


プログラミング言語 第一回



担当: 篠沢 佳久
栗原 聡

平成31年度: 春学期



管理工学科におけるプログラミングの講義

- 本講義(2年春) 選択必修
 - Pythonを対象として, プログラミングの基礎を中心に
- ソフトウェア工学(2年秋, 飯島先生) 選択必修
- 管理工学実験・演習3(3年春, COM系教員)
 - COM実験と呼ばれています 必修



クラス分け



クラス分け①

- プログラミング言語は二つの教室で同時に行います
 - 703教室(50人収容)
 - 704教室(100人収容)
- どちらの教室も同じ内容の講義をします



クラス分け②

■ 703

- K組 61809800までの学籍番号の学生
- L組 61812700までの学籍番号の学生

■ 704

- K組 61809800以降の学籍番号の学生
- L組 61812700以降の学籍番号の学生
- 管理工学科2年生以外の学生



講義のガイダンス

講義の目的, 進め方



この講義の目指すもの Part1

- プログラミングの基礎を理解
 - プログラミングの基礎知識を中心に学ぶ
 - プログラムとは
 - プログラムの実行とは
 - 命令とデータ
 - 判断と分岐
 - プログラミングの構造と実行制御



この講義の目指すもの Part2

- プログラミングという行為
 - 書く, テストする, 使う
- プログラミングが一人でできることを目的
 - アルゴリズム
 - データ構造
- プログラミング言語とは
- プログラミングの基本をプログラム言語Pythonを通して学ぶ



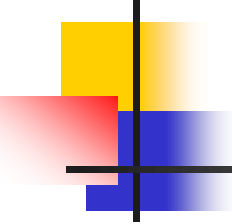
この講義の目指すもの Part3

- プログラム言語Pythonで基本的なプログラムが作れるように
 - 基本的な演算
 - 制御構造
 - 条件式
 - 繰り返し
 - 配列(リスト)
 - 標準入出力, ファイル入出力



Pythonとは何か？

- Python:オブジェクト指向型スクリプト言語
 - スクリプト言語: 動作内容を, 台本 (Script) のように記述するための簡易的なプログラミング言語の総称
 - かなり簡単に (周辺環境が) インストールできる
 - 皆さんのコンピュータでも容易に実習できる
 - かなり簡単にプログラムができる
 - 初心者にも容易に学習できる
 - プログラムが読みやすい
 - ライブラリが豊富
 - 近年の人工知能, 機械学習ブームで飛躍的に用いられるようになった



Pythonのバージョンの注意

- 現在用いられているPythonにはversion.2とversion.3があり, 最新版はversion.3です
- 二つのversionではプログラムの書き方が少々変わります
- 本講義ではversion.3を用います
- さらに, さまざまなライブラリがインストール済みのAnaconda版pythonを用います



この講義では

- 演習をできる限り行います
 - そのためには、Pythonプログラムを実行するシステムとして対話型シェル（インタラクティブシェル，REPL（Read-Eval-Print-Loop）とも呼ばれます）を最初の講義では用います
 - 対話型シェルは，一行ずつPythonプログラムを入力→実行できるので，ちょっと実習をするには，適しているのです



この講義の先には

- もう少し先（もっと先？）に行くと,
 - シミュレーション
 - データ解析, データマイニング
 - 日本語処理
 - データベース処理
 - デスクトップアプリケーション
 - Webアプリケーション
 - 組み込みアプリケーション
 - 機械学習, 人工知能



内容に関する注意

- 基本的(初歩的)なことに注力する
- Python特有の書き方はできるだけ省くようにします
 - 他の言語でもプログラムがすぐに書けるように
- ただし, ところどころ細かい話もする
 - 少し深いことを知りたい方への追加
 - 疑問に対する答えとして
 - 初級者は無視をしてよい



進め方

- (繰り返しになりますが)Pythonを使う
 - 実習を多く行ないます
- ある事例(課題)を考える
 - ある動作をする「プログラム」
 - もちろん, 簡単版



方針

- 多くのサンプルプログラムを用意します
 - 講義では全て話すことができません
 - 復習もして下さい(レポートがあります)
- 練習問題を多く行ないます



実習について

- この講義では理解を深めるために実習を交えて行ないます
- 教室・・・日吉ITC 地下一階
 - 703(50人収容)
 - 704(100人収容)
 - どちらも同じ講義内容



