九州大学学術情報リポジトリ
Kyushu University Institutional Repository

電子計算機入門

池田, 大輔
九州大学情報基盤センター

https://hdl.handle.net/2324/6097

出版情報：2003
バージョン：
権利関係：

電子計算機入門
第9回

池田 大輔

daisuke@cc.kyushu-u.ac.jp

情報基盤センター

目次

- 配列による頻度カウント
  - 前回課題の回答例
- 辞書
  - 辞書と配列の効率の違い
  - 辞書のソート
  - その他
- 第 2 回レポート課題発表

■復習：頻度カウントの基本的な方法

```
# 'str' の頻度を調べる場合
for i in in range(len(count)):

    (n, v) = count[i]
    if == str: # すでに記録済みの場合
        count[i] = (n+1, v) #値を増やす
        break)
else: # 記録していない場合
    count.append((1, str))
```

■復習：頻度カウントの基本的な方法

```
# 'str' の頻度を調べる場合
for i in in range(len(count)):全部にアクセス
    (n, v) = count[i]
    if == str: # すでに記録済みの場合
        count[i] = (n+1, v) #値を増やす
        break)
else: # 記録していない場合
    count.append((1, str))
```

■復習：頻度カウントの基本的な方法

```
# 'str' の頻度を調べる場合
for i in in range(len(count)):全部にアクセス
    (n, v) = count[i]
    if == str: # すでに記録済みの場合
        count[i] = (n+1, v) #値を増やす
        break)
else: # 記録していない場合
    count.append((1, str))
```

- 上の for の代りに if v in count: とはできない
  - count の要素は v ではなく (n, v) だが、n の値は具体的には予測できない!

前回課題の解答例

- 長さ 2 の部分文字列の頻度をカウントするプログラム “CountByList.py” をコピーして、頻度カウントを行なう部分を完成させなさい
  - 授業の Web ページに掲載

    http://www.cse.ec.kyushu-u.ac.jp/~z3id01in/

辞書

- マニュアル「2.1.6 マッピングタイプ」やチュートリアル「5.4 辞書」を参照
- ’物’と値の対応表
  - ハッシュ、連想配列などとも呼ばれる
  - 「記録ずみかどうか」が一瞬で分かる
- キーで添字づけした複数のデータ
  - キーは文字列や数字など、変更不能であれば何でもよい
    - 配列は変更可能なので、キーにはできない
  - 配列のキーは、0 以上の連続な自然数

■頻度カウントの基本的な方法：辞書

```
# 辞書は{}で作る (配列は [])
dict = { } # 空辞書を用意
# 新たな' 物'v がきた場合
if dict.has_key(v): # v がキーとして存在するか？
    dict[v] += 1 # v の値を増やす
    # 個々の値へは dict[' キー'] でアクセス
else:
    dict[v] = 1 # 始めての時は初期値 (1)
```

■頻度カウントの基本的な方法：辞書

```
# 辞書は{}で作る (配列は [])
dict = { } # 空辞書を用意
# 新たな' 物'v がきた場合
if dict.has_key(v): # v がキーとして存在するか？
    dict[v] += 1 # v の値を増やす
    # 個々の値へは dict[' キー'] でアクセス
else:
    dict[v] = 1 # 始めての時は初期値 (1)
```

■ ループが不要なので効率が非常によい

実習：頻度カウントと辞書

- “CountByDict.py” をコピーして残りを完成させなさい
  - “# check if ’substr’ is in count” 部分を完成させる

実習：頻度カウントと辞書

- “CountByDict.py” をコピーして残りを完成させなさい
  - “# check if ’substr’ is in count” 部分を完成させる

```
if dict.has_key(substr):
    dict[substr] = dict[substr] + 1
else:
    dict[substr] = 1
```

