TŘETÍ SEMINÁŘ / PROGRAMOVÁNÍ S AI

Jak programovat s AI?

Od rozhovoru ke spolupráci v editoru

Kdo určuje další krok a kdo rozhoduje o správnosti?

Mgr. Markéta Trnečková, Ph.D.

SPOLEČNÝ POKUS

Vyber číslo

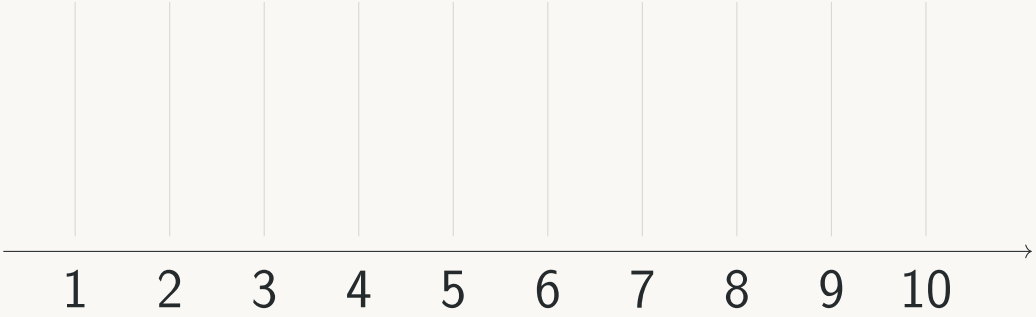
Vrať náhodné celé číslo od 1 do 10.

Otevřete nový chat a vyzkoušejte zadání.

Zapište své výsledky a použitý systém.

Jak vypadá rozdělení naší třídy?

Doplňme četnosti odpovědí.



Různá čísla $\neq$ prokázaný rovnoměrný výběr.

Jak získat náhodné číslo?

Jak se zeptat,
aby číslo bylo opravdu náhodné?

Vyzkoušejte svůj návrh. Co se změnilo?

Text, kód a spuštěný kód

1. **„Vyber číslo.“**
   Odpověď může vzniknout jako pokračování textu.

2. **„Napiš kód pro losování.“**
   Máme postup, ještě ne doklad losování.

3. **„Spusť generátor a ukaž výstup.“**
   Potřebujeme dostupný nástroj a skutečný běh.

Bez nástroje: vygenerovaný kód spustíme sami v notebooku.

Náhodnost a opakovatelnost

Simulace

```
import random
rng = random.Random(42)
rng.randint(1, 10)
```

Nepředvídatelný výběr

```
import secrets
secrets.randbelow(10) + 1
```

Pevný seed umožňuje zopakovat posloupnost.
Systémový zdroj náhodnosti slouží jinému cíli.

Jedno číslo ani pěkný histogram nejsou důkazem náhodnosti.

Generátor neopakujeme s týmž seedem před každým jednotlivým tahem.

Jak vygenerovat více náhodných čísel?

Je rozdíl mezi 100 nezávislými dotazy a jedním dotazem „Vyber 100 čísel.“?

Můžete vyzkoušet.

Jedna cesta, různé role AI

Chat → doplňování → kontext projektu

Dnes: zadávám, čtu, vybírám změnu a spouštím.

Příště: vibe coding a agentní programování.

Kdo vybírá další krok? Kdo mění soubory? Kdo spouští kód?

Co skrývá teplotní špička?

18, 19, 18, 20, 32, 21, 20, 19, 18, 20, 21, 20

Zobrazíme měření a postupně přidáme klouzavý průměr.

Je špička chyba senzoru, nebo skutečná událost?

Vyhlazení samo nerozhodne, zda je měření chybné.

1. Chat: popíšu, co chci

Napiš Python pro teploty [18,19,18,20,32,21,20,19,18,20,21,20]. Zobraz spojnicový graf s osami číslo měření a teplota v °C. Přidej klouzavý průměr.

Zadat → přečíst → přenést → spustit

Ukládejte si všechny verze kódu.

AI může být také učitel

„Vysvětli mi tento kód řádek po řádku.“

Které hodnoty vstupují do jednoho vybraného výstupu?

„Než odpovíš: polož mi otázku, zda tomu rozumím.“

Bez AI: ovlivní přidání nové hodnoty starší výsledky?

Srozumitelné vysvětlení ještě není důkaz vlastního porozumění.

Rozvíjíme program po jednom kroku

Porovnejte okna 3 a 5 ve stejném grafu.

Jaké okno váš program používá? Co dělá na začátku řady?

Nechte AI označit, které řádky přidala nebo změnila.

Porovnejte graf před změnou a po ní.

Větší okno špičku ztlumí, ale také rozmaže změnu v čase.

Co náš program opravdu zvládne?

Najděte vstup, který program prověří.

Nejprve bez AI navrhněte dva pokusy a očekávané chování.

AI pošleme: aktuální kód + vstup + celou chybovou zprávu + očekávané chování.

Požádejte nejprve o vysvětlení příčiny, potom o malou opravu.

Chyba neurčuje naše zadání

Převést?

Text „32“ může znamenat číslo z formuláře.

Odmítnout?

Funkce může vyžadovat pouze čísla.

Po vlastních pokusech: co má znamenat text „32“ ve vstupu?

Dohoda pro další práci: číselný text převedeme, jiný odmítneme.

AI navrhuje implementaci. Význam vstupu určujeme my.

Co v chatu dělá model a co člověk?

