

Legal Tech: I

Jan Černý, Tomáš Knap

crimczech

Charles University

Version 1.0, 2026

Motivace: why should we care?

Představte si, že máte v práci trestní spis NCOZ s tisíci stránkami

Dostali jste za úkol najít výsledky, ve kterých spoluobvinění mluví o vašem klientovi, Davidu Pokorném.

Co s tím?



Takhle může pomoci Python

Pár řádků kódu projde textové přepisy výslechů a vypíše soubory, které mohou zmiňovat klienta:

```
from pathlib import Path
for soubor in Path("vyslechy").glob("*.txt"):
    if "pokorn" in soubor.read_text(encoding="utf-8").lower():
        print(soubor.name)
```

- Hledá různé tvary příjmení: **Pokorný, Pokorného, Pokornému**
- Připraví **výběr dokumentů k přečtení**
- Význam nalezených pasáží posoudíte vy

Podobný skript se během kurzu naučíte napsat a upravit pro vlastní práci.

K čemu mi to bude?

- Méně rutiny
- Vlastní řešení
- Kontrola nad postupem a ownership
- Lepší práce s AI
- Nebudete meat proxy

Cíl: poznat, kterou část vlastní práce lze automatizovat, a umět s tím začít.

Babe, wake up. A new term just dropped!!

Meat proxy

A person who forwards AI-generated text, code, or other output without reading, understanding, or validating it. The person acts only as a relay between the AI system and the intended recipient.

Pojďte se to s námi naučit 😊

- **Základy Pythonu:** syntaxe, principy atd.
- **Práce s textem a soubory:** vyhledávání, úpravy textu, čtení a zápis běžných datových formátů
- **Tabulky a přehledy:** filtrování, propojování a vyhodnocování dat, jednoduché grafy
- **Regulární výrazy:** hledání údajů a opakujících se vzorů v textu
- **Práce s existujícími nástroji:** knihovny, objekty a získávání dat přes API

Vše na příkladech z právní praxe. Na kurz navazuje **Legal Tech II: LLMs pro právní praxi a vědu.**

Jak budeme v kurzu pracovat?

Web předmětu

Adresa: ggu.cz/lt

Na webu najdete slajdy a všechny podstatné informace i archiv

Poštovní sova

Adresa: owl.mff.cuni.cz

Enroll token: 711f16b18c68

Sova bude hlavní platforma pro odevzdávání úkolů a jejich hodnocení

Discord

Adresa: discord.gg/2dD7Ysngm

Ke komunikaci, pro dotazy a plánování

Co je potřeba ke splnění kurzu?

1. Zápočet

- Minimální počet bodů za úkoly
- Průběžné testy na papír

2. Písemná zkouška

Za úspěšné absolvování předmětu získáte **4 kredity**.

Co je potřeba ke splnění kurzu

1. Zápočet

Úkoly

- v poštovní sově se budou objevovat úkoly, na řešení každého bude čas nejméně do pondělka před další přednáškou. Úkol lze odevzdat i po stanoveném termínu, nejpozději však druhé pondělí následující po jeho zadání. Za takto opožděně odevzdaný úkol lze získat nejvýše 50 % maximálního počtu bodů. Za úkol odevzdaný později je 0
- pro zápočet musíte získat alespoň 90 bodů za úkoly, celkem lze získat 100
- za pomoc s předmětem můžete získat bonusové body (nahlašování chyb, bugů, překlepů...)

Testy

- testy se budou psát na papír
- musíte úspěšně projít všemi testy - hranice úspěšného složení testu je 70 %

2. Písemná zkouška

Známkování:

- 1 – 100–90 %
- 2 – 89–75 %
- 3 – 74–60 %
- 4 – 59–0 %

GenAI a domácí úkoly

Slepé spoléhání na AI při učení může omezit **osvojování vlastních dovedností**. Proto platí:

- **Zadání úkolů nevkládejte přímo do generativní AI**
- **Nepřebírejte ani nekopírujte kód vygenerovaný AI** do svého řešení
- **Vlastní, ručně napsaný kód s AI konzultovat můžete:** nechat si vysvětlit chybu, diskutovat svůj postup nebo si ověřit porozumění

Příklad vhodného dotazu

„Tento kód jsem napsal sám. Vysvětli mi, proč cyklus přeskočí poslední položku. Opravený kód mi napiš.“

K tématu: [Studie o AI a osvojování dovedností](#), [studie z klinické praxe](#)

Obecný úvod

Co je to počítač a jak mu říct, co má dělat

Co je to počítač?

