Tutorial V.I: Programming with AI

Programming: Everyday Decision-Making Algorithms

LECTURER
Dr. Nils Roemer, External Lecturer

AFFILIATION
Planningio GmbH

Programming with AI

Using AI to generate code

- Coding by hand is not the only way to generate code
- Most likely, a lot of you have already used **ChatGPT**

How do

Large Language

Models work?

Photo by Taylor Vick on Unsplash

Large Language Models (LLMs)

- Think of them like advanced pattern recognition systems
- They have “read” **massive amounts of text**
- Books, websites, articles, code, and more
- Text is broken into **tokens**, parts of words or punctuation
- Based on patterns, they can **generate new text**

Training LLMs

- Imagine learning a language by reading millions of books
- Learns patterns in **how words and ideas connect** via tokens
- Interconnected nodes with **weights representing patterns**
- During training, these **weights are adjusted**
- Once trained, **applying** them takes much less resources

> Tip
>
> Using a trained model is called **inference**.

Pattern Recognition

- Not like a search engine!
- When asked, it looks for **relevant patterns** it learned
- Like having a **huge library** in its “memory” to draw from
- It can find **patterns between concepts** and your question

- Knows only limited text at once (**context window**)

> ⚠ Warning
>
> Managing **context windows** is crucial!

Probability based responses

- After each written token, it predicts "what should come next?"
- Like an advanced version of the **word prediction** on your phone
- Chooses the **most likely next token** based on training
- But can't actually "think" or "understand" like humans

Limitations

- **No true understanding** of cause and effect[1]
- Sometimes **makes mistakes or "hallucinates"**
- Mostly only knows what it **was trained on**[2] and can **reflect biases**
- No emotional understanding (but can simulate responses!)[3]

Impact on Jobs

- Question: What do you think about their impact on jobs?
- Question: What are the implications for us?
- Question: Can we use them to our advantage?

> ⚠ Warning
>
> Free models may use your prompts for training. Don't enter private or company data.

A Great Overview by 3Blue1Brown

- **Grant Sanderson provides an excellent explanation of LLMs**
- Great starting point to understand LLMs
- Check out his YouTube channel, 3Blue1Brown for more

AI Coding Partner

What is GitHub Copilot?

GitHub Copilot is an *AI pair programmer* inside your editor:

---

[1] https://www.anthropic.com/research/tracing-thoughts-language-model
[2] This can partially be improved by using context from the internet.
[3] User can get attached to talking to models

- Autocomplete for entire lines or blocks of code
- A chat that knows your code
- A tool to learn common coding patterns

> Tip
>
> Alternatives are Cursor, Zed or Claude Code. Copilot is free for students.

Used autocomplete before?

When you type on your phone, it suggests the next word.

GitHub Copilot does the same for code:

- You write a comment describing what you want
- Copilot suggests the code to do it
- You accept, modify, or reject the suggestion

It's autocomplete but much better than what you are used to

Why Use Copilot?

Benefits you while learning and working with Python:

- **Faster coding:** Less time typing boilerplate code
- **Learn patterns:** See how experienced programmers code
- **Understand syntax:** Get correct syntax without memorizing
- **Stay in flow:** Focus on logic, not syntax errors

> Tip
>
> Especially helpful when you know **WHAT** you want to do but forget **HOW** to do it.

When should you use Copilot?

Good uses of Copilot

- Understanding Python syntax you forgot
- Writing repetitive or boilerplate code
- Getting unstuck on simple problems
- Exploring different approaches

What shouldn't you do with Copilot?

Not so good uses of Copilot

- Replacing learning fundamentals
- Accepting code you don't understand
- Skipping practice exercises

- Copy-pasting without reading

Always understand what Copilot suggests before accepting!

But of course I know you will not do that ;)

Accept without reading?

Copilot might suggest code that:

- Uses concepts you haven't learned yet
- Contains subtle bugs or misses edge cases
- Doesn't match your requirements
- Introduces security risks (an issue in companies, less in this course)

My take: Just be careful, ok?

Your code, your responsibility:

1. **Read** the suggestion carefully
2. **Understand** what it does
3. **Test** it with examples
4. **Modify** if needed

> ⚠ Warning
>
> Don't accept code blindly, especially later if things are more complicated.

Getting Started with Copilot

Get Free Access

GitHub Student Developer Pack gives you free Copilot access!

1. Go to education.github.com/pack
2. Sign up with your university email
3. Verify your student status
4. Wait for approval (usually 1-2 days)
5. Log in to your account in VS Code

> ℹ Note
>
> You'll need a GitHub account. Create one at github.com if you don't have one.

Verifying Copilot is Working

1. Create a new Python file (`.py`) and type something

2. Wait 1-2 seconds

If working, you'll see:

- Gray "ghost text" suggesting code
- Press `Tab` to accept
- Press `Esc` to reject

> Tip
>
> Try to get Copilot running on your own until next session.

Summary

Key Takeaways

1. **GitHub Copilot is a tool:** You still need to learn fundamentals
2. **Understand before accepting:** Read every suggestion carefully
3. **Use it strategically:** Syntax help yes, thinking replacement no
4. **Get free access:** GitHub Student Developer Pack
5. **Practice:** The more you use it, the more helpful it becomes