Jazykový model

Z dostupného kontextu generuje kód, vysvětlení nebo návrh opravy.

Prostředí Pythonu

Vykonává konkrétní instrukce nad konkrétními hodnotami.

Nová verze v chatu sama nepřepíše buňku ani běžící funkci.

Modelu předáváme právě tu verzi, kterou spouštíme.

Proč jazykový model umí psát kód?

Kód je také posloupnost tokenů.

Trénink může obsahovat kód, komentáře, dokumentaci i příklady použití.

Model se učí vztahy mezi popisem úlohy a její implementací.

Stejný základní cíl: $P(x_{t+1} \mid x_1, \ldots, x_t)$.

Historický Codex (2021) byl cíleně dotrénován na kódu.

Generování programu stále není jeho spuštění ani důkaz správnosti.

Jak doplňuje kód uprostřed souboru?

Před kurzorem + za kurzorem → chybějící část

```
def dvojnasobek(x):
    [ DOPLNIT ]
assert dvojnasobek(3) == 6
```

Fill-in-the-middle: při tréninku se prostřední část přesune za okolní text.

Model stále generuje zleva doprava, ale může dostat obě okolí.

Ne každý asistent používá stejný mechanismus. Bavarian et al., 2022.

2. Doplňování: návrh u kurzoru

```
def klouzavy(teploty, k):
  # Stejna delka vystupu; jen uplna zpetna okna.


  okno = teploty[i-k:i]
```

Ilustrační návrh. Přijmout, upravit, nebo odmítnout?

Ilustrační řez: obsahuje aktuální měření s indexem i?

Teď zpřesníme pravidla

Úplné zpětné okno zahrnuje aktuální hodnotu.

Konečná čísla; k je kladné celé číslo (ne bool). Jinak ValueError.

Stejná délka; neúplné okno: None; prázdný vstup: []. Vstup neměníme.

Ověřte také $k = 1$, k větší než délka a $k = 0$; k kontrolujte i pro [].

Zapište dohodu do komentáře funkce. Teprve potom přijímejte návrh asistenta.

Chat v editoru není autocomplete

Doplňování

Návrh pokračování přímo u kurzoru.

Přijímám nebo odmítám kus kódu.

Chat v editoru

Ptám se nad vybraným kódem.

Nechám vysvětlit či upravit určitou část.

Obojí může využívat kontext okolního kódu.

Ani jedno samo o sobě neznamená autonomní řešení úkolu.

3. Kontext: program začíná mít části

Náš notebook

1. data
2. funkce klouzavy
3. výpis výsledku
4. graf

Později projekt

data.py
vypocty.py
grafy.py
testy a README

Asistentovi zpřístupníme funkci i místo, kde ji používáme.

Vidět další části programu pomáhá zachovat souvislosti.

Jedna změna nad celým notebookem

Použij klouzavy ve výpisu i grafu pro okna 3 a 5. Zachovej pozice měření i None; žádné nuly místo chybějících hodnot. Původní řadu nemaž. Nejdřív popiš potřebné změny.

My posoudíme návrh, přijmeme úpravu a spustíme notebook.

Přístup ke kontextu není automatické svolení k dalším akcím.

Co vlastně asistent viděl?

Zadání + vybraný kód + kontext dodaný aplikací

Ne nutně všechny soubory, starší rozhovory nebo aktuální paměť Pythonu.

Ptejte se na konkrétní funkci a její použití.

Zkontrolujte navržené změny ve všech dotčených buňkách.

Větší kontext usnadňuje práci; stále potřebujeme jasný cíl.

Program dokončíme společně

[10, 20, 30, 40], okno 3:

[None, None, 20, 30]

k = 1: původní řada. Prázdný vstup: []. k = 0: chyba.

Restartujte prostředí a spusťte buňky od začátku.

V původní řadě je první průměr pro k = 3 roven 55/3.

Učím se programovat, nebo jen zadávat?

Bez AI vysvětlete jeden řádek svého programu.

Předpovězte výsledek pro nový malý vstup.

Pojmenujte rozhodnutí, které za vás neměl udělat model.

Co deleguji, když se učím? Co deleguji, když potřebuji výsledek?

Cognitive offloading: rozhoduje, kterou část práce právě procvičuji.

Příště předáme AI větší cíl

Agentní programování

AI volí kroky, upravuje a podle výsledku pokračuje.

Vibe coding

Iteruji hlavně podle chování prototypu; kód může zůstat mimo mou pozornost.

Jsou to různé osy: autonomie systému a způsob práce člověka.

Dnes nanejvýš ukázka: z našeho programu vznikne malý prototyp.

Studenti sledují, kdo zvolil další krok. Praktická práce patří příště.

Co dnes umíme s AI udělat?

Zadat → vysvětlit → upravit → opravit

Doplnit kód → propojit části → ověřit

Který režim byste zvolili pro malou funkci a proč?

Které rozhodnutí jste si ponechali vy?

Odevzdejte notebook a krátký záznam jedné přijaté nebo odmítnuté změny.

Zdroje a příprava prostředí

Koncepční návaznost: dodané slidy `ai_assisted_programming`.

Google Colab FAQ: notebooky a dostupnost AI funkcí.

GitHub Copilot: příklad doplňování a asistence v editoru.

Python: random a secrets.

Chen et al., 2021: učení na kódu; Bavarian et al., 2022: FIM.

Před hodinou ověřte dostupné režimy; přístup účtů zde nebyl testován.