Midterm test (vzorové řešení)

Legal Tech I
3. 11. 2026

Upozornění: Pište čitelně a svoje odpovědi dostatečně odůvodněte.

Otázka 1

Jaký je rozdíl mezi staticky a dynamicky typovanými jazyky?

Je python *staticky* nebo *dynamicky* typovaný jazyk?

> **Vzorová odpověď**
>
> - **Staticky typovaný** jazyk kontroluje typy už při překladu; proměnná má pevný typ a jiný do ní uložit nejde.
> - **Dynamicky typovaný** jazyk zjišťuje typy až za běhu; proměnná může typ měnit.
> - Python je **dynamicky** typovaný: `a = 1` a hned potom `a = "text"` je v pořádku.

Otázka 2

Jaký je rozdíl mezi kompilovanými a interpretovanými jazyky?

Je Python kompilovaný nebo interpretovaný jazyk?

> **Vzorová odpověď**
>
> - **Kompilovaný** jazyk se nejprve celý přeloží kompilátorem do strojového kódu a spouští se až výsledný program (C, C++).
> - **Interpretovaný** jazyk čte a provádí interpret přímo ze zdrojového kódu, překlad předem není potřeba.
> - Python je **interpretovaný**: `python program.py` spustí interpret, který kód rovnou vykonává.

Otázka 3

Jaký je rozdíl mezi vysokoúrovňovými a nízkoúrovňovými jazyky?

Je Python spíš vysokoúrovňový nebo nízkoúrovňový?

> **Vzorová odpověď**
>
> - **Vysokoúrovňový** jazyk je blízký lidskému uvažování: pracuje s proměnnými, seznamy a funkcemi a skrývá detaily hardwaru (paměť, registry).
> - **Nízkoúrovňový** jazyk je blízký stroji: programátor řídí paměť a instrukce procesoru (assembler, částečně C).
> - Python je **vysokoúrovňový**: jeden řádek `print(sum(cisla))` nahradí desítky strojových instrukcí.

Otázka 4

Jaký je rozdíl mezi syntaxí a sémantikou programovacího jazyka?

Co se stane, když má program chybu v syntaxi, a co když má chybu v sémantice?

> **Vzorová odpověď**
>
> - **Syntaxe** jsou pravidla zápisu (závorky, dvojtečky, odsazení). **Sémantika** je význam zapsaného, tedy co program dělá.
> - **Chyba v syntaxi**: program se vůbec nespustí, interpret ohlásí `SyntaxError` ještě před prvním řádkem.
> - **Chyba v sémantice**: program se spustí, ale dělá něco jiného, než má, nebo spadne až za běhu (třeba `TypeError`).

Otázka 5

Python je dynamicky typovaný jazyk. K čemu tedy slouží zápis `vek: int = 17`?

Kontroluje Python, že se do proměnné `vek` opravdu uloží celé číslo?

> **Vzorová odpověď**
>
> - Je to **typová nápověda** (anotace). Slouží čtenáři kódu, editoru (našeptávání) a kontrolním nástrojům jako mypy.
> - Python ji za běhu **nekontroluje**: `vek: int = "sedmnáct"` projde bez chyby a ve `vek` je string.

Otázka 6

```
a = 10
a /= 2
print(a)
print(type(a))
```

Co se vypíše jako první `print`?

Jaký datový typ vypíše druhý `print`?

(Nemusíte psát celou zprávu, ale pouze datový typ. Například místo `:<class 'int'>` stačí `int`.)

> **Vzorová odpověď**
>
> - První `print` vypíše `5.0`.
> - Druhý `print` vypíše `float`.
> - Dělení `/` vrací vždy `float`, i když vyjde celé číslo. Celočíselné dělení je `//`.

Otázka 7

```
a = 7
b = 2
c = a / b
d = a // b
print(c, d)
print(type(c), type(d))
```

Co vypíše první `print`?

Jaké datové typy vypíše druhý `print`?

(Nemusíte psát celou zprávu, ale pouze datový typ. Například místo `:<class 'int'>` stačí `int`.)

