ZBIERKA ÚLOH - ZADANIA

Úloha 1 *Rodina Nováková sa v januári 2018 rozhodla, že časť svojich úspor vloží do banky AK-NAB. Táto banka ponúka zaujímavý produkt: sumu na účte na konci každého roka zúročí úrokovou mierou, ktorá je najvyššia na súčasnom finančnom trhu. Vytvorte program, ktorý vypočíta a vypíše Novákovcom výšku ich investície na konci sledovaného obdobia (celé roky).*

Program načíta z konzoly výšku investície, úrokovú mieru (= o aké percento narastie aktuálny stav na účte na konci roka) a počet rokov, počas ktorých chceme nechať investované peniaze „rásť" . Do konzoly program vypíše stav účtu na konci sledovaného obdobia.

Úloha 2 *Jožka na hodine matematiky veľmi zaujala tzv. Fibonacciho postupnosť a jej prítomnosť v prírode a v umení. Rozhodol sa, že ju aplikuje aj do športu – konkrétne do svojho tréningového plánu. Každé ráno bude cvičiť kľuky, pričom každý pondelok zvýši ich počet. Pri zvyšovaní počtu kľukov sa bude riadiť práve Fibonacciho postupnosťou (t.j. 1. týždeň urobí každé ráno jeden kľuk, 2. týždeň tiež jeden kľuk, 3. týždeň dva kľuky, 4. týždeň 3 kľuky, 5. týždeň päť kľukov atď., 6. týždeň osem kľukov, 7. týždeň trinásť kľukov atď.).*

Vytvorte program, ktorý pre zadaný počet týždňov vypíše tréningový plán vo forme tabuľky. V prvom stĺpci vypíše poradové číslo týždňa, v druhom stĺpci počet kľukov prislúchajúci tomuto týždňu. Otestujte, či program pracuje správne. Ak je to potrebné, ošetrite, prípadne generujte potrebné výnimky. Definujte vhodne funkciu pre výpočet počtu kľukov v danom týždni.

Úloha 3 *Jožko videl v televízii reklamu finančnej inštitúcie Zbohatni. Tá ponúka zaujímavú možnosť investovania: na začiatku prvého mesiaca investujete 1 € a následne bude suma na vašom konte na začiatku každého ďalšieho mesiaca zdvojnásobená. (t.j. po 1. mesiaci budete mať na účte 2 €, po 2. mesiaci 4 €, po 3. mesiaci 8 €, po 4. mesiaci 16 € atď.).*

Jožko bol zvedavý, ako rýchlo porastie jeho investícia, vytvoril preto program ***13_05_investicia.py****, ktorý pomocou funkcie* `hodnota_na_konci_mesiaca()` *pre zadaný počet mesiacov investovania vypíše hodnotu investície na konci tohto obdobia.*

a) *Otestujte, či Jožkov program pracuje správne. Ak nájdete logickú chybu, opravte ju.*

b) *Upravte program tak, aby zobrazil vývoj investície počas celej doby investovania (tú určuje zadaný počet mesiacov investovania) vo forme tabuľky. V prvom stĺpci vypíše poradové číslo mesiaca, v druhom stĺpci aktuálnu hodnotu investície.*

ZBIERKA ÚLOH - RIEŠENIA

Úloha 3 *Rodina Nováková sa v januári 2018 rozhodla, že časť svojich úspor vloží do banky AK-NAB. Táto banka ponúka zaujímavý produkt: sumu na účte na konci každého roka zúročí úrokovou mierou, ktorá je najvyššia na súčasnom finančnom trhu. Vytvorte program, ktorý vypočíta a vypíše Novákovcom výšku ich investície na konci sledovaného obdobia (celé roky).*

Program načíta z konzoly výšku investície, úrokovú mieru (= o aké percento narastie aktuálny stav na účte na konci roka) a počet rokov, počas ktorých chceme nechať investované peniaze „rásť“ . Do konzoly program vypíše stav účtu na konci sledovaného obdobia.

