task automation. • Object-Oriented Programming (OOP): Uses classes and objects to structure and organize code. • Functional Programming: Supports higher-order functions and lambda expressions



Core Features

Memory Management	• Python uses automatic memory management with a garbage collector to handle memory allocation and deallocation. • Reference Counting: Tracks object references and frees memory when there are no references left. • Cycle Detection: Handles cyclic references that reference counting alone cannot resolve.
Concurrency	• Python supports concurrent programming to manage multiple tasks efficiently. • Threading: Allows creating lightweight threads for multitasking. • Asyncio: Enables asynchronous programming for cooperative multitasking.
Indentation	• Indentation is essential in Python to define code blocks. Missing or incorrect indentation causes syntax errors. • The standard practice is to use four spaces for indentation.
Comments	Python supports comments to improve code readability: • Single-line comments: Use the # symbol. • Multi-line comments: Use triple quotes (''' or """) for documentation or explanations.

Ecosystem and Libraries

Python’s vast ecosystem and extensive library support make it one of the most powerful programming languages. It provides a rich set of built-in modules and third-party libraries that simplify development across multiple domains.

1.Standard Library

Python comes with a comprehensive standard library that supports:

✅ File Handling (os, shutil) – Manage files and directories easily.

✅ Data Processing (csv, json, xml) – Work with structured data formats.

✅ Networking (socket, requests) – Build network applications.

✅ Mathematical Operations (math, random, decimal) – Perform complex calculations.

2. Popular Third-Party Libraries

Python's ecosystem is enriched by thousands of third-party libraries, covering diverse fields:

📌 Web Development – Django, Flask, FastAPI

📌 Data Science & Machine Learning – NumPy, Pandas, TensorFlow, Scikit-learn

📌 Automation & Scripting – Selenium, PyAutoGUI, Paramiko

📌 Cybersecurity & Hacking – Scapy, PyCryptodome

📌 Game Development – Pygame, Panda3D

3. Package Management

Python simplifies dependency management through tools like:

- pip – The standard package installer for Python.
- conda – Used in data science for managing packages and environments.
- virtualenv – Creates isolated environments for projects.

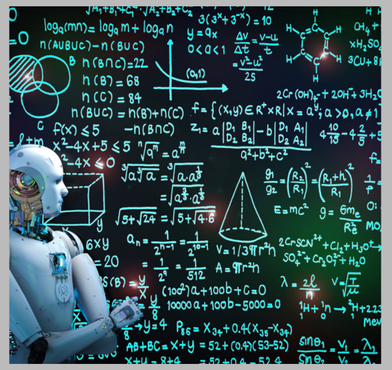
4. Community and Open Source Contributions

Python’s active open-source community constantly enhances its ecosystem. With millions of contributors, libraries are frequently updated, making Python a reliable and future-proof language.

Use Cases and Industry Adoption of Python

Web Development

Frameworks like Django and Flask power scalable platforms for companies like Instagram and Spotify



Data Science and Machine Learning

Netflix and Uber use Python for data analysis, machine learning, and AI-driven recommendations.

Automation and Scripting

Automates repetitive tasks, such as Reddit's post ranking system and Dropbox's file synchronization.

Scientific Computing

Used in research by organizations like NASA and CERN for simulations and data analysis.

Use Cases and Industry Adoption of Python

Game Development

Games like Civilization IV and Eve Online incorporate Python for logic and in-game tools.

DevOps and Cloud Computing

Google Cloud and AWS utilize Python for automation, API integrations, and serverless applications.