成績について①

- 成績のつけかた
 - 講義以外の時間にレポートを作成
 - 3回を予定
 - 講義の最終回(7/22)に最終課題を行ないます
 - 必ず出席して下さい
 - 講義中の演習問題(平常点)
 - 平常点+レポート(3回)+最終課題の成績から判定



成績について②

- Pythonでプログラムが書ける（自信のある）人は，授業に出席しなくてもレポートさえ出せば単位がとれる
 - 予め申告することが条件
 - ただし3回のレポートは必ず提出して下さい
 - また最終回（7/22）は出席し，最終課題は必ず受けて下さい



講義に関する情報

- 講義資料のURL

- <http://lecture.comp.ae.keio.ac.jp/program2019/>

- 教員, TAへの質問

- 電子メール
- 直接質問(矢上なのでアポイントが必要です)



プログラムとは

プログラミングの必要性
プログラムとプログラム言語



なぜプログラミング？

- 他の講義・実験・演習・卒論に必要
- 管理工学の基盤的な技術(道具)
- 今後, 必要な技術
- 今後, 知っておくべき技術
- 論理的思考力の訓練



プログラムとは①

- 日常使う「プログラム」はどのような意味か？
 - コンサートに行くと...
- すなわち，手順・動作を記した書類
 - 書類といっても，紙に書かれているわけではない