Riešenie:

```
def vypocet(vklad, urok, roky):
    try:
        vklad = float(vklad)
        urok = float(urok)
        roky = int(roky)
    except ValueError:
        #neciselny vstup
        raise ValueError('Nečíselná hodnota pre výšku vkladu, úrokovú mieru alebo
počet rokov.')
    if vklad <= 0 or urok <= 0 or roky <= 0:
        #nekladne hodnoty na vstupe
        raise ValueError('Niektorá zo vstupných hodnôt bola nekladná.')
    #vypocet stavu uctu
    for i in range(roky):
            vklad = vklad * (1 + urok / 100)
    return vklad


#nacitanie vstupnych hodnot
vklad = input('Zadaj výšku investície: ')
urokova_miera = input('Zadaj úrokovú mieru: ')
pocet_rokov = input('Dĺžka obdobia investície (celé roky): ')

try:
    print(f'Výška investície po {pocet_rokov} rokoch: {vypocet(vklad,
urokova_miera, pocet_rokov)} EUR')
except ValueError as chyba:
    print(chyba)
```

Úloha 4 *Jožka na hodine matematiky veľmi zaujala tzv. Fibonacciho postupnosť a jej prítomnosť v prírode a v umení. Rozhodol sa, že ju aplikuje aj do športu – konkrétne do svojho tréningového plánu. Každé ráno bude cvičiť kľuky, pričom každý pondelok zvýši ich počet. Pri zvyšovaní počtu kľukov sa bude riadiť práve Fibonacciho postupnosťou (t.j. 1. týždeň urobí každé ráno jeden kľuk, 2. týždeň tiež jeden kľuk, 3. týždeň dva kľuky, 4. týždeň 3 kľuky, 5. týždeň päť kľukov atď., 6. týždeň osem kľukov, 7. týždeň trinásť kľukov atď.).*

Vytvorte program, ktorý pre zadaný počet týždňov vypíše tréningový plán vo forme tabuľky. V prvom stĺpci vypíše poradové číslo týždňa, v druhom stĺpci počet kľukov prislúchajúci tomuto týždňu. Otestujte, či program pracuje správne. Ak je to potrebné, ošetrite, prípadne generujte potrebné výnimky. Definujte vhodne funkciu pre výpočet počtu kľukov v danom týždni.

Riešenie:

```
def pocet_klukov(n):
    try:
        n = int(n)
    except ValueError:
        raise ValueError('Počet týždňov musí byť číslo.')
    if n <= 0:
        raise ValueError('Na vstupe nebolo prirodzené číslo.')
    if n == 1:
        return 1
    f1 = 0
    f2 = 1
    for i in range(n):
        f = f1 + f2
        f1 = f2
        f2 = f
    return f


pocet_tyzdnov = input('Počet týždňov tréningového plánu: ')
try:
    pocet_tyzdnov = int(pocet_tyzdnov)
except ValueError:
    print('Na vstupe bol nečíselný/neceločíselný vstup.')
else:
    try:
        for i in range(pocet_tyzdnov):
            print(f'{i + 1}. týždeň: {pocet_klukov(i + 1):5}')
    except ValueError as chyba:
        print(chyba)
```

Fibonacciho postupnosť tvoria jej členy: 1, 1, 2, 3, 5, 8, 13, ..., kde každý nový člen (s výnimkou prvých dvoch) je súčtom dvoch svojich najbližších predchodcov ($a_1 = 1$, $a_2 = 1$, $a_n = a_{n-1} + a_{n-2}$). Keďže predpis ako vypočítať nový člen platí až od tretieho člena, museli sme špeciálne ošetriť 1. týždeň a potom sme si pomohli malou fintou, keď sme za prvý člen postupnosti určili číslo 0.

Je veľmi pravdepodobné, že žiakov potrápi cyklus `for` vo funkcii `pocet_klukov()` (ak nie, môžeme túto situáciu navodiť tým, že to potrápilo nás). Pri hľadaní problematického miesta v programe využijeme nástroj krokovania.