> **Vzorová odpověď**
>
> - První `print` vypíše `3.5 3`.
> - Druhý `print` vypíše `float` a `int`.
> - `/` dělí vždy na `float`, `//` dělí celočíselně a ze dvou `int` vrací `int`.

Otázka 8

```
a = "17"
b = 1
c = int(a) + b
d = a + str(b)
print(c, d)
print(type(c), type(d))
```

Co vypíše první `print`?

Jaké datové typy vypíše druhý `print`?

(Nemusíte psát celou zprávu, ale pouze datový typ. Například místo `:<class 'int'>` stačí `int`.)

> **Vzorová odpověď**
>
> - První `print` vypíše `18 171`.
> - Druhý `print` vypíše `int` a `str`.
> - `int("17") + 1` je sčítání čísel, `"17" + "1"` spojení stringů.

Otázka 9

```
a = 7
b = 2
c = a % b
d = a ** b
print(c, d)
print(type(c), type(d))
```

Co vypíše první `print`?

Jaké datové typy vypíše druhý `print`? (Nemusíte psát celou zprávu, ale pouze datový typ. Například místo :`<class 'int'>` stačí `int`.)

> **Vzorová odpověď**
>
> - První `print` vypíše `1 49`.
> - Druhý `print` vypíše `int` a `int`.
> - % je zbytek po dělení (7 : 2 = 3, zbytek 1), `**` mocnina ($7^2 = 49$). Oba z `int` vrací `int`.

Otázka 10

```
podepsano = True
zaplaceno = False

if podepsano and zaplaceno:
    print("Hotovo")
elif podepsano or zaplaceno:
    print("Rozpracováno")
else:
    print("Nezačato")
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> Rozpracováno
> ```
>
> - `True and False` je `False`, `True or False` je `True`.

Otázka 11

```
a = 4
b = 7

if a < b:
    if a % 2 == 0:
        print("sudé")
    else:
        print("liché")
else:
    print("větší")
print(a + b)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> sudé
> 11
> ```
>
> - `4 < 7` platí, uvnitř `4 % 2 == 0` platí. Poslední `print` je mimo podmínky.

Otázka 12

```
a = 2
b = 2

if a == b:
    print("rovná se")
    a += 1
if a == b:
    print("pořád se rovná")
else:
    print(a, b)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> rovná se
> 3 2
> ```
>
> - První `if` platí a zvýší a na 3. Druhý `if` už neplatí, `else` vypíše `3 2`.

Otázka 13

Uživatel do tohoto programu zadal `3`.

```
a = input()
print(3 * a)
```

Co tento program vypíše?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 333
> ```
>
> - `input()` vrací vždy string, takže `3 * "3"` string třikrát zopakuje.

Otázka 14

Uživatel do tohoto programu zadal `7`.

```
a = input()
if a == 7:
    print("sedm")
else:
    print("něco jiného")
```

Co tento program vypíše a proč?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> něco jiného
> ```
>
> - `input()` vrátil string `"7"` a ten se číslu `7` nerovná (`"7" == 7` je `False`). `True` by bylo `int(a) == 7` nebo `a == "7"`.

Otázka 15

```
a = 0
for i in range(3):
    for j in range(3):
        for k in range(3):
            a += 1
print(a)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 27
> ```
>
> - Tři vnořené cykly po třech průchodech: `3 * 3 * 3 = 27` přičtení.

Otázka 16

```
for i in range(2, 7):
    print(i)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 2
> 3
> 4
> 5
> 6
> ```
>
> - `range(2, 7)` začíná na 2 a končí **před** 7.

Otázka 17

```
smlouvy = ["Kupní", "Nájemní", "O dílo"]
for smlouva in smlouvy:
    print(smlouva, len(smlouva))
print("Hotovo")
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> Kupní 5
> Nájemní 7
> O dílo 6
> Hotovo
> ```
>
> - `len()` počítá znaky včetně mezery (`"O dílo"` má 6). `Hotovo` je mimo cyklus.