プログラムとは②

- コンピュータにおいて用いる「プログラム」とは？
- コンピュータが行う動作を
 - 事細かに
 - 逐一記述したもの

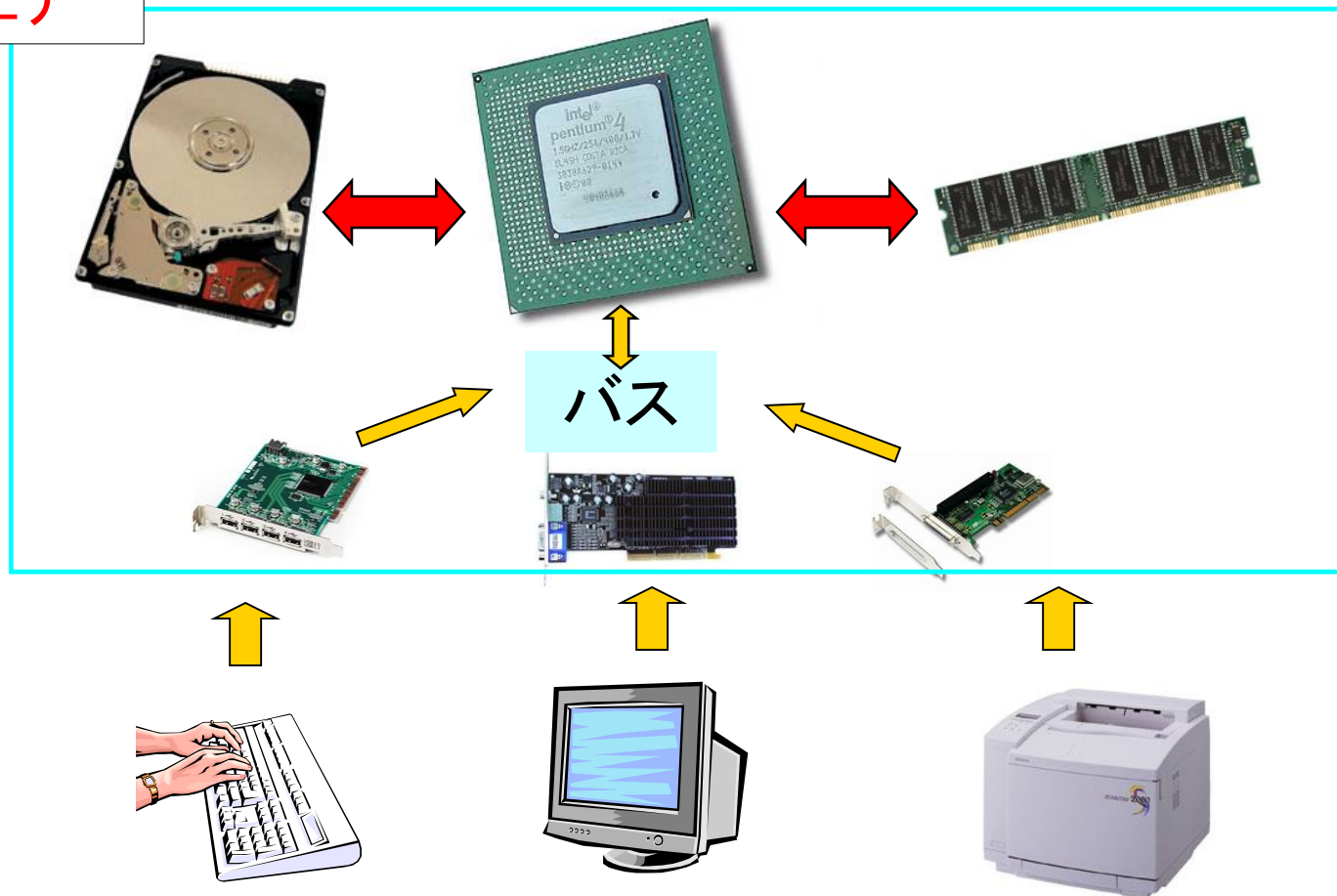


プログラムとは③

- コンピュータの「記憶装置」に蓄えられている
 - メモリ: 普通はコンピュータの中に隠されている
 - 内容を持ち運びたいときに, USBメモリとかDVD-Rとかいったものにコピーする
- すなわち, プログラムは「ソフトウェア(軟件)」
 - ハードウェア(硬件)ではない
 - つまり, 触って感じる物ではない

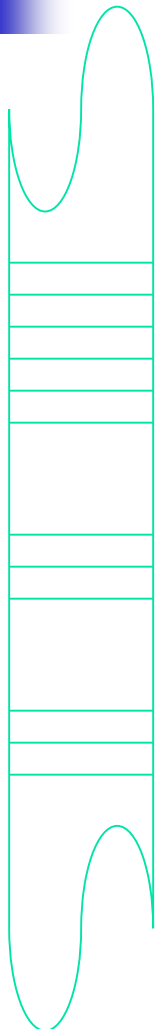
コンピュータとは

ハードウェア





プログラム プログラム プログラム



- Operating System (OS) : プログラムの実行の制御や, ハードウェアの制御と管理など, コンピュータを安全にそして効率良く働かせるための基本ソフトウェア
 - 例: Windows 7/8/10, UNIX, Linux, FreeBSD, Solaris, Tron, Mac OS
- アプリケーション
 - 例: 表計算, 文書作成, プレゼンテーション作成, ブラウザ
- ユーザ作成プログラム ← **本講義はここです**
 - 例: 「こんにちは」プログラム



プログラムは何語で書くか

- 「書類」だから、記述する言語が必要
 - 言語: 意味のある文字列
- 日本語や英語がだめなことは、勿論
