FrontPage / NLP 100 Drill Exercises

もくじ †

↑

言語処理100本ノックについて †

言語処理100本ノックは，言語処理を志す人を対象とした，プログラミングのトレーニング問題集です．　乾・岡崎研の新人研修勉強会の一つであるLearning Programmingで使われています．　このトレーニングは，以下の点に配慮してデザインされています．

- 自然言語処理の研究を進める上で，一度は書いておいた方がよいプログラム
- 統計，機械学習，データベースなどの便利な概念・ツールを体験する
- 実用的で，かつワクワクするようなデータを題材とする
- 研究を進めるうえで重要なプログラミングのルール・作法を身につける
    - モジュール性や組み合わせを考慮しつつ，短くてシンプルなプログラムを書く
    - プログラムの動作を確認（デバッグ）しながらコーディングする
    - 労力を節約する（既存のツール／プログラム／モジュールが使えるときは流用する）
    - 計算資源（メモリ・実行時間）を無駄にしない方法を学ぶ

- Pythonの基礎な事項を一通りカバーする
- 今どきの言語処理系エンジニアが知っておくべき世界の入り口へ誘う

第１セット: テキスト処理の基礎

標準入力からタブ区切り形式のテキスト（address.txt）を読み込み，以下の内容を標準出力に書き出すプログラムを実装せよ．また，ヒントに挙げたツールを用いても，同じ内容が得られることを確認せよ．

- (1) 行数をカウントしたもの．確認にはwcコマンドを用いよ．
- (2) タブ１文字につきスペース１文字に置換したもの．確認にはsedコマンド，trコマンド，もしくはexpandコマンドを用いよ．
- (3) 各行の１列目だけを抜き出したものをcol1.txtに，２列目だけを抜き出したものをcol2.txtとしてファイルに保存せよ．確認にはcutコマンドを用いよ．
- (4) (3)で作ったcol1.txtとcol2.txtを結合し，元のタブ区切りテキストを復元したもの．確認にはpasteコマンドを用いよ．
- (5) 自然数Nをコマンドライン引数にとり，入力のうち先頭のN行だけ．確認にはheadコマンドを用いよ．
- (6) 自然数Nをコマンドライン引数にとり，入力のうち末尾のN行だけ．確認にはtailコマンドを用いよ．
- (7) １コラム目の文字列の異なり数（種類数）．確認にはcut, sort, uniq, wcコマンドを用いよ．
- (8) 各行を２コラム目の辞書順にソートしたもの（注意: 各行の内容は変更せずに並び替えよ）．確認にはsortコマンドを用いよ（この問題は結果が合わなくてもよい）．
- (9) 各行を２コラム目，１コラム目の優先順位で辞書の逆順ソートしたもの（注意: 各行の内容は変更せずに並び替えよ）．確認にはsortコマンドを用いよ（この問題は結果が合わなくてもよい）．
- (10) 各行の２コラム目の文字列の出現頻度を求め，出現頻度の高い順に並べよ．ただし，(3)で作成したプログラムの出力（col2.txt）を読み込むプログラムとして実装せよ．確認にはcut, uniq, sortコマンドを用いよ．

第２セット: 正規表現・日本語の扱い

標準入力からテキスト（tweets.txt.gz）を読み込み，以下の処理を行うプログラムを実装せよ．

- (11) 「拡散希望」という文字列を含むツイートを抽出せよ．
- (12) 「なう」という文字列で終わるツイートを抽出せよ．
- (13) 非公式RTのツイートの中で，RT先へのコメント部分のみを抽出せよ．
- (14) ツイッターのユーザー名（@で始まる文字列）を抽出せよ．
- (15) ツイッターのユーザー名（例えば@xxxxxxx）を，そのユーザーのページへのリンク（@xxxxxxxで囲まれたHTML断片）に置換せよ．

- (16) 括弧表現のうち，括弧の内側がアルファベット大文字の文字列，括弧の左側が漢字の文字列のものを抽出せよ．このとき，括弧の左側の表現と括弧の内側の表現をタブ区切り形式で出力せよ．
- (17) 人名らしき表現にマッチする正規表現を各自で設計し，抽出せよ．（例えば「さん」「氏」「ちゃん」などの接尾辞に着目するとよい）
- (18) 仙台市の住所らしき表現にマッチする正規表現を各自で設計し，抽出せよ．
- (19) 郵便番号・県名・市町村名の３要素を組で（まとめて）抽出せよ．
- (20) ツイートから絵文字らしき文字列を抽出せよ．