Úloha 3 *Jožko videl v televízii reklamu finančnej inštitúcie Zbohatni. Tá ponúka zaujímavú možnosť investovania: na začiatku prvého mesiaca investujete 1 € a následne bude suma na vašom konte na začiatku každého ďalšieho mesiaca zdvojnásobená. (t.j. po 1. mesiaci budete mať na účte 2 €, po 2. mesiaci 4 €, po 3. mesiaci 8 €, po 4. mesiaci 16 € atď.).*

Jožko bol zvedavý, ako rýchlo porastie jeho investícia, vytvoril preto program ***13_05_investicia.py****, ktorý pomocou funkcie* `hodnota_na_konci_mesiaca()` *pre zadaný počet mesiacov investovania vypíše hodnotu investície na konci tohto obdobia.*

c) Otestujte, či Jožkov program pracuje správne. Ak nájdete logickú chybu, opravte ju.

d) Upravte program tak, aby zobrazil vývoj investície počas celej doby investovania (tú určuje zadaný počet mesiacov investovania) vo forme tabuľky. V prvom stĺpci vypíše poradové číslo mesiaca, v druhom stĺpci aktuálnu hodnotu investície.

```
def hodnota_na_konci_mesiaca(mesiac):
    try:
        #overujeme celociselnost
        mesiac = int(mesiac)
    except ValueError:
        raise ValueError('Počet mesiacov nie je číslo.')
    if mesiac <= 0:
        raise ValueError('Počet mesiacov nie je prirodzené číslo.')
    hodnota = 2 * mesiac
    return hodnota


#pocet mesiacov investovania
pocet_mesiacov = input('Počet mesiacov: ')
print(f'na konci {pocet_mesiacov}. mesiaca:
{hodnota_na_konci_mesiaca(pocet_mesiacov):5}')
```

Riešenie:

Úloha je určená najmä tým žiakom, ktorí radi skúmajú nové situácie. V tomto prípade síce program nevytvárajú, len opravujú, majú však možnosť preskúmať situáciu, kedy je potrebné ošetriť výnimky aj v hlavnom programe (ošetrenie výnimiek sme v predchádzajúcich úlohách riešili len vo funkcii). Keďže ale teraz voláme funkciu v cykle `for`, musíme hodnotu, na ktorú odkazuje premenná `pocet_mesiacov`, otestovať pred týmto príkazom.

Vo vytvorenom programe nájdu žiaci jednu logickú chybu – nesprávny výpočet výsledku vo funkcii. Po jej opravení môžu riešiť úlohu b). Tu je nutné použiť cyklus `for` (keďže chceme vývoj investície sledovať priebežne, po mesiacoch). Narazíme na výnimky, ktoré síce ošetruje funkcia `hodnota_na_konci_mesiaca()`, ale tu, v hlavnom programe, ich musíme kvôli cyklu `for` tiež generovať.

```
def hodnota_na_konci_mesiaca(mesiac):
    try:
        #overujeme celociselnost
        mesiac = int(mesiac)
    except ValueError:
        raise ValueError('')
```

```
    if mesiac <= 0:
        raise ValueError('Počet mesiacov nie je prirodzené číslo.')
    hodnota = 2 ** mesiac
    return hodnota


#pocet mesiacov investovania
pocet_mesiacov = input('Počet mesiacov: ')
try:
    #testujeme vstup - aby sme mohli v cykle for vypisat pozadovane sumy
    pocet_mesiacov = int(pocet_mesiacov)
except ValueError:
    print('Počet mesiacov nie je číselná hodnota.')
else:
    #testujeme, ci je hodnota kladna
    if pocet_mesiacov <= 0:
        print('Počet mesiacov nie je prirodzené číslo.')
    else:
        #ak je hodnota prirodzene cislo, vypiseme vysledky vypoctu
        try:
            for i in range(1, pocet_mesiacov + 1):
                print(f'na konci {i}. mesiaca: {hodnota_na_konci_mesiaca(i):5}')
        except ValueError as chyba:
            print(chyba)
```