 - なぜか？
- コンピュータが分かる言語？
 - 比喩が過ぎる. コンピュータは意味は分からないから
 - コンピュータが、文字列から自分がすべき動作に変換できればよい
- コンピュータ用の言語を作ればよい(プログラム言語)



それをプログラム言語という

- コンピュータは、メモリのどこかに書いてある「命令」を自分の動作に変換すればよい
 - この「命令」の構成規則が言語
 - この変換規則は言語ではない
 - コンピュータ(機械)にとっては言語(かな?)なので、機械語といったりする
 - この変換規則の例:
 - 01100 → 出力電圧を5Vに
某神経細胞on → 右手親指曲る (人間の脳)



プログラミング言語とは

- 人間の思いをコンピュータに伝える言葉
 - といったって相手はコンピュータですから
 - 人間の言葉より、機械の言葉にずっと近い.
ということは
 - 硬い. すなわち, 規則にやかましい
 - 手書き文字ではない. すなわち, キーボード入力



どんなものがあるか？

- 高級 (high-level) 言語
 - 実行方法による分類
 - コンパイラ言語
 - Ex. C, Java, Fortran, Cobol
 - インタプリタ言語
 - Ex. Python, Ruby
 - 概念による分類
 - 命令型言語
 - Ex. C, Java, Fortran, Cobol, Python, Ruby
 - 関数型言語
 - EX. Lisp
- アセンブリ言語・機械語