↑

第３セット: 文分割・トークン化（英語） †

英語のテキストmedline.txtを，１行１文形式のテキストmedline.txt.sentに変換するプログラムを，次のように実装せよ．

- (21) 標準入力から英語のテキストを読み込み，ピリオドを文の区切りと見なし，１行１文の形式で標準出力に書き出せ．
- (22) (21)のプログラムは破棄して，標準入力から英語のテキストを読み込み，ピリオド→スペース→大文字を文の区切りと見なし，１行１文の形式で標準出力に書き出せ．

１行１文形式のテキストmedline.txt.sentを，１行１単語形式のテキストmedline.txt.sent.tokに変換するプログラムを次のように実装せよ．

- (23) (22)の出力を標準入力から１行（１文）を読み込む毎に，スペースで単語列に分割し，１行１単語形式で標準出力に書き出せ．文の終端を表すため，文が終わる毎に空行を出力せよ．
- (24) (23)のプログラムを修正し，各トークンの末尾が記号で終わる場合は，その記号を別のトークンとして分離せよ．
- (25) (24)の出力を標準入力から１行（１単語）を読み込む毎に，その単語を小文字に変換した文字列を各行の最終列にタブ区切り形式で追加し，標準出力に書き出せ．

１行１単語形式のテキストmedline.txt.sent.tokを用いて，以下の分析を行え．

- (26) -nessと-lyの両方の派生語尾をとる単語をすべて抜き出せ．
- (27) (10)のプログラムを呼び出すことで，頻度の高い英単語トップ100（単語と頻度がソートされたもの）を作成せよ．
- (28) 各単語から文字バイグラムを抽出するプログラムを実装せよ．また，(27)と同様の方法で，頻度の高い文字バイグラムトップ100（バイグラムと頻度がソートされたもの）を作成せよ．

ステミング（単語を語幹に変換し単語の語尾変化を取り除く処理）に関する以下のプログラムを実装せよ．

- (29) stemming 1.0（Porterのステマー）を各自の環境にインストールし，正しくインストールされているか確認せよ．

- (30) (25)の出力を標準入力から読み込み，stemming.porter2を用いて語幹（ステム）を最終列に追加し，medline.txt.sent.tok.stemというファイルに保存せよ．

↑

第４セット（前半）：辞書引き †

inflection.table.txt には，英語の語彙（単語）に関する情報が収録されている． このファイルは"|"区切り形式で，第１列に単語（活用あり），第２列に品詞，第４列に活用形，第７列に基本形が格納されている．

- (31) このファイルを読み込み，単語をキーとして，品詞，活用形，基本形のタプルのリストを値とするマッピング型に格納せよ．プログラムの動作を確認するため，標準入力から読み込んだ単語の語彙項目を閲覧するプログラムを実装せよ．
- (32) (31)で読み込んだマッピング型オブジェクトを，marshalモジュールを使ってファイルに書き出せ．書き出したファイルを今後「辞書」と呼ぶ．
- (33) (32)で作成した辞書を読み込み，１行１単語形式（例えばmedline.txt.sent.tok.stem）で標準入力から読み込んだ単語に対して，辞書引き結果を付与するプログラムを実装せよ．
- (34) 辞書引きを行った結果，辞書に載っていなかった語のみを出力せよ．
- (35) 辞書引きを行った結果，３つ以上のエントリが見つかったもののみを出力せよ．

↑

第４セット（後半）：Nグラム言語モデル †

- (36) １行１単語形式（medline.txt.sent.tok）を読み込み，単語の連接を出力するプログラムを実装せよ．ただし，出力形式は"(現在の単語)\t(次の単語)"とせよ．
- (37) (36)の出力を読み込み，単語の連接の頻度を求めよ．ただし，出力形式は"(連接の頻度)\t(現在の単語)\t(次の単語)"とせよ．
- (38) (37)の出力を読み込み，ある単語wに続く単語zの条件付き確率P(z|w)を求めよ．ただし，出力形式は"(条件付き確率)\t(現在の単語)\t(次の単語)"とせよ．
- (39) (38)の出力を読み込み，単語の連接(w, z)をキーとして，その条件付き確率P(z|w)を値とするハッシュデータベースを構築せよ．ハッシュデータベースの構築には，Kyoto CabinetのPythonモジュールを用いよ．ただし，Kyoto CabinetのPythonモジュールはすでに導入済みであるので，各自でインストール作業を行わなくてもよい．
- (40) (39)で構築したデータベースを読み込み，標準入力から読み込んだ文の生起確率を計算せよ．入力された文が単語列(w1, w2, ..., wN)で構成されるとき，生起確率はP(w2|w1)P(w3|w2)...P(wN|wN-1)と求めればよい．試しに，"this paper is organized as follows"と"is this paper organized as follows"の生起確率を計算せよ．