Otázka 18

```
a = 0
for i in range(5):
    if i % 2 == 0:
        continue
    a += i
print(a)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 4
> ```
>
> - Sudá `i` (0, 2, 4) přeskočí `continue`, sečtou se jen lichá: `1 + 3 = 4`.

Otázka 19

```
a = 0
while a < 6:
    a += 1
    if a % 2 == 0:
        continue
    print(a)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 1
> 3
> 5
> ```
>
> - a se zvýší před testem; sudé hodnoty (2, 4, 6) přeskočí `continue` a nevypíší se.

Otázka 20

```
a = 5
while a > 0:
    print(a)
print("Konec")
```

Co udělá tento program? Vypíše se někdy `Konec`?

> **Vzorová odpověď**
>
> Program běží **nekonečně** a stále vypisuje 5:
>
> ```
> 5
> 5
> 5
> ...
> ```
>
> - Uvnitř cyklu se a nemění, podmínka `a > 0` platí pořád. `Konec` se nikdy nevypíše; program je nutné ukončit (Ctrl+C).

Otázka 21

```
a = 3
while a > 10:
    print(a)
    a += 1
print("Konec")
```

Co vypíše tento program? Kolikrát proběhne tělo cyklu?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> Konec
> ```
>
> - `3 > 10` neplatí hned na začátku, tělo cyklu neproběhne **ani jednou**.

Otázka 22

```
smlouvy = ["Kupní", "Nájemní", "O dílo", "Darovací"]
print(smlouvy[0])
print(smlouvy[-1])
print(smlouvy[1:3])
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> Kupní
> Darovací
> ['Nájemní', 'O dílo']
> ```
>
> - Index `0` je první prvek, `-1` poslední. Výřez `[1:3]` bere indexy 1 a 2, index 3 už ne.

Otázka 23

```
smlouvy = ["Kupní", "Nájemní"]
smlouvy.insert(0, "Darovací")
smlouvy.remove("Kupní")
posledni = smlouvy.pop()
print(smlouvy)
print(posledni)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> ['Darovací']
> Nájemní
> ```
>
> - `insert(0, …)` vloží na začátek, `remove()` odstraní podle hodnoty, `pop()` odebere **poslední** prvek a vrátí ho.

Otázka 24

```
cisla = [3, 1, 4, 1, 5]
print(sum(cisla))
print(max(cisla))
print(cisla.count(1))
print(cisla.index(4))
print(7 in cisla)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 14
> 5
> 2
> 2
> False
> ```
>
> - `sum()` sečte prvky, `max()` vrátí největší, `count(1)` spočítá výskyty jedničky, `index(4)` vrátí první index hodnoty 4, `7 in cisla` je `False`.

Otázka 25

```
mnozina = {1, 2, 3}
print(mnozina[0])
```

Co tento program vypíše? Pokud spadne, napište jakou chybou a proč.

> **Vzorová odpověď**
>
> Program spadne na druhém řádku:
>
> ```
> TypeError: 'set' object is not subscriptable
> ```
>
> - Množina **nemá pořadí**, nelze se ptát na „první prvek“ přes `[0]`. Je nutné převést na seznam: `list(mnozina)[0]`.

Otázka 26

```
mnozina = {140, 173, 206}
mnozina.discard(999)
mnozina.remove(140)
print(sorted(mnozina))
mnozina.remove(999)
print("Konec")
```

Co vypíše tento program? Pokud spadne, napište na kterém řádku a jakou chybou.

> **Vzorová odpověď**
>
> Vypíše `[173, 206]` a pak spadne na pátém řádku `mnozina.remove(999)`:
>
> ```
> KeyError: 999
> ```
>
> - `discard()` chybějící prvek tiše ignoruje, `remove()` na něm spadne. `Konec` se nevypíše.

Otázka 27

```
a = {1, 2}
b = {1, 2, 3, 4}
print(a.issubset(b))
print(b.issubset(a))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> True
> False
> ```
>
> - `{1, 2}` je podmnožinou `{1, 2, 3, 4}`, obráceně to neplatí.