Pythonの長所・短所

■ 長所

- 始めやすい
- インストールが簡単
- プログラムもその実行も簡単
- 一行から始められる
- ライブラリが豊富
- （実は隠れた長所がたくさんあります。急成長中）

■ 短所

- 「作法」「行儀」が学びにくい
- 個性が非常に強い



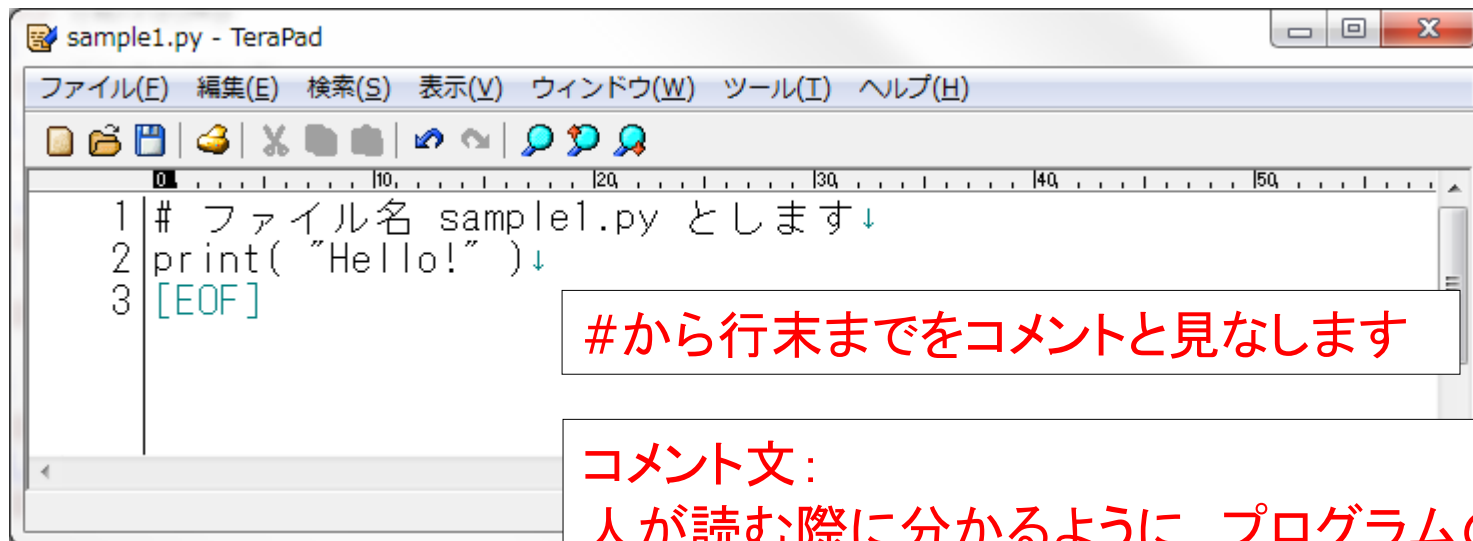
プログラミング実習

Pythonプログラムの作成と実行
(Anaconda Prompt上での実行)

"Hello!"を表示するPythonプログラム

作成するプログラム

```
# ファイル名 sample1.py とします  
print( "Hello!" )
```



```
1 # ファイル名 sample1.py とします↓  
2 print( "Hello!" )↓  
3 [EOF]
```

#から行末までをコメントと見なします

コメント文:

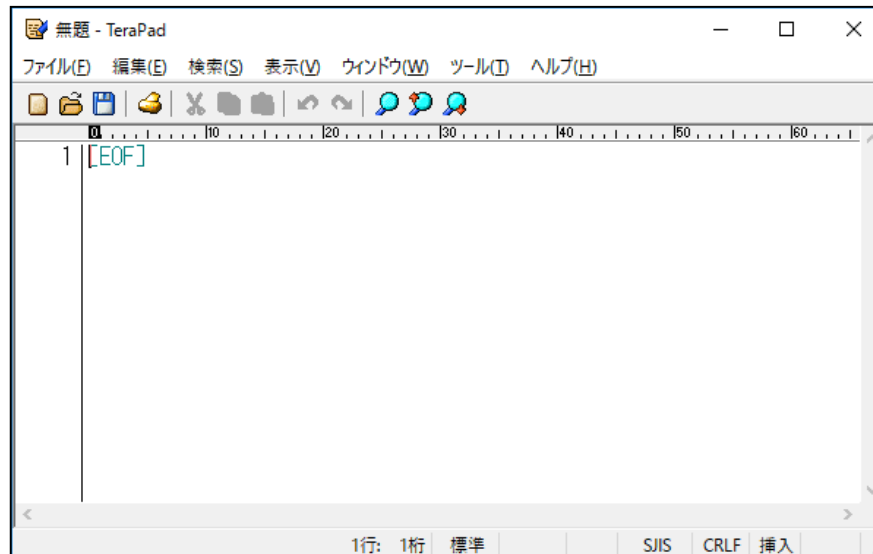
人が読む際に分かるように、プログラムの説明を書きます

"Hello!"を表示するPythonプログラム

プログラム

→ テキストエディタで記述する

TeraPad



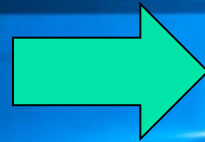
まずは、やってみよう

- 皆さんの「ドキュメント」は、日吉のPCでは、Zドライブになっています
- そこに、Python という名のフォルダを作ってください

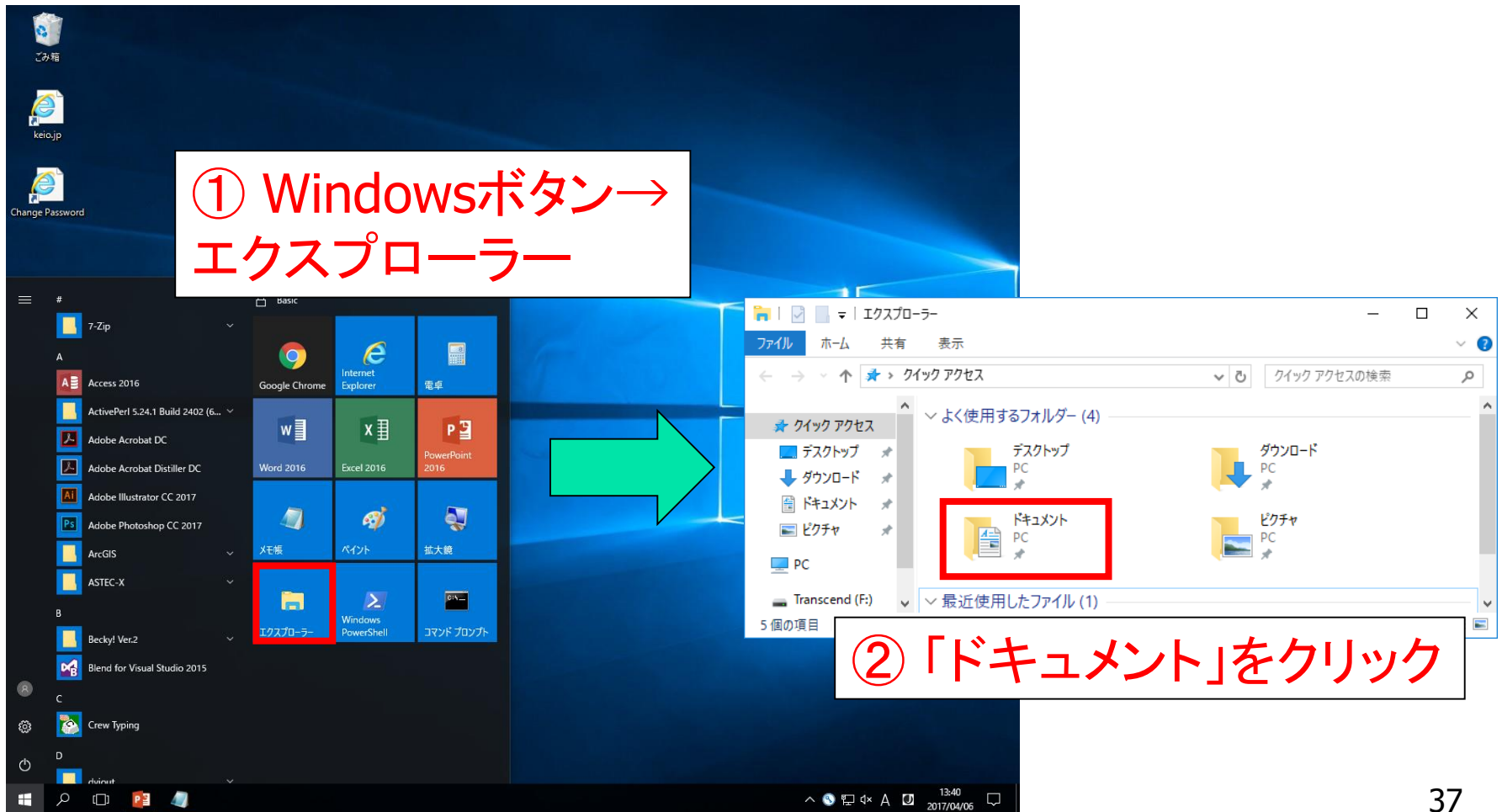


日吉ITCの場合 (OSはWindows10)