↑

第５セット: 形態素解析／グラフ描画 †

日本語の文章japanese.txtに対して，以下の処理を行え．

- (41) 日本語の文章をMeCabで形態素解析し，その結果を読み込むプログラムを実装せよ．ただし，各形態素は表層形（surface），基本形（base），品詞（pos），品詞細分類1（pos1）をキーとするマッピング型に格納し，1文は形態素（マッピング型）のリストとして表現せよ．
- (42) 文章中に出現するすべての動詞を抜き出せ．
- (43) 文章中に出現するすべての動詞の基本形を抜き出せ．
- (44) サ変名詞をすべて抜き出せ．
- (45) 文章中の「AのB」という表現（AとBは名詞の1形態素）をすべて抜き出せ．
- (46) 文章中のすべての名詞の連接（1形態素以上）を抜き出せ．
- (47) (42)から(46)までの処理を1つのプログラムに統合し，処理内容をコマンドライン引数でOn/Offできるようにせよ．コマンドライン引数の処理には，optparseモジュールを用い，オプションには適当な名前（例えば(42)は--verbなど）とドキュメント（-hを引数にすることで表示される）を書け．
- (48) 標準入力から読み込んだ各行の文字列の頻度を求めるプログラムを書き，(47)のプログラムと組み合わせることによって，文章中に出現する各動詞の出現頻度を求めよ．さらに，出現頻度の高い順に動詞を並べよ．
- (49) (48)の出力を利用して，文字列の頻度を横軸，その文字列の異なり数（種類数）を縦軸として，ヒストグラムをプロットせよ．なお，プロットにはgnuplotやmatplotlibを用い，グラフを画像ファイルとして保存せよ．
- (50) (48)の出力を利用して，文字列の出現頻度の順位（高い順）を横軸，その出現頻度を縦軸として，プロットせよ．

↑

第６セット: 係り受け解析／クラス †

日本語の文章japanese.txtに対して，以下の処理を行え．

- (51) 日本語の文章をCaboChaで係り受け解析し，ラティス形式（-f1オプション）の解析結果を得よ．
- (52) 形態素を表すクラスMorphを実装せよ．このクラスは表層形（surface），基本形（base），品詞（pos），品詞細分類1（pos1）をメンバ変数に持つこととする．さらに，(51)の解析結果を1文毎に読み込み，1文をMorphオブジェクトのリストとして表現し，適当に表示するプログラムを実装せよ．
- (53) 文節を表すクラスChunkを実装せよ．このクラスは形態素のリスト（morphs），係り先文節インデックス番号（dst），係り元文節インデックス番号のリスト（srcs）をメンバ変数に持つこととする．さらに，(51)の解析結果を1文毎に読み込み，1文をChunkオブジェクトのリストとして表現し，適当に表示するプログラムを実装せよ．
- (54) 以降のプログラムを実装しやすくするため，(53)のプログラムをモジュール化せよ．
- (55) 係り元の文節と係り先の文節をタブ区切り形式ですべて抽出せよ．ただし，句読点などの記号は出力しないようにせよ．

- (56) 名詞を含む文節が，動詞を含む文節に係るとき，これらをタブ区切り形式で抽出せよ．
- (57) (56)を修正し，非自立語は出力に含めないようにせよ．
- (58) 係り元が２つ以上ある文節に対し，その文節とすべての係り元を表示せよ．
- (59) 各文に含まれるすべての体言（名詞を含む文節）の組み合わせに対し，その文節間の表現（係り受けパス）を出力せよ．
- (60) (57)の出力を「係り元」→「係り先」の有向グラフとみなし，Graphvizを使ってグラフを描画せよ．すなわち，(57)の出力をGraphvizの入力フォーマットであるDOT形式に変換するプログラムを実装すればよい．グラフを描画するときは「neato -Tsvg」コマンドを用い，SVG形式に書き出すとよい．