Otázka 28

```
tresty = {"krádež": 205, "podvod": 209}
tresty["loupež"] = 173
tresty["krádež"] = 206
print(tresty["krádež"])
print(len(tresty))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> 206
> 3
> ```
>
> - `tresty["loupež"] = 173` přidá nový klíč, `tresty["krádež"] = 206` hodnotu u existujícího klíče přepíše.

Otázka 29

```
tresty = {"krádež": 205, "podvod": 209}
print(tresty["podvod"])
print(tresty["loupež"])
```

Co tento program vypíše? Pokud spadne, napište na kterém řádku, jakou chybou a proč.

> **Vzorová odpověď**
>
> Vypíše `209` a pak spadne na třetím řádku:
>
> ```
> KeyError: 'loupež'
> ```
>
> - Klíč `loupež` ve slovníku není. Pád zabrání `tresty.get("loupež")` nebo test `"loupež" in tresty`.

Otázka 30

```
tresty = {"krádež": 205, "podvod": 209}
for cin in tresty:
    print(cin, tresty[cin])
print(list(tresty))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> krádež 205
> podvod 209
> ['krádež', 'podvod']
> ```
>
> - Cyklus `for` i `list()` jdou po klíčích, a to v pořadí, v jakém byly vloženy.

Otázka 31

```
slova = ["smlouva", "žaloba", "smlouva", "smlouva"]
pocty = {}
for slovo in slova:
    pocty[slovo] = pocty.get(slovo, 0) + 1
print(pocty)
print(pocty["smlouva"])
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> {'smlouva': 3, 'žaloba': 1}
> 3
> ```
>
> - `pocty.get(slovo, 0)` dá u nového slova 0, u známého dosavadní počet; `+ 1` ho zvýší. Klasický vzor počítání výskytů.

Otázka 32

```
slovo = "Právo"
print(slovo[0])
print(slovo[-1])
print(slovo[1:3])
print(len(slovo))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> P
> o
> rá
> 5
> ```
>
> - String se indexuje jako seznam: `[0]` první znak, `[-1]` poslední, `[1:3]` znaky na indexech 1 a 2. `len()` počítá znaky.

Otázka 33

```
jmeno = "  Anna "
print(jmeno.strip())
print(jmeno.upper())
print(jmeno)
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> Anna
>   ANNA
>   Anna
> ```
>
> - `strip()` odstraní mezery na krajích, `upper()` převede na velká písmena. Obě metody vrací **nový** string, původní `jmeno` zůstává i s mezerami.

Otázka 34

```
radek = "Kupní;Nájemní;O dílo"
smlouvy = radek.split(";")
print(smlouvy)
print(len(smlouvy))
print(", ".join(smlouvy))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> ['Kupní', 'Nájemní', 'O dílo']
> 3
> Kupní, Nájemní, O dílo
> ```
>
> - `split(";")` rozdělí string podle středníku na seznam, `", ".join()` seznam zase spojí s čárkou a mezerou.

Otázka 35

```
jmeno = "Anna"
vek = 17
print(f"{jmeno} má {vek} let")
print(f"Za rok bude mít {vek + 1} let")
print(jmeno + " má " + vek + " let")
```

Co vypíše tento program? Pokud spadne, napište na kterém řádku a proč.

> **Vzorová odpověď**
>
> Vypíše první dva řádky a pak spadne na třetím `print`:
>
> ```
> Anna má 17 let
> Za rok bude mít 18 let
> TypeError: can only concatenate str (not "int") to str
> ```
>
> - F-string převede `vek` na string sám a umí i výraz `{vek + 1}`.
> - Při spojování `+` je nutné číslo převést ručně: `str(vek)`.