① Windowsボタン→
エクスプローラー

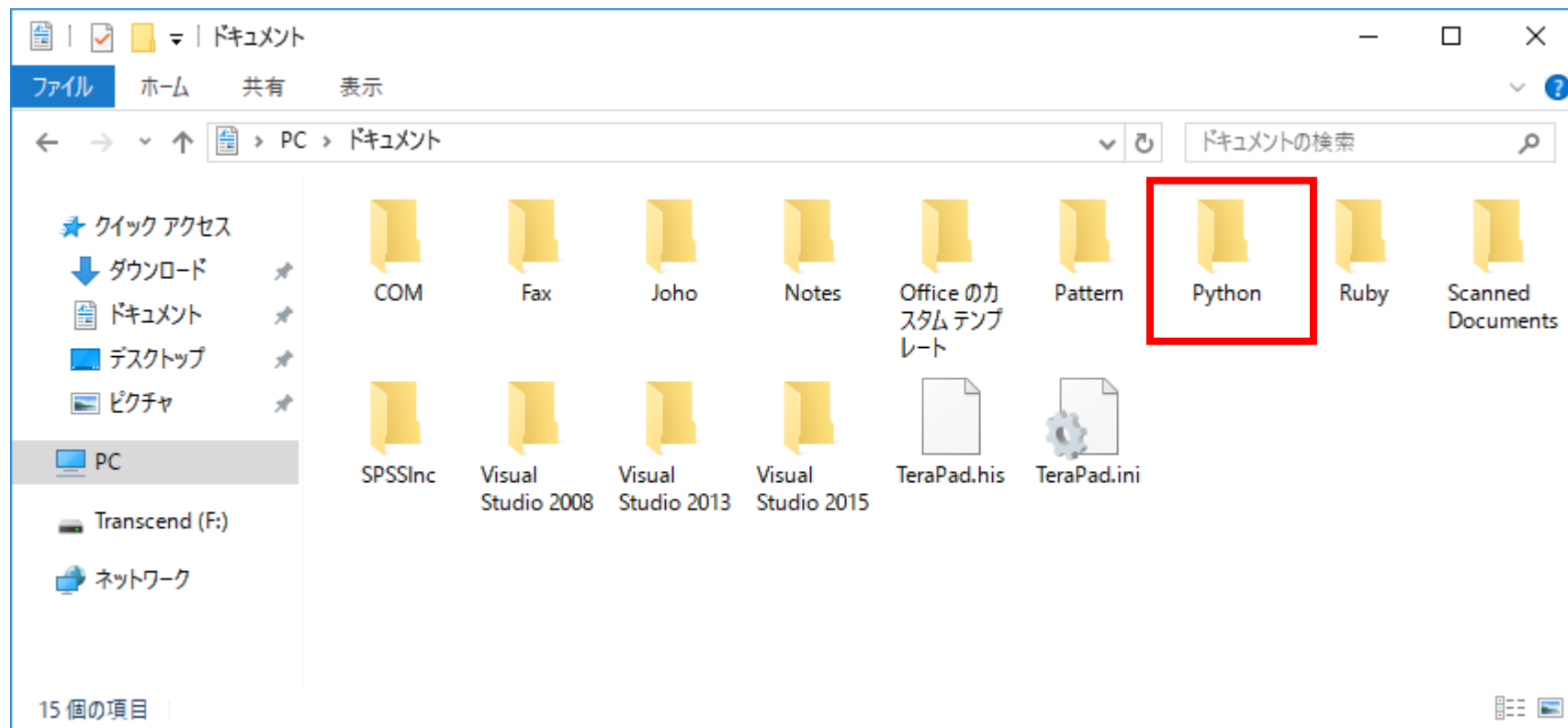


② 「ドキュメント」をクリック




日吉ITCの場合 (OSはWindows10)

③「Python」という名前のフォルダーを作成して下さい






ディレクトリ/フォルダとは

- ハードディスクやCD-ROMなどの記憶装置において、ファイルを分類・整理するための保管場所
- UNIXやMS-DOSではディレクトリといい、MacintoshやWindowsではフォルダという
- Windows の GUI では  のように見えるもの

どうすれば、プログラムを書いたことになるの？

■ Anaconda Prompt上での実行方法

- ①「Anaconda Prompt」というプログラムを起動して行う
- ② TeraPad(でなくてもいいが)で、プログラムを書く(キーボードから入力する)
- ③ ファイルにセーブ(ハードディスクに入れること)
 - 仮に sample1.py (全て小文字)という名前だとして以下のは、
- ④「Anaconda Prompt」上でpython sample1.py  と入力
- ⑤ エラーがなければ結果が得られる