Počítač

Stroj, který umí vykonávat jednoduché instrukce. Velmi rychle, neúnavně a naprosto doslovně.

- „Jednoduché“ opravdu znamená jednoduché: sečti dvě čísla, porovnej je, přesuň číslo jinam, pokračuj jinou instrukcí
- Síla není v „chytrosti“, ale v rychlosti: miliardy instrukcí za sekundu
- Z jednoduchých instrukcí skládáme **programy**, které dělají složité věci
- Počítač neví, co dělá: nepozná smlouvu od fotografie, jen přesouvá čísla
- Udělá **přesně** to, co mu řekneme, ne to, co jsme mysleli



- **1642** – Pascalina: mechanická sčítačka Blaise Pascala
- **1837** – Analytický stroj Charlese Babbage: první návrh univerzálního počítače
- **1843** – Ada Lovelace pro něj píše první program v historii; stroj nikdy nebyl dostavěn
- **1890** – Hollerithovy děrné štítky sčítají lid v USA; z jeho firmy vyroste IBM
- **1936** – Alan Turing popisuje matematický model počítače: Turingův stroj
- **1941** – Z3 Konrada Zuseho: první fungující programovatelný počítač
- **1944–1945** – Colossus a ENIAC: první elektronické počítače, zrozené z války
- **1947** tranzistor, **1971** mikroprocesor, **1981** IBM PC, **1991** web (a Python), **2007** iPhone

ENIAC (1945): první univerzální elektronický počítač



- 18 000 elektronek, 27 tun, 170 m² podlahy
- 5 000 sčítání za sekundu, asi tisíckrát rychleji než dosavadní stroje
- „Programovalo“ se přepojováním kabelů a nastavováním přepínačů; program nebyl uložen v paměti
- Určen pro výpočet balistických tabulek pro dělostřelectvo americké armády
- Dnešní telefon je zhruba stomilionkrát rychlejší

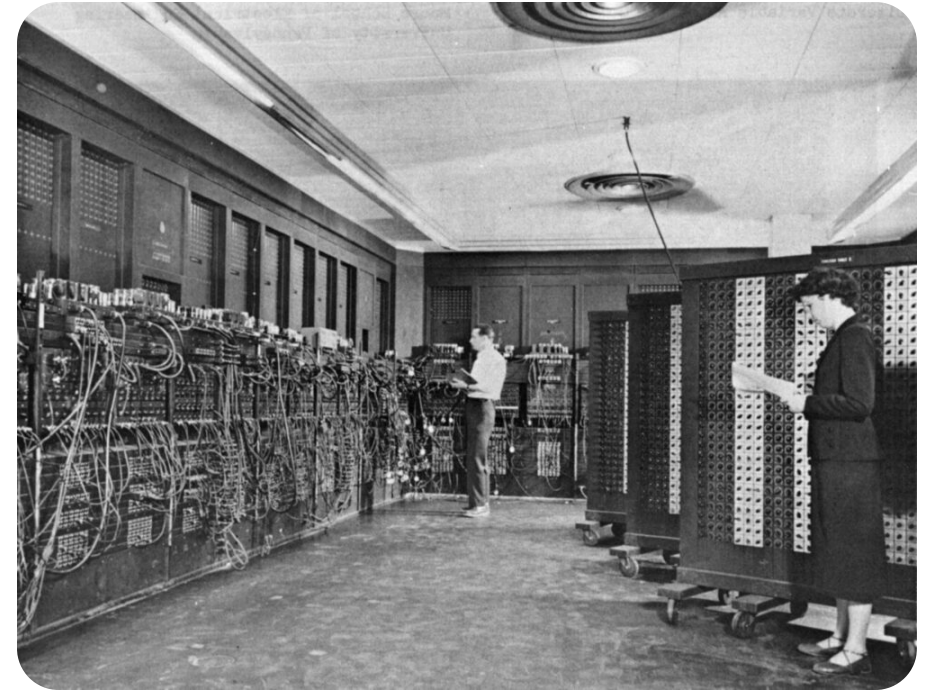


Foto: U.S. Army, volné dílo ([Wikimedia Commons](#))

