Řešení domácích úkolů - 3. týden

Maximum - řádky

Na vstupu dostanete posloupnost celých čísel ukončených -1, která již do seznamu nepatří. Vypište největší z nich. Číslo bude alespoň jedno. Každé číslo je zapsáno na samostatném řádku. Všechna čisla se vejdou do typu integer.

Řešení

Analýza Načítáme sérii čísel, která můžou být libovolná, kladná, záporná i 0. Pouze číslo -1 je výjimečné, protože při jeho objevení na vstupu načítání posloupnosti končí.

Pro nalezení maxima nepotřebujeme ukládat celou posloupnost - můžeme průběžně aktualizovat největší dosud načtenou hodnotu, která po načtení posloupnosti bude rovna jejímu maximu. Zadání řeší problém inciializace průběžného maxima - protože "číslo bude alespoň jedno", můžeme toho využít a inicializovat průběžné maximum první načtenou hodnotou.

Vzorové řešení

```
maximum = int(input()) # prubezne maximum, první hodnota urcite != -1
while (n := int(input())) != -1:
    if n > maximum:
        maximum = n
print(maximum)
```

Alternativní řešení

Řada z vás volila načtení hodnot do seznamu. Pro tuto úlohu je to plýtvání pamětí a pro dostatečně velká vstupná data nemusí takovéto řešení fungovat.

```
hodnoty = []
while True:
    n = int(input())
    if n == -1:
        break
    hodnoty.append(n)
print(max(hodnoty))
```

Používáme "superfunkci" max, která vrací maximum z kolekce hodnot - seznamu, pole, množiny, n-tice a pod.

Našlo se i řešení s nezvykle řešeným načítáním:

```
import sys

lines = sys.stdin.readlines()     # načteme vstup najednou jako seznam
                                  # řádek
numbers = [int(l) for l in lines]
numbers.pop()                     # odstraníme z konce -1
print(max(numbers))
```

Tady přistupujeme k standardnímu vstupu jako k souboru. Tak se chová i zadávání dat - zatímco u vstupu pomocí funkce `input` postupně píšeme jednotlivá čísla a pokaždé stiskneme Enter, tady data najednou napíšeme v požadovaném formátu (číslo na řádek, na posledním řádku -1) a stiskneme Ctrl-D. Pro úlohy, kde potřebujeme všechna data najednou načíst do paměti, je toto rovnocenný přístup, zatímco pro sekvenční vstup takovéto načítání dat poněkud ruší smysl úlohy.

Obvyklé problémy v řešeních

Kritizoval jsem řešení, které pro nalezení maxima používali setřídění posloupnosti.

```
hodnoty = []
while True:
    n = int(input())
    if n == -1:
        break
    hodnoty.append(n)
print(sorted(hodnoty)[-1])
```

Na nalezení maxima neuspořádané kolekce potřebujeme nějaký násobek n operací, na setřídění násobek $n \log n$. Jednoduše řečeno, pro nalezení maxima určitě nepotřebujeme znát pořadí malých hodnot.

Několikrát se v odevzdaných řešení vyskytla takováto porucha:

```
maximum = int(input())
n = int(input())
while n:
    if n == -1:
        break
    if maximum < n:
        maximum = n
    n = int(input())
print(maximum)
```

Toto je zjevně vedeno záměrem dodatečně zabezpečit, aby se načítání ukončilo, když už nejsou žádné hodnoty n. Bohužel to tak nefunguje: pokud by se při načítání přišlo na konec vstupu (např. by uživatel nezadal číslo ale jednom stisknul Enter nebo Ctrl-D nebo Ctrl-Z, nebo by se došlo na konec vstupního souboru), nastala by havárie nejdřív ve funkci `input`. Je to tedy zbytečně defenzivní přístup, ale to by nevadilo: skutečná porucha je, že tento kód přeruší načítáni, když se na vstupu objeví nula, tedy zcela přípustná hodnota.