配列 v.s. 辞書

- 配列が便利な場合
  - 自然な番号づけがある
  - この番号 (のみ) でデータにアクセスする
  - 番号が連続している (連続してアクセスする)
- 例えば、すべての学生に関するデータ
  - 学籍番号があり、番号の抜けもない
- 逆に、
  - どのデータへアクセスするかわからない
  - 頻繁にいろんなデータへアクセスする

  場合は辞書がよい

全辞書データへアクセス

- 辞書は順番がないので for 文で処理できない

  →すべての “キー”、“値”、“(キー、値) のペア” の配列を返す関数を利用

```
for key in dict.keys(): # キーでアクセス
    print key, dict[key]
```

```
for val in dict.values(): # 値でアクセス
    print val # この場合、キーは不明
```

```
for (key, val) in dict.items(): # (キー、値)
    print key, val
```

実習：辞書データへアクセス

- “CountByDict.py” の “# print all substrings” 部分を完成させなさい
  - すべての要素のキーと値を出力しなさい

実習：辞書データへアクセス

- “CountByDict.py” の “# print all substrings” 部分を完成させなさい
  - すべての要素のキーと値を出力しなさい

```
for (key, val) in dict.items():
    print key, " occurs ", val, " times"
```

辞書に関する注意

- 辞書においてはすべてのキーは異なっている

辞書に関する注意

- 辞書においてはすべてのキーは異なっている
- ただし、値は同じものがあるかもしれない

辞書に関する注意

- 辞書においてはすべてのキーは異なっている
- ただし、値は同じものがあるかもしれない

  例えば、部分文字列をキーとして、その頻度を値とした場合

実習：配列 v.s. 辞書

- 配列と辞書の効率の違いを体験しよう！
  - 授業の Web ページから “CountByList2.py” と “CountByDict2.py” を Z フォルダにダウンロード
    - ほとんど “CountByXXX.py” と同じだが、入力をデータファイルから読み込む点が異なる
  - 同じく “NC_000913.fna”(大腸菌 K12 株の DNA 配列) をダウンロード
  - “CountByList2.py” と “CountByDict2.py” を実行する
    - 引数にデータのファイル名を指定する

比較の結果

- Power Book G4
  - CPU 1GHz (1 秒間に 10 億回の命令)
  - メモリ 1G バイト
- 結果
  - CountByDict2: 約 24.4 秒
  - CountByList2 (ソートなし): 約 99.4 秒
  - CountByList2 (ソートあり): 約 96.6 秒

辞書(2)

- {}でキーと値を“:”(コロン) で区切って生成

```
tel = {'jack': 4098, 'sape': 4139}
# 名前と内線番号データ
```

- アクセスするときはキーを指定する

```
print tel['jack']
del tel['sape']# データの削除
```

辞書のソート

- 辞書は `sort()` 関数を持たない

  `tel.sort()` とはできない
- リストに直してから、ソートする

  dict を適当な辞書変数とする
  - キーでソートする場合

    array = [(key, dict[key]) for key in dict.keys()]

    array.sort()
  - 値でソートする場合

    array = [(dict[key], key) for key in dict.keys()]

    array.sort()

実習：辞書のソート

- “CountByDictSort.py” の “# sort dictionaries” 部分を完成させなさい
  - ソートのための配列の名前は “`array`” とする

```
array = [(dict[key], key) for key in
dict.keys() ]

array.sort()
```

今日の課題

- 頻度カウントするプログラム “CountByDict.py” を改良し、長さ 3 の部分文字列の頻度をカウントするようにしなさい
  - 現状の “CountByDict” では、長さは 2 に固定している
  - 頻度の高い順に出力しなさい

第2回レポート

- 以下を行なうプログラムを作成しなさい
  - 入力：文字列を 1 つと整数 n
  - 出力：この文字列の長さ n の部分文字列
    ただし、頻度の高い順に出力する
- 提出はプログラムと出力結果の例を 1 つ
  - プログラムのファイル名はユーザ名にすること
    例えば “ag10xxxxReport2.py” など
- 締切：7/1(火)