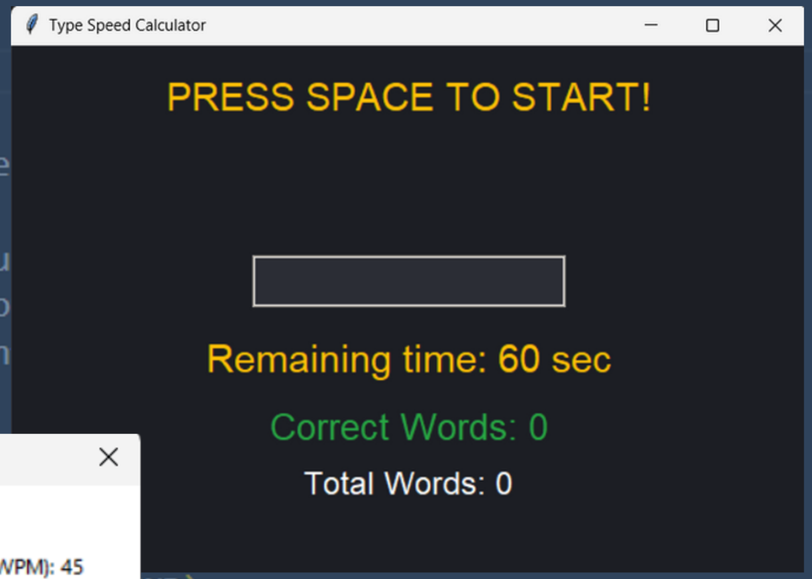
Cybersecurity

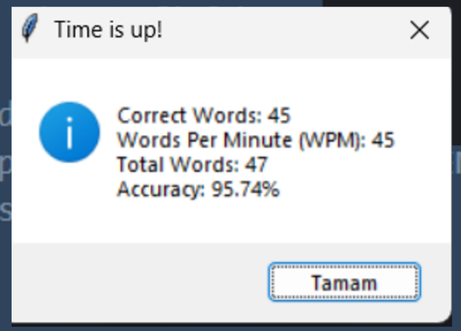
Python-based tools like Metasploit and Wireshark aid in penetration testing and network analysis.

Education and Training

Python is a beginner-friendly language, used in courses by MIT and online platforms like Codecademy.

```
    if start_time is None:
        start_time = time.time()
        instruction_label.destroy()
        start_timer()
        display_new_word()

    user_input = input_field.get().strip()
    total_word_count += 1

    if not user_input:
        return "break"

    event.widget.delete

    if user_input == cu
        correct_word_co
        input_field.con
    else:

    upd
    inp                                    ND)
    dis


def validate_partial_input(event):
    user_input = input_field.get().strip()
    if current_word and current_word.startswith(user_input):
        input_field.config(foreground="green")
    else:
        input_field.config(foreground="red")
```





Project Implementation

This project implements a Type Speed Calculator using Python, designed to measure a user's typing speed and accuracy. The application displays random words for the user to type within a one-minute time limit, then calculates wpm, accuracy percentage and real time feedback.

Project Implementation

How It Works

1️⃣ Displays a word for the user to type.
2️⃣ User presses the spacebar to submit the word.
3️⃣ The program tracks:
✔️ Correctly typed words
✔️ Total words typed
✔️ Accuracy percentage
✔️ Words Per Minute (WPM)
4️⃣ Results are displayed at the end in a message box.

Key Variables in the Code

- WORDS – Stores the list of words loaded from a file.
- correct_word_count – Tracks the number of correctly typed words.
- total_word_count – Tracks the total number of typed words.
- start_time – Records the moment the user starts typing.
- time_left – Tracks the remaining time (initially 60 seconds).
- current_word – Holds the current word to be typed.

Libraries Used

- Tkinter – Creates the GUI elements.
- Random – Selects random words for typing.
- Time – Tracks countdown and game duration.
- ttk (Themed Tkinter Widgets) – Enhances the UI design

Project Implementation

Key Functions in the Code

- load_words_from_file(file_path) – Loads a list of words from a file.
- start_timer() – Decrements time_left and updates the timer.
- end_game() – Stops the game, calculates accuracy and WPM, and displays results.
- handle_space_press(event) – Checks user input, updates the word count, and highlights correct/incorrect input.
- validate_partial_input(event) – Validates partial input as the user types.
- display_new_word() – Selects and displays a new word.
- update_labels() – Updates score labels (correct words, total words).

User Interface (UI) Setup

Instruction Label – Displays game instructions.
Word Label – Shows the current word to type.
Input Field – Where users type words.
Timer Label – Displays the remaining time.
Correct Words Label – Shows the count of correct words.
Total Words Label – Shows the total words typed.

Comparison and Evaluation of Python and LISP

Python and LISP are two programming languages with distinct features and use cases, each excelling in particular domains of software development.

Python Vs LISP Comparison			
	Criteria	Python	LISP
1	Syntax	Clean and highly readable, beginner-friendly.	Heavily reliant on nested parentheses, ideal for symbolic computation.
2	Performance	Slower for intensive tasks but boosted by libraries like NumPy and Cython.	Efficient for recursive algorithms and symbolic computations.
3	Community Support	Massive and active community with extensive resources and libraries.	Smaller but passionate community, fewer modern resources.
4	Applicability	Versatile, used in web development, data science, machine learning, and automation.	Focused on AI research and academic settings; limited industry use.

Conclusion

The Type Speed Calculator project demonstrated Python’s versatility and practicality in building interactive applications. Through this implementation, we explored:

✓ Python’s history, core features, and industry applications

✓ GUI development with Tkinter

✓ Real-time functionality using randomization and time-based operations

By comparing Python with LISP, we highlighted Python’s readability, extensive use cases, and strong community support, reinforcing its position as a powerful general-purpose language. This project showcased Python’s adaptability across various domains, making it a valuable skill in modern software development.

Thank You!