Otázka 36

```
a = "2026"
b = "20.5"
c = "Anna2"
print(a.isdigit())
print(b.isdigit())
print(c.isalpha())
print(c.isalnum())
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> True
> False
> False
> True
> ```
>
> - `isdigit()` platí jen pro samé číslice, tečka v `"20.5"` ji pokazí.
> - `isalpha()` chce samá písmena (2 vadí), `isalnum()` písmena nebo číslice (`"Anna2"` vyhovuje).

Otázka 37

```
a = "Anna"
b = "anna"
print(a == b)
print(a.lower() == b.lower())
print(len(a) == len(b))
```

Co vypíše tento program?

> **Vzorová odpověď**
>
> Výstup:
>
> ```
> False
> True
> True
> ```
>
> - Porovnání stringů rozlišuje velikost písmen. Po `lower()` jsou oba `"anna"`. Délka je stejná, 4 znaky.

Otázka 38

```
def soucet(seznam: list[int]) -> int:
    celkem = 0
    for cislo in seznam:
        celkem += cislo
    return celkem
```

Jaká je časová složitost této funkce v O-notaci, když má seznam n prvků?

> **Vzorová odpověď**
>
> - **O(n)**, lineární.
> - Jeden cyklus projde každý z n prvků právě jednou a u každého udělá konstantní práci.

Otázka 39

```
def ma_duplicity(seznam: list[int]) -> bool:
    for i in range(len(seznam)):
        for j in range(len(seznam)):
            if i != j and seznam[i] == seznam[j]:
                return True
    return False
```

Jaká je časová složitost této funkce v O-notaci v nejhorším případě, když má seznam n prvků? Jaký vstup je ten nejhorší?

> **Vzorová odpověď**
>
> - **O(n²)**, kvadratická: dva vnořené cykly přes n prvků dají až n · n porovnání.
> - Nejhorší je seznam **bez duplicit**: funkce nikdy nenarazí na `return True` a projde všechny dvojice.

Otázka 40

Rozhodněte u každé složitosti, zda je polynomiální, nebo ne:

1. $O(n^2)$
2. $O(2^n)$
3. $O(n^{10})$
4. $O(1)$

Která z nich je pro velké vstupy nejhorší v praxi?

> **Vzorová odpověď**
>
> - **Polynomiální**: O(n²), O(n¹⁰), O(1).
> - **Nepolynomiální**: O(2ⁿ), roste exponenciálně.
> - Nejhorší je **O(2ⁿ)**: každý prvek vstupu navíc počet kroků zdvojnásobí, pro n = 100 je to už asi 10³⁰ kroků.
> - O(n¹⁰) je také velmi pomalá, ale roste polynomiálně, takže pro velké n zůstává o mnoho řádů pod O(2ⁿ).

Otázka 41

```
seznam = list(range(1_000_000))
mnozina = set(seznam)

print(999_999 in seznam)
print(999_999 in mnozina)
```

Jakou časovou složitost má operátor `in` na prvním a na druhém řádku s `print`? Proč se liší?

> **Vzorová odpověď**
>
> - Seznam: **O(n)**. `in` prochází prvky po jednom od začátku a hledaná hodnota je až na konci.
> - Množina: **O(1)**. Z hodnoty se spočítá hash a podívá se rovnou do správné přihrádky.
> - Proto je na velkých datech `in` nad množinou mnohem rychlejší, i když oba řádky vypíší `True`.

Otázka 42

