C2184 Úvod do programování v Pythonu

Ukázkový průběžný test (A) – 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_A.ipynb` (a testy si spustit pomocí testovacích buněk)
  - Nebo každý úkol zvlášť do souborů `prubezny_A_?.py`, kde `?` je číslo úkolu (a testy si spustit z příkazového řádku: `python testing.py prubezny_A_?.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 (26 bodů)

Naším úkolem v této úloze je vytvořit program, který ze vstupu načte soubor hodnot (na jednom řádku) a vypíše na výstup statistický přehled k tomuto souboru.

Vstupní řetězec bude obsahovat jednotlivé prvky souboru (řetězce) oddělené mezerami.

- Načtěte řádek s hodnotami ze vstupu a **převeďte na seznam řetězců** (2)

- Vypište **rozsah souboru** (počet prvků) (1)
- Vypište **počet různých hodnot a výčet těchto hodnot** (abecedně) (4)
- Vypište **nejkratší a nejdelší prvek** (předpokládejte, že bude právě jeden nejkratší a jeden nejdelší) (4)
- Vypište jednoduchý sloupcový graf, tj. ke každé hodnotě:
  - Vypište **relativní četnost** v procentech (= počet výskytů hodnoty / rozsah celého souboru * 100%) (3)
  - Znázorněte **graficky počet výskytů** (1 výskyt = 1 symbol #) (3)
  - Hodnoty řaďte **od nejčetnější po nejméně četnou** (4)
- **Formátování** co nejvíc přibližte vzorovému výstupu (5)

Čísla v závorkách udávají maximální počet bodů za každý podúkol. Na uznání konkrétního podúkolu nemusíte nutně splnit předchozí podúkoly – například pokud si neumíte poradit s převodem řetězce na seznam hodnot, zadefinujte si prostě seznam natvrdo a pokuste se vyřešit další podúkoly. Při řešení si samozřejmě můžete zadefinovat i pomocné funkce.

Tip: defaultní řazení n-tic v Pythonu je podle prvního prvku (pokud je první prvek stejný, tak podle druhého atd.)

Vzorový vstup:

```
red green purple green green purple red purple red green red green
```

Vzorový výstup:

```
Sample size:     12
Distinct values: 3 (green, purple, red)
Shortest:        red
Longest:         purple
Graph:
    green :  42% #####
    red   :  33% ####
    purple:  25% ###
```

[ ]:
```python
values = input().split()

n = len(values)
distinct = sorted(set(values))
shortest = min(distinct, key=len)
longest = max(distinct, key=len)
counts = {value: 0 for value in distinct}
for value in values:
    counts[value] += 1
counts = [(count, value) for value, count in counts.items()]
counts.sort(reverse=True)
max_length = len(longest)
```

```
print(f'Sample size:     {n}')
print(f'Distinct values: {len(distinct)} ({", ".join(distinct)})')
print(f'Shortest:        {shortest}')
print(f'Longest:         {longest}')
print(f'Graph:')
for count, value in counts:
    rel_count = count / n
    bar = '#' * count
    print(f'    {value:<{max_length}}: {rel_count:4.0%} {bar}')

# Vzorový vstup pro kopírování:
# red green purple green green purple red purple red green red green
```

Úkol 2 (6 bodů)

Doplňte funkci `my_filter`, která bere jako parametr seznam celých čísel. Návratovou hodnotou funkce bude přefiltrovaný seznam obsahující pouze čísla, která jsou dělitelná třemi ale ne devíti.

Vzorové volání:

```
my_filter([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

Vzorový výsledek:

```
[3, 6, 12]
```

[ ]:
```
from __future__ import annotations

def my_filter(numbers: list[int]) -> list[int]:
    return [n for n in numbers if n%3 == 0 and n%9 != 0]

# my_filter([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

Úkol 3 (10 bodů)

V tomto úkolu budeme řešit dvojsměrku – jednodušší verzi osmisměrky. Zadání dvojsměrky sestává z dlouhého řetězce písmen a seznamu hledaných slov. Tato slova potřebujeme najít v řetězci a škrtnout je. Slova mohou být v řetězci zapsána zleva doprava nebo zprava doleva. Tajenku pak získáme spojením zbylých neškrtnutých písmen.

Doplňte funkci `solve_puzzle`, která bere dva argumenty – řetězec písmen a seznam hledaných slov. Návratovou hodnotou funkce bude vyluštěná tajenka.

Příklad dvojsměrky:

ZFUNKCENAVTŠIMOSTROMMVÝLETAŠYM

Hledaná slova: FUNKCE MOST MYŠ STROM VÝLET ŠTVANEC

Řešení:

Z ~~F U N K C E N A V T Š~~ I ~~M O S T R O M~~ M ~~V Ý L E T~~ A ~~Š Y M~~

Po vyškrtnutí všech hledaných slov jsme našli tajenku: ZIMA.

Vzorové volání:

```
solve_puzzle('ZFUNKCENAVTŠIMOSTROMMVÝLETAŠYM', ['FUNKCE', 'MOST', 'MYŠ', 'STROM',
```

Vzorový výsledek:

```
'ZIMA'
```

[ ]:
```python
from __future__ import annotations

def solve_puzzle(text: str, words: list[str]) -> str:
    crossed = [False for c in text]
    for word in words:
        i = text.find(word)
        if i < 0:
            i = text.find(word[::-1])
        for j in range(i, i+len(word)):
            crossed[j] = True
    return ''.join(c for i, c in enumerate(text) if not crossed[i])

# solve_puzzle('ZFUNKCENAVTŠIMOSTROMMVÝLETAŠYM', ['FUNKCE', 'MOST',␣
  ↪'MYŠ', 'STROM', 'VÝLET', 'ŠTVANEC'])  # 'ZIMA'
# solve_puzzle('AZÁVSVBANÁNAPOKRMAKOŽÍŠEKARAMŘLIBOMÁZUBYK', ['BANÁN',␣
  ↪'KOPANÁ', 'KOŽÍŠEK', 'MARAKEŠ', 'MOBIL', 'POKRM', 'VÁZA', 'ZUBY'])␣
  ↪ # 'SVAŘÁK'
# solve_puzzle('ŽALHICHLAPATITCPESPEKSUKRUMORRSOLONOHTYPVČIBALGINÍ',␣
  ↪['APATIT', 'BIČ', 'CHLAP', 'CIHLA', 'IBALGIN', 'LOS', 'MOR',␣
  ↪'PES', 'PYTHON', 'RUM', 'SKEPSE', 'ŽAL'])  # 'CUKROVÍ'
```

Úkol 4 (8 bodů)

Napište program, který načte ze vstupu jeden řádek složený z reálných čísel, oddělených libovolným počtem mezer. Z těchto čísel pak vybere pouze čísla v intervalu 0 až 10 (včetně), spočítá jejich součet, a tento součet vypíše na výstup jako reálné číslo se dvěma desetinnými místy. Počet čísel není omezený (může být i nula čísel, pak je součet nula).

Vzorový vstup:

```
3.14 11 -3.5 9
```

Vzorový výstup:

```
12.14
```

[ ]:
```python
numbers = [float(s) for s in input().split()]
numbers = [x for x in numbers if 0 <= x <= 10]
print(f'{sum(numbers):.2f}')

# 3.14 11 -3.5 9
```