電子計算機入門

池田, 大輔
九州大学附属図書館

電子計算機入門
第6回(06/09)

池田 大輔

z4id01in@cse.ec.kyushu-u.ac.jp

附属図書館

目次

- 第１回レポートの回答例
- 前回課題の回答例
- 頻度カウント
- 頻度によるソート

第1回レポートの解答例

- 与えられた文字列のすべての部分文字列を出力するプログラムを書け

第1回レポートの解答例

- 与えられた文字列のすべての部分文字列を出力するプログラムを書け

```
import sys
str = sys.argv[1]
for beg in range(len(str)):
    for end in range(beg, len(str)):
    # range の開始位置を変更
        print str[beg:end+1]
```

第1回レポートの解答例

- **与えられた文字列のすべての部分文字列を出力するプログラムを書け**

```
import sys
str = sys.argv[1]
for beg in range(len(str)):
    for end in range(beg, len(str)):
    # range の開始位置を変更
        print str[beg:end+1]
# これでは重複が生じてしまう!
```

第1回レポートの解答例(Cont.)

```
import sys
str = sys.argv[1]
res = [] # 結果を格納する配列を初期化
for beg in range(len(str)):
    for end in range(beg, len(str)):
        if str[beg:end+1] not in res:
        # まだ追加していない時のみ追加する
            res.append(str[beg:end+1]) # 追加
for i in range(res):
        print i # 各要素を出力
```

第1回レポートの解答例(Cont.)

```
import sys
str = sys.argv[1]
res = [] # 結果を格納する配列を初期化
for beg in range(len(str)):
    for end in range(beg, len(str)):
        if str[beg:end+1] not in res:
        # まだ追加していない時のみ追加する
            res.append(str[beg:end+1]) # 追加
for i in range(res):
        print i # 各要素を出力
```

第1回レポートの解答例(Cont.)

```
import sys
str = sys.argv[1]
res = [] # 結果を格納する配列を初期化
for beg in range(len(str)):
    for end in range(beg, len(str)):
        if str[beg:end+1] not in res:
        # まだ追加していない時のみ追加する
            res.append(str[beg:end+1]) # 追加
for i in range(res):
        print i # 各要素を出力
```

前回課題の解答例

- ファイルの中身を、行ごとに辞書式順序に並べかえて出力せよ

前回課題の解答例

- ファイルの中身を、行ごとに辞書式順序に並べかえて出力せよ

```
import sys
f = open(sys.argv[1], 'r')
all = f.readlines() # 読み込み
all.sort() # 並べかえ
for i in all:
    print i, # 最後のカンマに注意
print # 改行
```

頻度カウント

- あなたは、ある自動車メーカーから依頼を受け、街頭を走る車の台数を車種ごとに調べないといけません。

  あなたは車に詳しく、実物を見ればその車種名は即座に分かるものとします。

問題

- どのような紙を用意するか？
- 新たに一台の車を見たときに、どのような処理をすればよいか？

頻度カウント

- あなたは、ある自動車メーカーから依頼を受け、街頭を走る車の台数を車種ごとに調べないといけません。

  あなたは車に詳しく、実物を見ればその車種名は即座に分かるものとします。

問題
- どのような紙を用意するか？
- 新たに一台の車を見たときに、どのような処理をすればよいか？

回答例
1. 世の中の全ての車が書かれた紙を用意し、一台見るたびにその車の書かれた欄を探し「正」に加える。

頻度カウント

- あなたは、ある自動車メーカーから依頼を受け、街頭を走る車の台数を車種ごとに調べないといけません。

  あなたは車に詳しく、実物を見ればその車種名は即座に分かるものとします。

問題

- どのような紙を用意するか？
- 新たに一台の車を見たときに、どのような処理をすればよいか？

回答例

1. 世の中の全ての車が書かれた紙を用意し、一台見るたびにその車の書かれた欄を探し「正」に加える。
2. 新たな車を見るごとに車種名を空欄に追加し、「正」の「一」を書く。すでに名前を書いていたら単に「正」に追加。

多次元配列による付加情報の追加

- 文字列や数値だけでなく配列も要素になりうる
  - 二次的な情報を含めた多次元配列にする
- さきほどの車の問題では

多次元配列による付加情報の追加

- 文字列や数値だけでなく配列も要素になりうる
  - 二次的な情報を含めた多次元配列にする
- さきほどの車の問題では

```
cars = [("カローラ", 58), ("MPV", 28), ...]
```

とし、新たに車を見たら次の処理を行なう

1. `cars` の中に車があるか調べる
2. あれば `("車", n)` という要素があるはずなので、`n` を `n+1` にする
3. なければ `("車", 1)` という要素を追加する

より具体的には...