第７セット: 文脈類似度，クラスタリング

- (61) コーパスディレクトリ中の各ファイルに対して，cabochaを適用し，適当なディレクトリに係り受け解析の結果を格納せよ．ただし，各ファイルの文字コードがUTF-16LEであること，文区切りを自前で行う必要があることに注意せよ．

```
コーパスディレクトリ                作業用ディレクトリ
+ OW1X_00000.txt        →      + OW1X_00000.txt
+ OW1X_00001.txt        →      + OW1X_00001.txt
+ OW1X_00003.txt        →      + OW1X_00003.txt
...
```

- (62) 61で作成した各ファイルから，名詞句（文節中の名詞の連接）を抜き出して，個別のファイルに格納せよ．

```
作業用ディレクトリ                作業用ディレクトリ
+ OW1X_00000.txt        →      + OW1X_00000.txt.n
+ OW1X_00001.txt        →      + OW1X_00001.txt.n
+ OW1X_00003.txt        →      + OW1X_00003.txt.n
...
```

- (63) 62の結果を用い，それぞれの名詞句のTF*IDF値を計算し，"(名詞句)\t(TF*IDF値)\t(TF値)\t(DF値)"の形式で出力せよ．ある名詞句wがあるとき，freq(w)をコーパス全体での名詞句wの出現頻度，df(w)を名詞句wが出現するファイルの数，Nを総ファイル数とし，TF*IDF値は freq(w) * log(N / df(w)) として計算せよ．

```
平成    7561.083955     7943.000000     579
昭和    4894.864348     7604.000000     788
企業    3898.705525     2658.000000     346
```

- (64) 63の結果を用い，TF*IDF値が高い名詞句トップ100のリストを作成せよ．

このとき，数字を含む表現はトップ100のリストから除外せよ．

- (65) 64のリストに含まれる名詞句に対し，その名詞句に係る文節・その名詞句が係る文節の単語（周辺単語と呼ぶ）を出力するプログラムを実装せよ．周辺単語は名詞，動詞，形容詞の基本形とせよ．出力形式は，"(名詞句)\t(方向) (周辺単語)"とし，名詞句に係る文節では「方向」を"<-"とし，名詞句が係る文節では「方向」を"->"とせよ．以降，「方向」と「周辺単語」を組み合わせたものを，名詞句の「文脈」と呼ぶ．

```
効果　　-> 現われる
効果　　-> やすい
効果　　<- 公共
効果　　<- 投資
...
```

- (66) 名詞句xの文脈ベクトルを求めよ．ただし，出力形式は"(名詞句i)\t(文脈i_1):(値i_1)\t...(文脈i_m):(値i_m)\n"とせよ（ただしmは名詞句iの文脈の種類数）．文脈ベクトルは長さが1になるように正規化しておくとよい．

```
我が国　-> する:0.657597　　-> いる:0.487923
-> れる:0.146816　　<- 年:0.139783　-> 貿易:0.139783
...
```

- (67) 66の出力の各行を名詞句の（疎）ベクトルとみなし，「日本」と「我が国」のベクトル間の内積（コサイン類似度）を求めよ．
- (68) すべての名詞句のペアに対し，文脈ベクトルの内積が0.6以上のペアをすべて抜きだし，内積値が大きい順に並べよ．このとき，出力形式は"(内積値)\t(名詞句1)\t(名詞句2)\n"とせよ．

```
0.988265　　図　　表
0.987233　　増加　　減少
0.966998　　上昇　　低下
```

- (69) 68の結果を名詞句をノードとする無向グラフとみなし，Graphvizを使ってグラフを描画せよ．すなわち，68の出力をGraphvizの入力フォーマットであるDOT形式に変換するプログラムを実装すればよい．グラフを描画するときは「neato -Tsvg」コマンドを用い，SVG形式に書き出すとよい．
- (70) 68の結果を用い，最長距離法（furthest neighbor method; complete link method）で名詞句のクラスタリングを行い，クラスタを抽出せよ．

↑

第８セット: 機械学習／分類器 †

実際に書かれている大量の英語のテキストから，冠詞のタイプ（不定冠詞／定冠詞／無冠詞）を学習し，冠詞の誤りの訂正を行うシステムを以下の手順で構築せよ．

- (71) 5つのファイル（英語のテキスト）にGENIA taggerを適用せよ．GENIA taggerは1文1行形式の入力を受け取るので，22のプログラムを再利用せよ．また，入力ファイルはgzipで圧縮されていることに注意せよ．

```
コーパスディレクトリ        作業用ディレクトリ
+ data00.txt.gz        →      + data00.genia.gz
+ data01.txt.gz        →      + data01.genia.gz
+ data02.txt.gz        →      + data02.genia.gz
+ data03.txt.gz        →      + data03.genia.gz
+ data04.txt.gz        →      + data04.genia.gz
```

