Python with Generative AI

Learning Objectives

1. Identify and define key Python syntax and data types.
2. Explain how loops, conditionals, and functions work.
3. Describe how to use ChatGPT for coding assistance.
4. Write Python scripts for basic tasks.
5. Use ChatGPT to debug and refine code.
6. Evaluate ChatGPT suggestions for accuracy and relevance.
7. Critique code for readability and efficiency.
8. Design Python programs to solve real-world problems.
9. Integrate ChatGPT into coding workflows.

Challenges with Learning Programming

1. Abstract Thinking and Logic
2. Syntax and Semantics
3. Debugging and Error Handling
4. Algorithmic Thinking
5. Understanding of Computational Concepts
6. Overcoming the “Magic” Perception
7. Fear of Failure
8. Lack of Immediate Feedback or Clarity
9. Pacing and Learning Curve
10. Lack of Immediate Real-World Applications
11. Overwhelming Choice of Tools and Languages
12. Lack of Strong Foundational Math Skills

Basic Syntax

This # is a **comment.**
The computer ignores it

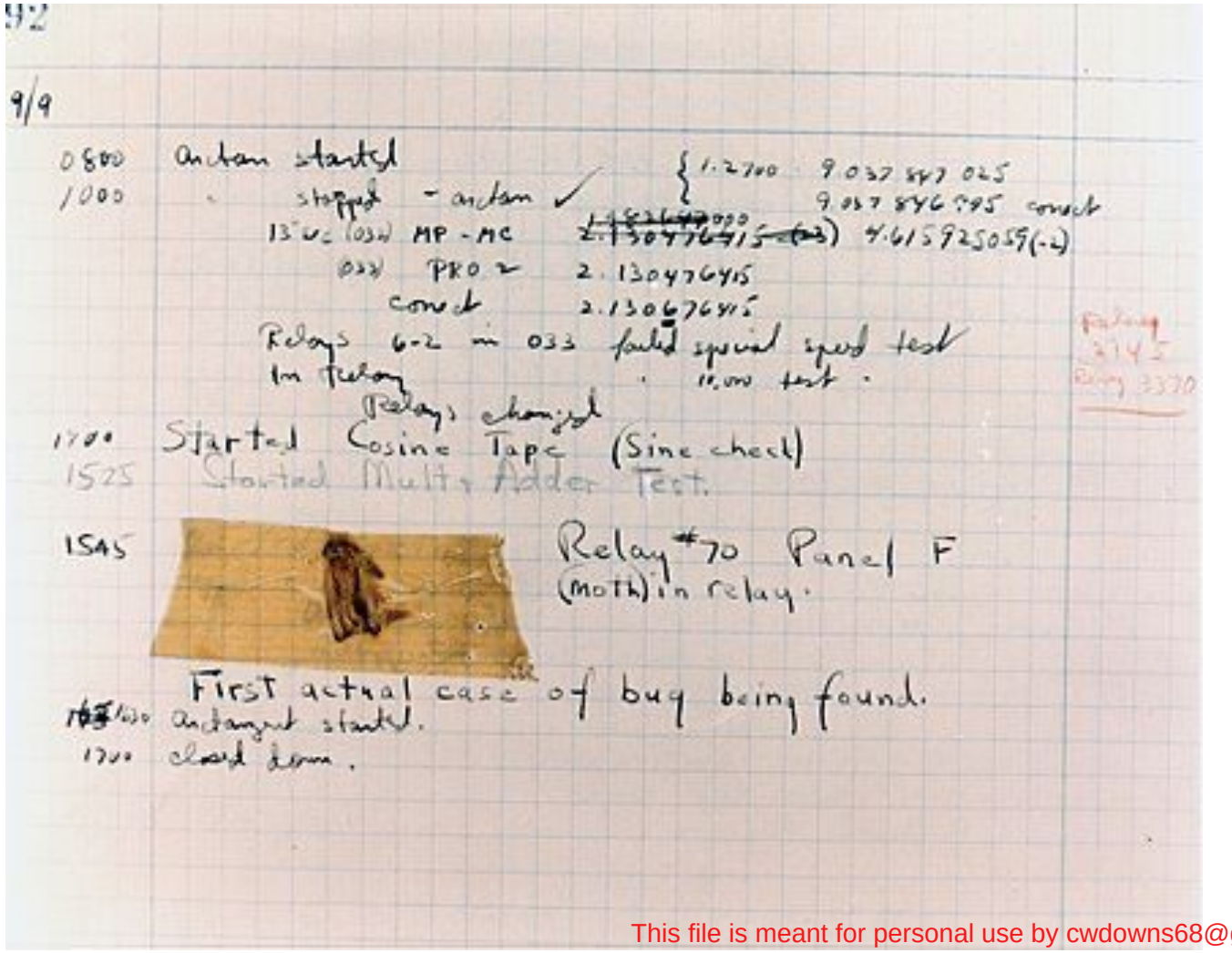
```
# Python program to print "Hello World!"
print("Hello, World!")
```

The " " indicate the boundary of a **string** of text to be printed. Note they are different than “ ”.