Turingův stroj (1936)

- Nekonečná **páska** rozdělená na políčka se symboly
- **Hlava**, která políčko přečte, případně přepíše a posune se o jedno
- Konečná tabulka **pravidel**: „ve stavu q_1 vidím 1 $\rightarrow$ zapiš 0, posuň se doprava, přejdi do q_2 “
- Nic víc. A přesto: cokoli lze vůbec spočítat, spočítá i tento stroj
- Každý dnešní počítač je v principu Turingův stroj, jen rychlejší a s konečnou páskou





- **Alan Turing** (1912–1954): britský matematik, autor Turingova stroje a zakladatel teoretické informatiky
- **Enigma**: německý šifrovací stroj; nastavení se mění každý den a možností je asi $1,6 \cdot 10^{20}$
- 1932: polští matematici Rejewski, Różycki a Zygański šifru poprvé prolomí
- 1940: Turingova „Bomba“ v Bletchley Parku zkouší nastavení strojově; 1944 Colossus luští i těžší šifru Lorenz
- 1950: Turingův test – „Mohou stroje myslet?“



Foto: Alessandro Nassiri, Museo scienza e tecnologia Milano, [CC BY-SA 4.0](#) ([Wikimedia Commons](#))

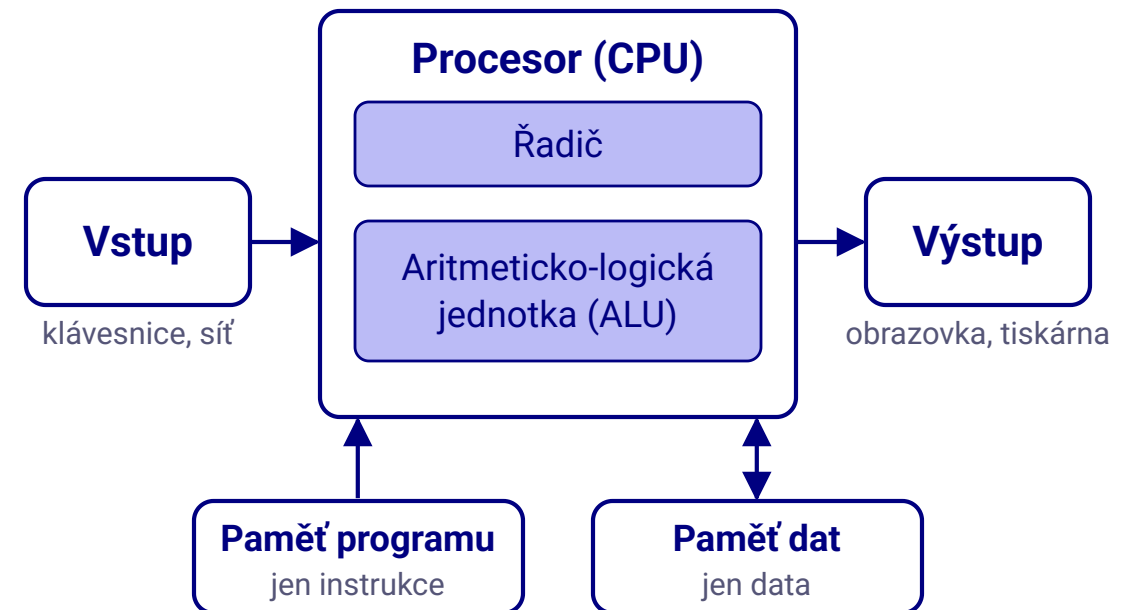
Z čeho se počítač skládá

Součást	Co dělá	V kanceláři
Procesor (CPU)	vykonává instrukce: počítá, porovnává, rozhoduje	úředník
Operační paměť (RAM)	rychlá, ale malá; po vypnutí se vymaže	pracovní stůl
Úložiště (HDD, SSD)	velké, pomalejší; data přežijí vypnutí	spisovna
Vstup a výstup (I/O)	klávesnice, myš, obrazovka, síť, tiskárna	podatelna