```
def trojice(seznam: list[int]) -> int:
    pocet = 0
    for a in seznam:
        for b in seznam:
            for c in seznam:
                if a + b + c == 0:
                    pocet += 1
    return pocet
```

Jaká je časová složitost této funkce v O-notaci, když má seznam n prvků? Kolik kroků udělá pro $n = 1\,000$ a kolik pro $n = 10\,000$?

> **Vzorová odpověď**
>
> - **O(n³)**, tři vnořené cykly přes n prvků.
> - Pro `n = 1 000` asi 10^9 (miliarda) kroků, pro `n = 10 000` asi 10^{12} (bilion).
> - Desetkrát větší vstup znamená tisíckrát víc práce.

Otázka 43

Klient má dostat hotovou smlouvu ke čtení a tisku. Záleží na tom, aby stránky vypadaly všude stejně.

Jaký datový formát zvolíte a proč? Zaručuje tento formát, že dokument nikdo neupraví?

> **Vzorová odpověď**
>
> - **PDF**: zachová vzhled stránek na každém zařízení i tiskárně, fonty a rozložení jsou uložené uvnitř.
> - Neupravitelnost **nezaručuje**: PDF lze editovat nebo přepsat. Pravost a nezměněnost zajistí až elektronický podpis.

Otázka 44

Každý měsíc dostáváte od kolegy tabulku smluv se sloupci `id`, `typ`, `castka` a `mesto`. Program ji má automaticky načíst a sečíst částky.

Jaký datový formát si pro předání tabulky vyžádáte a proč? Co si musíte s kolegou dohodnout, aby načtení fungovalo?

> **Vzorová odpověď**
>
> - **CSV**: prostý text, jedna smlouva na řádek, Python ho načte vestavěným modulem `csv`.
> - Dohodnout je třeba: **oddělovač** (čárka nebo středník), **kódování** (UTF-8), **hlavičku** s názvy sloupců, **zápis čísel** (desetinná tečka, bez mezer a měny) a **uvozovky** u hodnot, které oddělovač obsahují.

Otázka 45

```
id,nazev,castka
S04,"Novák, s.r.o.",50000
```

Kolik hodnot má druhý řádek tohoto CSV souboru? K čemu slouží uvozovky? Co by se stalo, kdybychom řádek ručně rozdělili podle každé čárky?

> **Vzorová odpověď**
>
> - Druhý řádek má **tři** hodnoty: `S04`, `Novák, s.r.o.` a `50000`.
> - Uvozovky říkají, že čárka uvnitř je **součást hodnoty**, ne oddělovač sloupců.
> - Při dělení podle každé čárky by vznikly **čtyři** kusy (`"Novák` a `s.r.o."` zvlášť), sloupce by se posunuly a částka by skončila ve špatném sloupci. Proto se CSV čte modulem `csv`, ne `split(",")`.

Otázka 46

```
{
  "id": "S03",
  "castka": 120000,
  "zkontrolovano": false,
  "strany": ["Alfa s.r.o.", "Beta s.r.o."]
}
```

Jakého typu je hodnota u každého ze čtyř polí (string, číslo, logická hodnota, seznam)? Do jaké datové struktury v Pythonu se tento záznam načte?

> **Vzorová odpověď**
>
> - `id`: **string** (`"S03"`), `castka`: **číslo** (`120000`), `zkontrolovano`: **logická hodnota** (`false`), `strany`: **seznam** stringů.
> - V Pythonu vznikne **slovník** (`dict`) s hodnotami `str`, `int`, `bool` a `list`:
>
> ```
> {
>     "id": "S03",
>     "castka": 120000,
>     "zkontrolovano": False,
>     "strany": ["Alfa s.r.o.", "Beta s.r.o."],
> }
> ```

Otázka 47

Napiš funkci, která dostane jako argument seznam čísel a vrací jeho sumu. Nesmíte použít zabudovanou funkci `sum()`

Jakou bude mít časovou složitost?

> **Vzorová odpověď**

```
def suma(seznam: list[int]) -> int:
    celkem = 0
    for cislo in seznam:
        celkem += cislo
    return celkem
```

- Čísla se sčítají ručně v proměnné `celkem`, která začíná na nule; zabudovaná `sum()` se nepoužije.
- Složitost **O(n)**: cyklus projde každý prvek právě jednou a u každého udělá jedno přičtení.

Otázka 48

Napiš funkci, která dostane jako argument seznam intů a vrátí `True` pokud je každé následující číslo v seznamu větší než to předchozí, jinak vrátí `False`.

Jakou má tato funkce časovou složitost?

> **Vzorová odpověď**

```
def rostouci(seznam: list[int]) -> bool:
    for i in range(1, len(seznam)):
        if seznam[i] <= seznam[i - 1]:
            return False
    return True
```

- Porovnává každý prvek s předchozím; při první nerostoucí dvojici skončí.
- Složitost **O(n)**: v nejhorším případě jeden průchod seznamem.