The **( )** are used to pass **arguments** to the function

print is a **function**.
It tells the computer to run a set of instructions

Debugging Code



- 1878 – Thomas Edison
- 1940s – Grace Hopper
- 1940s – Aircraft engines
- 1952 – ACM meetings
- 1960s - Commonplace

Data

Personal Data: Phone, email, calendars, etc

Location Data: Travel, traffic, GPS, weather

Social Media Data: Posts, friend/follower

Consumer Data: Purchase, views, psychographic

Data Types in Python

1, 42, -5, ...

Integer

`int`

1.0, 3.14, -0.1, ...

Float

`float`

3 + 5j

Complex

`complex`

'Hello'

Text

`str`

True/False

Boolean

`bool`

Variables

1. **Use descriptive names**: Instead of calling your variable x, use something like `total_sales` or `user_name`. It makes your code easier to understand.

2. **No spaces**: Variable names can't have spaces. Instead, use underscores, like `user_name`.

3. **Start with a letter**: Variable names must start with a letter or an underscore, not a number. For example, `1name` is not valid, but `_name1` is valid.

1. **Case-sensitive**: Python is case-sensitive, so `Age` and `age` are considered two different variables.

Variables - Tips

1. **Use meaningful names**: Variable names should reflect what they represent. For example, `total_price` is more meaningful than `tp`.

1. **Keep names concise**: While being descriptive is important, don't go overboard with long names. Keep it simple and to the point.

1. **Avoid reusing variable names**: It's a good practice to avoid using the same name for different purposes, as it may cause confusion or bugs in your code.

Variables - Summary

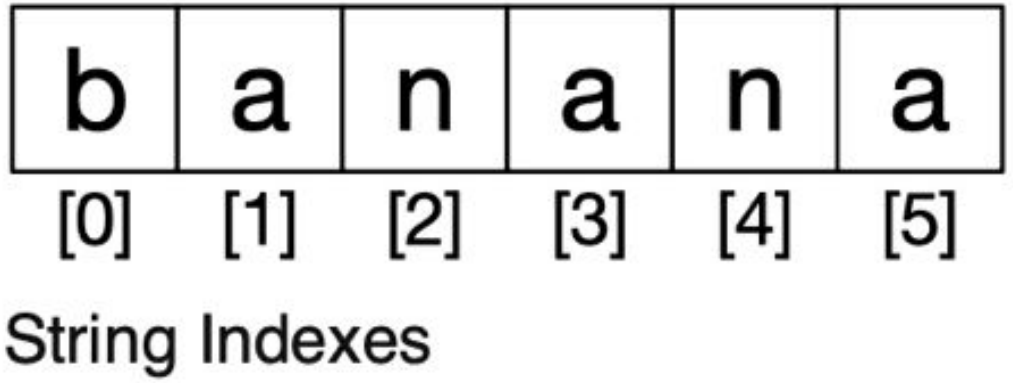
- Variables are names for storing values.
- You declare them by assigning a value with =.
- You can store different types of data in variables, like numbers, strings, and Booleans.
- Variable names should be descriptive and follow Python's naming rules.

Strings and Indexing

```
>>> fruit = 'banana'
>>> letter = fruit[1]
```

```
>>> print(letter)
a
```

```
>>> letter = fruit[0]
>>> print(letter)
b
```



String Indexes

https://www.youtube.com/watch?v=FN2RM-CHkuI#ddg-play

A function is a block of code designed to perform a specific task.

Why Use Functions

1. **Promote code reusability**: Instead of writing the same code over and over again, you can call a function whenever you need it.

1. **Improve readability**: By breaking your code into smaller, manageable parts, it becomes easier to read and understand.

1. **Simplify debugging**: If something goes wrong, it's easier to find and fix issues within a specific function rather than in the entire program.

1. **Help with organization**: Functions allow you to logically separate tasks in your program, making it more structured.

Functions - Summary

- A **function** is a reusable block of code that performs a specific task.
- Define a function using the `def` keyword, followed by name and parameters.
- Functions can have **parameters** to accept input and return values using the **return** statement.
- You can also provide **default values** for parameters or accept a variable number of arguments using `*args` and `**kwargs`.
- Functions help you make your code cleaner, more organized, and easier to maintain.

Key Differences Between Lists and Tuples

Feature	Lists	Tuples
Mutability	Mutable (can be modified)	Immutable (cannot be modified)
Syntax	Square brackets [ ]	Parentheses ( )
Methods available	More (e.g., append, pop)	Fewer (mainly count, index)
Performance	Slower (due to mutability)	Faster (due to immutability)
Use cases	When frequent updates are needed	When data should remain constant
Memory usage	Takes more memory	More memory efficient