- Úředník pracuje jen s tím, co má na stole; spis si musí nejdřív nechat přinést ze spisovny
- Vše ostatní (grafická karta, síťová karta, USB) je jen obměnou těchto čtyř rolí

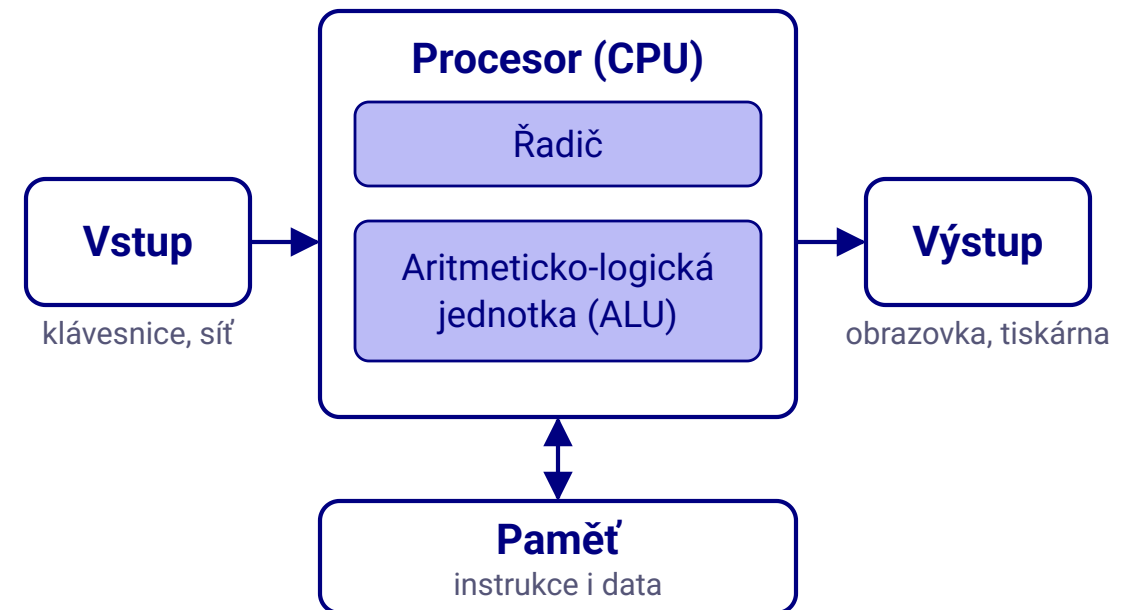
Harvardská architektura (1944)

- **Program a data mají každý vlastní paměť a vlastní cestu do procesoru**
- Procesor čte instrukci i data najednou;
- program nejde za běhu přepsat
- Název podle Harvard Mark I (1944): program na děrné pásce, data v počítadlech
- Cena: dvě paměti a menší pružnost
- Dnes: mikrokontroléry (auto, pračka, platební karta); moderní procesory obojí kombinují



Von Neumannova architektura (1945)

- **Program i data leží ve stejné paměti.**
Program je tedy jen další data: dá se nahrát, kopírovat, měnit
- Procesor = **řadič** (rozhoduje, co dál) + **aritmeticko-logická jednotka** (počítá)
- Stále dokola: **načti** instrukci z paměti → **dekóduj** → **proved'** → další
- Podle tohoto schématu funguje skoro každý počítač, od telefonu po superpočítač



Všechno jsou čísla: text

Počítač nezná písmena. Domluvíme se proto na **tabulce**: každý znak dostane své číslo.

