Приклади опрацювання виключень

Приклад 1. В один рядок надрукували декілька чисел. Перетворити рядок в список чисел.

Вірогідна проблема: запис окремих чисел не відповідає правилам, наприклад:

```
textxn = '5  6  -4  3A  24  1.2'   # задано
vector = [ int(x) for x in textxn.split() ]
print(vector)
```

В такому варіанті отримаємо повідомлення про помилку `ValueError: ... '3A'`. Для автоматичного виправлення подібних помилок потрібно вирішити, як це зробити. Наприклад, помилкові записи чисел подавати як числа нуль:

```
textxn = '5  6  -4  3A  24  1.2'   # задано
vector = []  # будуємо список чисел по одному
for x in textxn.split() :
    try :
        vector.append(int(x))
    except ValueError :
        vector.append(0)
print(vector)
```

Тепер отримаємо: `[5, 6, -4, 0, 24, 0]`.

Приклад 2. На основі заданого словника перекладу і списку слів надрукувати переклади слів.

Вірогідна проблема: у словнику відсутні переклади деяких слів. Вирішення: друкувати прочерки:

```
dictEU = { 'show':'показувати', 'merry':'черешня', 'bird':'птах',
'let':'дозволяти' }
ForTranslation = "bird merry book show task"
for word in ForTranslation.split() :
    try :
        print(word, ':', dictEU[word])
    except KeyError :
        print(word, ':', '--')
pass
```

Буде надруковано:

```
bird : птах
merry : черешня
book : --
show : показувати
task : --
```

Збудження виключень

Розглянуті вище особливі ситуації виникали в результаті операцій, які не може виконати програма. Якщо бути точним, то особливі ситуації збуджував інтерпретатор – виконавець алгоритму. Прикладна програма має можливість самостійно збуджувати виключення інструкцією raise. Такий прийом часто використовують для налагодження програм, а також для продовження стану особливої ситуації після блоку except. Приклад:

```
test = 'abcdefgh'
try :
    word1 = test[:3] + '****'
    print(word1)
    word2= 'second:' + test[4]
    raise IndexError    # перевірка реагування
    print(word2)
```

```
except IndexError :
    print('перевірка:')
    #raise IndexError   # якщо треба продовжити стан помилки
    #raise    # або так - помилка останнього типу
print('продовжуємо...')
pass
```

Протокол:

```
abc****
перевірка:
продовжуємо...
```

Інструкція assert

Мова Python має додаткову інструкцію *assert*, як різновид умовної інструкції raise. Інструкцію assert використовують лише в режимі налагодження програми, переважно для перевірки допустимості даних, а не для справжніх помилок. Формат інструкції:

```
assert <test>, <data>      # частина <data>  необов'язкова
```

Така інструкція еквівалентна до наступного фрагмента:

```
if   __debug__  :    # виникає лише в режимі налагодження
    if  not  <test>  :    # виникає при невиконанні умови
        raise  AssertionError(<data>)    # тип + інформація
```

exception `AssertionError` - спеціальне виключення для випадку оператора assert.
Ім'я `__debug__` – це є вбудована ознака, яка автоматично отримує значення True. Для відключення всіх інструкцій assert потрібно при запуску програми через командний рядок вимкнути ознаку: `python -O main.py`

Присвоєння вбудованим величинам в програмі заборонено.

Приклад: перевірка обмежень (але не помилок виконання):

```
# інструкція assert
def anykoef(k) :
    assert  k>=1, 'anykoef - коефіцієнт менший 1'
    return  3*k-0.5
# ----------------------------------------
import sys
try :
    print(anykoef(2))
    print(anykoef(0.6))
except :   # без зазначення типу - для всіх (решти)
    info = sys.exc_info()
    print(info[1])
pass
```

Отримаємо:

```
5.5
anykoef - коефіцієнт менший 1
```