- 新たに車 (*new* とする) を見たら

```
i == 0
while i < len(cars) :  # cars の全要素
    (car, n) = cars[i] # 一つの要素を取り出す
    if car == new:
        cars[i] == (car, n+1)
        break # 見つかれば直ちに反復を停止
else: # break が呼ばれなかった時に、ここが呼ばれる
    cars.append((new, 1))
```

() と [] の違い

- [] で作られるものは配列であり、可変 (追加や変更可能)
- () で作られるものはタプル (tuple) であり、不変 (追加や変更不可)
- タプルの例
  - さきほどの cars の各要素
    - 各要素は “車種名” と “台数” の 2 つであり、要素を増やす必要はない
  - 関数の引数 (len() など)

実習：頻度カウント

- 引数に文字列を与え、全ての部分文字列とその頻度を出力するプログラムを作れ
- 入力として “babaa” を与えた場合、

```
('b', 2)
('ba', 2)
('bab', 1)
('baba', 1)
('babaa', 1)
('a', 3)
('ab', 1)
('aba', 1)
('abaa', 1)
('baa', 1)
('aa', 1)
```

という出力が得られる。

実習：頻度カウント(Cont.)

```
import sys
str = sys.argv[1]
res = []

for b in range(len(str)):
    for e in range(b+1, len(str)+1):
        # 2重forループで全ての部分文字列を生成
        # str[b:e]が新たな一台
        i = 0
        while i < len(res) : # 既出かどうかチェック
            (substr, n) = res[i]
            if substr == str[b:e]:
                res[i] = (substr, n+1)
                break
            i = i + 1
        else:
            res.append((str[b:e], 1))

for i in res:
    print i
```

実習：頻度カウントとソート

- 先のプログラムを以下のように改良して、実行結果を確認しなさい。
- プリントアウトする箇所

```
for i in res:
    print i
```

を

```
res.sort() # この行を追加
for i in res:
    print i
```

に変更。

頻度によるソート

- さきほどの `res.sort()` は頻度の順に並ばなかった
- 多次元配列の場合は、各要素の先頭の要素で並べられる
  - 二次的な情報を先に配置する

頻度によるソート

- さきほどの *res*.sort() は頻度の順に並ばなかった
- 多次元配列の場合は、各要素の先頭の要素で並べられる
  - 二次的な情報を先に配置する

```
array = [(3, "baa"), (4, "accc"),
        (3, "abc"), (3, "bab")]
# (長さ, 文字列) の配列
array.sort() # 長さの小さい順
array.reverse() # 逆順
print array
```

頻度によるソート

- さきほどの *res*.sort() は頻度の順に並ばなかった
- 多次元配列の場合は、各要素の先頭の要素で並べられる
  - 二次的な情報を先に配置する

```
array = [(3, "baa"), (4, "accc"),
        (3, "abc"), (3, "bab")]
```

(長さ, 文字列) の配列

```
array.sort() # 長さの小さい順
array.reverse() # 逆順
print array
```

→

```
[(4,'accc'),(3,'bab'),(3,'baa'),(3,'abc')]
```

実習：文字列配列のソート

- **入力として文字列を複数与え、長さの長い順に出力せよ**

  入力が “baa accc bc bab” **の時は** “bc baa bab accc” **と出力される**

実習：文字列配列のソート

- **入力として文字列を複数与え、長さの長い順に出力せよ**

  入力が “baa accc bc bab” **の時は** “bc baa bab accc” **と出力される**

```
import sys
res = []
for i in sys.argv:
    res.append((len(i), i)) # (長さ, 要素)
res.sort()
res.reverse()
print res
```

今日の課題

- 入力として文字列を一つ与えたとき、この中に現われるすべての部分文字列とその頻度を出力せよ。出力の順番は頻度の高い順とする。