Znak	A	B	C	...	a	b	...	mezera	§	Č	ř
Číslo	65	66	67	...	97	98	...	32	167	268	345

- Slovo **Ahoj** je pro počítač **65 104 111 106**; celá smlouva nebo zákon je jen delší řada čísel
- Dnešní tabulka se jmenuje **Unicode** a pokrývá všechny abecedy světa i emoji (starší a menší je ASCII)

Zkuste si

Co říká počítač? **80 114 225 118 111** – nápověda: á = 225, o = 111, P = 80, r = 114, v = 118

Všechno jsou čísla: úplně všechno

- **Barva:** tři čísla 0–255 pro červenou, zelenou a modrou; modrá v záhlaví je (0, 0, 128)
- **Obrázek:** mřížka bodů (pixelů), každý bod je barva; fotka z telefonu ≈ 36 milionů čísel
- **Zvuk:** 44 100krát za sekundu změříme výchylku membrány mikrofonu → dlouhá řada čísel
- **Datum a čas:** počet sekund od 1. 1. 1970; právě teď asi 1 790 000 000
- **Rozhodnutí soudu:** text, tedy čísla; jeho metadata (soud, datum, spisová značka) také

Tatáž čísla ale mohou znamenat cokoli: 65 je „A“, věk, § 65 i odstín červené. **Význam dává až kontext** – a ten musí počítači někdo říct.

Proč zrovna nuly a jedničky?

- Elektronika spolehlivě rozliší dva stavy: napětí **je** / **není**
- Jedna hodnota ano/ne je **bit**; osm bitů je **bajt** a pojme 256 možností (0 až 255)
- Libovolné číslo zapíšeme jen nulami a jedničkami, ve **dvojkové soustavě**:

Desítkově	0	1	2	3	4	5	10	65	255
Dvojkově	0	1	10	11	100	101	1010	1000001	11111111

- Pro člověka nepohodlné, pro stroj ideální; převod obstará počítač sám
- Objemy dat: stránka textu $\approx$ 2 kB, kniha $\approx$ 1 MB, fotografie $\approx$ 3 MB, film $\approx$ 2 GB (1 GB $\approx$ miliarda bajtů)

Počítač rozumí jen číslům

- **Data jsou čísla:** text, obrázky, zvuk, cokoli
- **Instrukce jsou čísla:** i program leží v paměti jako řada čísel, tzv. **strojový kód** (skutečné instrukce procesoru x86):

```
184 5 0 0 0    →  ulož číslo 5 do registru A  
131 192 1      →  přičti k registru A jedničku
```

- Procesor nic jiného neumí. Lidé ale v číslech nemyslí: takový zápis je nečitelný, pracný a plný chyb
- Potřebujeme prostředníka: zápis, který **čtou lidé** a který **umíme strojově přeložit na čísla**
- Tomu se říká **programovací jazyk**

Co je to programovací jazyk

Programovací jazyk

Umělý formální jazyk pro zápis instrukcí počítači. Přesná pravidla umožňují strojový překlad na čísla.

- Účel: **lidé píšou, počítač vykonává**. Kód se čte častěji, než píše, proto má být čitelný
- Například v Pythonu i bez znalosti jazyka odhadnete, co dělá:

```
vek = 17
if vek < 18:
    print("Nezletilý")
```

- Na rozdíl od přirozeného jazyka (i toho právního) nepřipouští výklad: zápis znamená právě jednu věc

Paradigma

- způsob, jak o programu přemýšlíme. Stejný úkol lze zapsat několika způsoby:
 - **imperativní**: posloupnost příkazů „udělej A, pak B, když C, tak D“ (Python, C, Java)
 - **funkcionální**: skládání funkcí jako ve vzorcích v Excelu (Haskell, Lisp)
 - **logické**: zapíšeme fakta a pravidla, závěry odvodí stroj (Prolog)
 - **objektové**: data a operace nad nimi drží pohromadě (Java, C#, Python), uvidíme později
 - a další...

