C2184 Úvod do programování v Pythonu

Ukázkový průběžný test (B) – vzorová řešení

Toto je pouze ukázkový test a slouží na procvičení. Řešení není potřeba nikam odevzdávat, níže uvedené pokyny pouze popisují, jak by test probíhal naostro.

Pokyny:

- Během testu je povoleno používat libovolné materiály vč. internetu. Zakázána je pouze komunikace (online i fyzická) s jiným člověkem (kromě učitele).
- Na řešení testu jsou vyhrazeny 2 hodiny, lze získat maximálně 50 bodů.
- Řešení je nutné vyplnit:
  - Přímo do tohoto souboru `ukazkovy_prubezny_test_B.ipynb` (a testy si spustit pomocí testovacích buněk)
  - Nebo každý úkol zvlášť do souborů `prubezny_B_?.py`, kde `?` je číslo úkolu (a testy si spustit z příkazového řádku: `python testing.py prubezny_B_?.py`).
- Pozor, úkoly jsou dvou typů:
  - Napsat program: je třeba načítat vstup pomocí `input` a vypsat výstup pomocí `print`. V zadání je uveden **vzorový vstup** a **vzorový výstup**.
  - Napsat funkci: není třeba načítat vstup (vstupní data budou předána v parametrech funkce), výsledek má být vrácen jako návratová hodnota funkce (`return`). V zadání je uvedeno **vzorové volání** a **vzorový výsledek**. Typové anotace v šablonách funkcí považujte za součást zadání.
- Řešení odevzdejte včas do připravené odevzdávárny.

Úkol 1 (10 bodů)

Napište program, který načte ze vstupu řádek složený z několika obdélníků (každý obdélník je zapsán jako `axb`, kde `a`, `b` jsou délky jeho stran). Program pak vypíše na výstup délky stran `a`, `b` a obsah každého obdélníku v požadovaném formátu (viz vzorový výstup, každý sloupec má šířku 8 znaků, mezi nimi je trojice znaků mezera-svislítko-mezera, hodnoty se vypisují na 2 desetinná místa zarovnané doprava).

Vzorový vstup:

```
1x1 0.4x0.25 80x500
```

Vzorový výstup:

```
Strana a | Strana b |    Obsah
    1.00 |     1.00 |     1.00
    0.40 |     0.25 |     0.10
   80.00 |   500.00 | 40000.00
```

[ ]:
```python
rectangles = input().split()
print('Strana a | Strana b |    Obsah')
for rect in rectangles:
    a, b = rect.split('x')
    a = float(a)
    b = float(b)
    S = a*b
    print(f'{a:8.2f} | {b:8.2f} | {S:8.2f}')

# Pro kopírování:
# 1x1 0.4x0.25 80x500
# Strana a | Strana b |    Obsah
```

Úkol 2 (10 bodů)

Doplňte funkci `my_filter`, která bere jako parametr seznam řetězců (složených pouze z písmen). Návratovou hodnotou funkce bude přefiltrovaný seznam obsahující pouze řetězce začínající velkým písmenem (další písmena v řetězci mohou být velká nebo malá).

Vzorové volání:

```
my_filter(['Hello', 'jaj', 'KONEC', 'mRNA', 'X', 'camelCase', ''])
```

Vzorový výsledek:

```
['Hello', 'KONEC', 'X']
```

[ ]:
```python
from __future__ import annotations

def my_filter(strings: list[str]) -> list[str]:
    return [s for s in strings if len(s) > 0 and s[0].isupper()]

# my_filter(['Hello', 'jaj', 'KONEC', 'mRNA', 'X', 'camelCase', ''])
```

Úkol 3 (10 bodů)

Sekvence DNA se skládá ze čtyř typů nukleových bazí (A, C, G, T). Relativní četnost báze vyjádřuje, jaká část sekvence je tvořena daným typem báze (počet výskytů báze

/ délka sekvence). Součet relativních četností je tedy vždy roven 1.

Napište definici funkce `count_bases`, která vezme jako parametr sekvenci DNA a vrátí slovník s absolutní četností nukleových bazí.

Vzorové volání:

```
count_bases('ACGTTTTGAG')
```

Vzorový výsledek:

```
{'A': 0.2, 'C': 0.1, 'G': 0.3, 'T': 0.4}
```

[ ]:
```python
from __future__ import annotations

def count_bases(sequence: str) -> dict[str, float]:
    n = len(sequence)
    counts = {base: 0 for base in 'ACGT'}
    for base in sequence:
        counts[base] += 1
    return {base: count/n for base, count in counts.items()}

# count_bases('ACGTTTTGAG')
# count_bases('AAA')
```

Úkol 4 (10 bodů)

Napište definici funkce `format_names`, která vezme jako parametr řetězec obsahující jména a příjmení osob oddělené čárkami (před a za čárkou může být libovolný počet mezer, mezi jménem a příjmením je jedna nebo více mezer, každá osoba má pouze jedno jméno a jedno příjmení). Funkce přeformátuje jména osob do formátu `Příjmení, Jméno`, jednotlivé osoby oddělí středníkem a mezerou. Návratovou hodnotou je takto přeformátovaný řetězec.

Vzorové volání:

```
format_names('Cyril   Novák ,    Alice Černá,Bob  Marley')
```

Vzorový výsledek:

```
'Novák, Cyril; Černá, Alice; Marley, Bob'
```

[ ]:
```python
def format_names(text: str) -> str:
    result = []
    for person in text.split(','):
        if person.strip() != '':
            name, surname = person.split()
```

```
            result.append(f'{surname}, {name}')
    return '; '.join(result)
    # Šlo by naformátovaná jména ukládat přímo do řetězce pomocí +=
    # (místo do seznamu pomocí append). Takové řešení by však bylo
    # výrazně méně efektivní (každé použití += vytváří nový řetězec,
    # do kterého se musí zkopírovat starý + pridávané znaky).

# format_names('Cyril   Novák ,    Alice Černá,Bob  Marley')
```

Úkol 5 (10 bodů)

Napište definici funkce `even_odd`, která vezme jako parametr seznam celých čísel. Funkce vrátí:

- řetězec `'even'`, pokud je víc sudých než lichých čísel,
- řetězec `'odd'`, pokud je víc lichých než sudých čísel,
- řetězec `'?'`, pokud je stejný počet sudých a lichých čísel.

Vzorové volání:

```
even_odd([5, 11, 2])
```

Vzorový výsledek:

```
'odd'
```

[ ]:
```
from __future__ import annotations

def even_odd(numbers: list[int]) -> str:
    n_even = sum(1 for i in numbers if i % 2 == 0)
    n_odd = len(numbers) - n_even
    if n_even > n_odd:
        return 'even'
    elif n_odd > n_even:
        return 'odd'
    else:
        return '?'

# even_odd([5, 11, 2])
# even_odd([100])
# even_odd([5, 10, -3, 8, -1, 0])
```