- (72) 71の解析結果を読み込むモジュールを実装せよ．各トークンは表層形（w），レンマ（lem），品詞（pos），チャンクタグ（chunk）をキーとするマッピング型に格納し，1文はトークン（マッピング型）のリストとして表現せよ（固有表現情報は読み込まなくてもよい）．
- (73) 72のモジュールを用い，名詞句（NP）を抽出するプログラムを実装せよ．ただし，ひとつの名詞句の出現に対し，"# (名詞句)\n"という形式で出力せよ．

```
# the literature
# It
# a few exceptions
# the Mm
# peptides
...
```

- (74) 73で作成したプログラムを改変し，名詞句の冠詞が不定冠詞の場合は"A"，定冠詞の場合は"THE"，無冠詞の場合は"NONE"と表示するプログラムを実装せよ．ただし，73の表示内容に追加し，"# (名詞句)\n(冠詞タイプ)\n"という出力形式にせよ．

```
# the literature
THE
# It
NONE
# a few exceptions
A
# the Mm
THE
# peptides
NONE
...
```

- (75) 74で作成したプログラムを改変し，以下の11種類の素性を抽出・表示するプログラムを実装せよ．ただし，これらの素性はタブ区切り形式で"# (名詞

句)\n(冠詞タイプ)\t(タブ区切り形式の素性)\n"という形式で出力せよ．例えば，"A major challenge is [the limited number] of NOE restraints ..."の[]の部分の名詞句に対して，次の出力を得る（ただし"\"は実際にはここで改行しないことを表す）

```
# the limited number
THE     w[-1]=be        fw=limited      fpos=JJ w[0]=l
imited number     fw|fpos=limited|JJ      \
hw=number       hw|hpos=number|NN       pos[+1]=IN
  hpos=NN pos[-1]=VBZ     w[+1]=of
```

各素性の意味は次の通りである．

- hw=(末尾の単語)
- hpos=(末尾の品詞)
- hw|hpos=(末尾の単語)|(末尾の品詞)
- fw=(先頭の単語)
- fpos=(先頭の品詞)
- fw|fpos=(先頭の品詞)|(先頭の単語)
- w[0]=(名詞句の単語列)
- w[-1]=(名詞句の 1 語前の単語)
- pos[-1]=(名詞句の 1 語前の品詞)
- w[1]=(名詞句の 1 語後の単語)
- pos[1]=(名詞句の 1 語後の品詞)

- (76) 75で作成したプログラムをdata00.genia.gz〜data04.genia.gzに適用し，学習データdata00.f〜data04.fを作成せよ．

```
作業用ディレクトリ              作業用ディレクトリ
+ data00.genia.gz     →     + data00.f
+ data01.genia.gz     →     + data01.f
+ data02.genia.gz     →     + data02.f
+ data03.genia.gz     →     + data03.f
+ data04.genia.gz     →     + data04.f
```

- (77) data01.f〜data04.fを学習データとし，冠詞のタイプを予測するモデルを，Classiasで学習せよ．モデルのファイル名はmodel.txtとせよ．
- (78) 学習したモデルmodel.txtを使い，data00.fの素性から冠詞のタイプを予測せよ．このとき，予測された冠詞のタイプと，正解の冠詞のタイプを並べて出力せよ．
- (79) data00.f〜data04.fを用いて５分割交差検定を行い，冠詞予測の精度，適合率，再現率，F1スコアを求めよ．
- (80) 入力された英語の文に対して，各名詞句の冠詞のタイプを推定し，誤りがあれば訂正を行うシステムを構築せよ．

↑

第９セット: データベース †

tweets100.tsv.gzはタブ区切り形式でツイートを格納している.

#	名前	型	内容
0	url	string	ツイートのURL
1	date	date/time	ツイートの投稿日時
2	user	string	ユーザID
3	nickname	string	ユーザ名
4	body	string	ツイートの内容
5	rt_url	string	RT先のURL
6	reply_url	string	返信先のURL
7	qt_url	string	QT先のURL
8	freq_rted	int	ツイートがRTされた回数
9	freq_replied	int	ツイートにreplyされた回数
10	freq_qted	int	ツイートが引用された回数

以下の手順により，MongoDBを用いたツイートデータベースを構築せよ.