Syntax

- **Syntax** jsou pravidla zápisu: které znaky, v jakém pořadí, co znamená odsazení nebo středník
- Syntaktická chyba: program se vůbec nespustí, počítač ohlásí `SyntaxError` a řádek, kde je
- Syntax je forma, **sémantika** význam: formálně bezvadné podání může být nesmyslné, u programu stejně

Stejný program ve čtyřech jazycích: vypiš „Dobrý den“

```
print("Dobrý den") # Python
```

```
console.log("Dobrý den"); // JavaScript
```

```
#include <stdio.h> /* C */  
int main(void) { printf("Dobrý den\n"); }
```

```
mov rax, 1          ; assembly (x86-64, Linux): chci volat „write“, zápis na výstup  
mov rdi, 1          ; kam: na obrazovku  
mov rsi, zprava      ; odkud: adresa, na které je uložen text „Dobrý den“  
mov rdx, 10         ; kolik: 10 bajtů („ý“ zabírá dva)  
syscall             ; proved'
```

Jak se z textu stane běžící program

Kompilace

Překladač (kompilátor) přeloží **celý program předem** do strojového kódu; ten pak běží přímo a rychle. (C, Rust, Go)

Interpretace

Interpret čte program **za běhu**, příkaz po příkazu, a rovnou ho provádí. (Python, JavaScript)

- Analogie: přeložená kniha vs. tlumočník u soudu
- Kompilované programy běží rychleji; interpretované se rychleji píšou a zkoušejí
- Python je interpretovaný: napíšeme řádek, spustíme, hned vidíme výsledek

Úroveň jazyka: jak daleko je od procesoru

Úroveň	Jazyky	Co to znamená
Vysoká	Python, MATLAB, JavaScript	blízko lidskému uvažování; jeden řádek udělá hodně; o paměť a detaily se stará jazyk
Střední	Rust, C#, Java	kompromis mezi výkonem a pohodlím
Nízká	C, Assembly	blízko procesoru; plná kontrola, ale i plná zodpovědnost; hodně řádků

- Čím vyšší úroveň, tím „blíže člověku“ a tím méně řešíme, **jak** to procesor provede
- Cena: vyšší úroveň bývá pomalejší. Pro práci s textem a daty to (skoro) nikdy nevadí
- Assembly je jen čitelnější zápis strojového kódu: `mov eax, 5` místo `184 5 0 0 0`

Úroveň jazyka v praxi: sečti čísla v seznamu

```
soucet = sum(seznam_cisel)
```

```
int soucet = 0;  
for (int i = 0; i < n; i++) soucet += seznam_cisel[i];
```

```
    xor    eax, eax          ; součet = 0  
smycka:  
    add    eax, [rsi]        ; přičti číslo, na které ukazuje rsi  
    add    rsi, 4            ; posuň se na další číslo  
    loop   smycka            ; sniž počítadlo; není-li nula, opakuj
```

Python: jeden řádek. C: dva. Assembly: pět – a to je ten nejjednodušší případ.

Vlastnosti programovacích jazyků: Práce s pamětí

- **Staticky typované jazyky** (C, Java, Rust): typ proměnné je dán předem a kontroluje se ještě před spuštěním
- **Dynamicky typované jazyky** (Python, JavaScript): typ se zjistí až za běhu a proměnná ho může i změnit

```
x = 5          # číslo  
x = "pět"      # teď text, Python nic nenamítá
```

```
int x = 5;  
x = "pět";    // chyba už při překladu: text do čísla nejde
```

- **Paměť:** v C si ji programátor půjčuje a vrací sám (a často zapomene); Python, Java i JavaScript ji uklízejí samy („garbage collector“); Rust ji hlídá překladačem
- Python: dynamické typy a automatická paměť. Chyba v typu se ale ukáže až za běhu

Vlastnosti programovacích jazyků: Účel a filozofie

- **Univerzální** jazyky zvládnou cokoli (Python, Java, C)
- **Specializované** jsou stavěné na jeden druh úloh: SQL pro dotazy do databází, R pro statistiku, MATLAB pro inženýrské výpočty
- Každý jazyk má **filozofii**, která se propisuje do jeho podoby, a typické **použití**:

Jazyk	Motto	Typické použití
Python	„Čitelnost se počítá“	data, věda, automatizace, umělá inteligence
JavaScript	„Běží v každém prohlížeči“	web a webové aplikace
C, Rust	„Důvěřuj programátorovi“	operační systémy, zařízení, vše, kde jde o výkon
Java, C#	„Napiš jednou, spust' všude“	firemní a bankovní systémy
SQL	„Řekni, co chceš, ne jak“	dotazy do databází: rejstříky, evidence rozhodnutí

- **Čitelný:** skoro jako angličtina, málo závorek
- **Interpretovaný:** napíšeme, spustíme, vidíme
- **Univerzální:** text, data, web, automatizace, AI
- **Obrovský ekosystém:** knihovny i pro .docx a .pdf
- **Nejpopulárnější jazyk,** zdarma, otevřený kód
- Guido van Rossum, 1991; název podle Monty Pythona



Logo: Python Software Foundation

```
smlouvy = ["Smlouva o dílo", "Kupní smlouva",  
           "Smlouva o dílo", "Nájemní smlouva"]  
pocet = 0  
for smlouva in smlouvy:  
    if smlouva == "Smlouva o dílo":  
        pocet = pocet + 1  
print("Smluv o dílo:", pocet)
```

- Čte se skoro jako věta: *pro každou smlouvu v seznamu: pokud je to smlouva o dílo, přičti jedničku*
- Odsazení není kosmetika: říká Pythonu, které řádky patří k sobě
- Podobný program brzy napíšete sami – jen místo čtyř názvů bude číst stovky souborů

- **Průvodce labyrintem algoritmů**: Martin Mareš, Tomáš Valla. Srozumitelný úvod do algoritmů a datových struktur, zdarma online. *Velmi doporučeno*
- **KSP Encyklopedie**: programátorské „kuchařky“
Korespondenčního semináře z programování MFF UK
- **Ponořme se do Python(u) 3**: Mark Pilgrim. Český překlad učebnice *Dive Into Python 3*, zdarma z Edice CZ.NIC



Obálka: pruvodce.ucw.cz

Jaké nástroje budeme používat

VSCodium

Editor kódu: otevřená verze VS Code bez telemetrie Microsoftu.

uv

Spravuje Python i knihovny: instalace, projekty, spouštění. Jeden nástroj místo pěti.

```
uv init muj-projekt      # založ nový projekt
uv add python-docx       # přidej knihovnu pro práci s dokumenty Wordu
uv run main.py           # spusť program
```

Instalaci obou nástrojů si ukážeme společně.

Bash a příkazová řádka

Ovládání počítače psaním příkazů místo klikání: rychlejší, opakovatelné, skládá se do skriptů.

Git

Správa verzí: eviduje každou změnu, vrátí se zpět, spojí práci více lidí. „Sledování změn“ pro celé projekty.

- Kde začít: [Pro Git](#) (kniha zdarma, i česky) a [The Missing Semester](#) (MIT: příkazová řádka, Git, editory)
- Obojí se hodí i právníkovi: dohledatelná historie dokumentů a opakovatelné postupy

1. Nainstalujte si [VSCodium](#) a [Python 3](#).
2. Ve VSCodiu otevřete záložku **Extensions (Rozšíření)** a nainstalujte rozšíření **Python** od `ms-python`.
3. Přes **File** → **Open Folder** otevřete složku pro své úkoly.
4. Vytvořte v ní soubor `uko1.py`, vložte do něj následující kód a soubor uložte:

```
import base64
print(base64.b64decode("S3JhanRh").decode())
```

5. Otevřete paletu příkazů pomocí **Ctrl+Shift+P** (na Macu **⌘ ↑ P**), zvolte **Python: Select Interpreter** a vyberte nainstalovaný Python 3.
6. Spusťte soubor tlačítkem ► **Run Python File** vpravo nahoře. Výsledek se zobrazí dole v terminálu.

Děkujeme za pozornost