- (81) このデータをMongoDBに格納せよ．ただし，MongoDBのデータベース名は"nlp100_{ユーザ名}"，コレクション（テーブル）名は"tweets"とし，ドキュメント（レコード）のフィールドには適当な名前を付けよ.
- (82) MongoDBの対話型シェルを用い，url, date, user, rt_url, reply_url, qt_urlに対してインデックスを構築せよ.
- (83) MongoDBの対話型シェルを用い，特定のURLのツイートを検索せよ.
- (84) MongoDBの対話型シェルを用い，特定のURLに返信しているツイートの本文をすべてを検索せよ.
- (85) MongoDBの対話型シェルを用い，特定のユーザのツイートをRT数の多い順に10件まで検索せよ.
- (86) 83を実現するプログラムを実装せよ．なお，MongoDBをPythonから利用するときには，pymongoモジュールを用いる.
- (87) 85を実現するプログラムを実装せよ.
- (88) 81-82の処理に加え，ツイート本文の文字バイグラムをドキュメントに追加し，このフィールドでインデックスを構築せよ.
- (89) 「東北」「北大」「大学」の文字バイグラムを本文に含むツイートを検索せよ.

- (90) 引数に渡された文字列を含むツイートを，RT数の多い順に表示するプログラムを実装せよ．

第１０セット: 構造化データ（XML）の処理／CGIによるデモシステム

Stanford Core NLPのXML形式の出力から，以下の情報を抽出せよ．ただし，XML形式の解析にはlxmlモジュールを用いよ．

- (91) ２番目の文の５番目のトークンの単語
- (92) １番目の文のlemmaを並べたもの．
- (93) １番目の文の４番目のトークンの，係り受け構造上での子（dependent）の単語

Wikipediaのダンプ（XML形式）を解析し，以下の情報を抽出せよ．ただし，XML形式の解析にはxml.saxモジュールを用いよ．

- (94) 記事のタイトル．
- (95) 記事にリダイレクト（転送）先が設定されているときは，記事のタイトルとリダイレクト先．
- (96) 記事のタイトルとカテゴリ情報．複数のカテゴリが設定されているときは，全て抜き出せ．
- (97) 記事のタイトルと対応する英語のページタイトル（言語間リンク）．

ウェブ上のデモシステムを，以下のように構築せよ．

- (98) 現在時刻を表示するCGIを，自分のウェブサイト上で公開せよ．
- (99) 90で作ったプログラムをCGIに改変し，「q」というパラメータで検索クエリを受け取り，検索結果をHTMLで出力せよ．
- (100) 99で作ったプログラムを改変し，ブラウザ上からツイートを検索できるようにせよ．

データについて

乾・岡崎研の内部の人は，以下のディレクトリからデータにアクセスすることができます．

```
/home/work/data/nlp100
```

このトレーニングで用いられるデータセットは，以下の方法で構築しました． これらのデータセットは，著作権などの問題から再配布しておりません．

address.txt

日本郵便の郵便番号データの全データ（KEN_ALL.CSV）にaddress.pyを適用し，都道府県市町村名を取り出したもの．

tweets.txt.gz
日本語のツイートをクロールしたもの（非公開）

medline.txt
MEDLINEの論文抄録をスクリプトで抽出したもの．

inflection.table.txt
UMLS SPECIALIST Lexiconで配布されているもの．

japanese.txt
Wikipediaの中で長いページから適当に本文を抽出したもの

↑

参考情報 †

↑

Python †

- Pythonの日本語ドキュメント: 初めての人はチュートリアルをやると良いと思います
- PEP 8 -- Style Guide for Python Code
- Google Python スタイルガイド — PyGuide - 1.0

↑

UNIXコマンド †

- Linux標準教科書
- Linuxコマンドでテキストデータを自在に操る

↑

生い立ち †

- 言語処理研究を目指す学生が100問のプログラミング課題を解くというコンセプトは，名古屋大学の佐藤・駒谷研で昔から行われていたものです．
- 「100本ノック」の問題は岡崎直観が作りました．
- 「100本ノック」の名付け親は杉浦純さんです．

Last-modified: 2014-01-15 (Wed) 01:54:28 (218d)