Enterキー

詳細は次頁以降のスライドを見て下さい



Anaconda Prompt上での実行

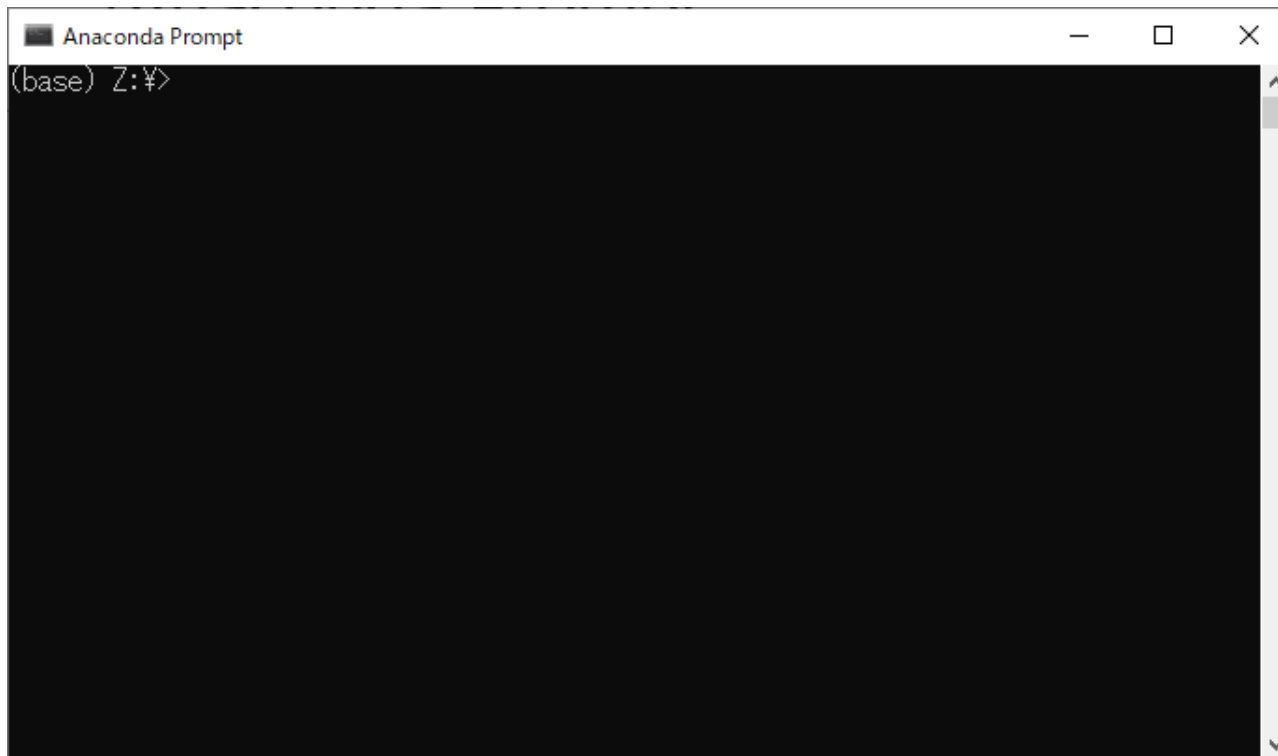
プログラムの書き方その①②

Pythonプログラムの実行

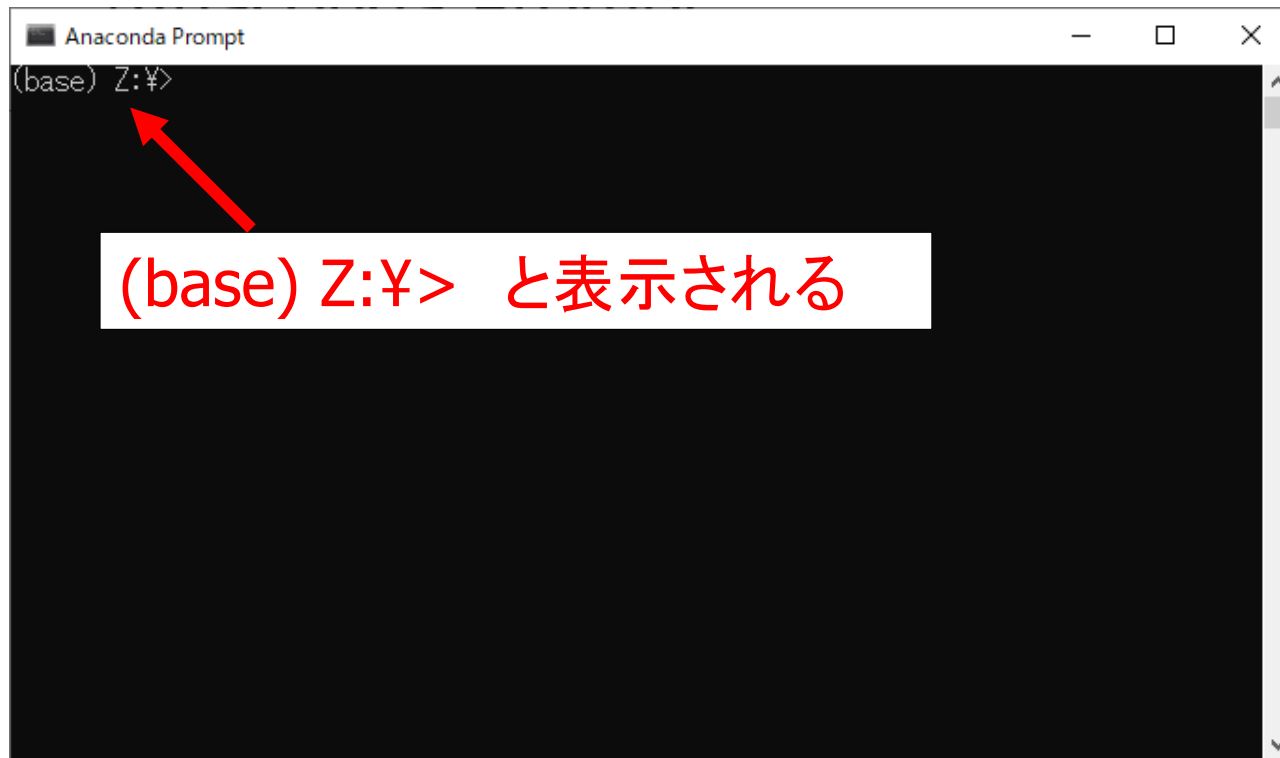
Anaconda Promptの起動①

(日吉ITCの場合)

- Windowsボタン→Anaconda3(64-bit)
→Anaconda Prompt



Anaconda Promptの画面

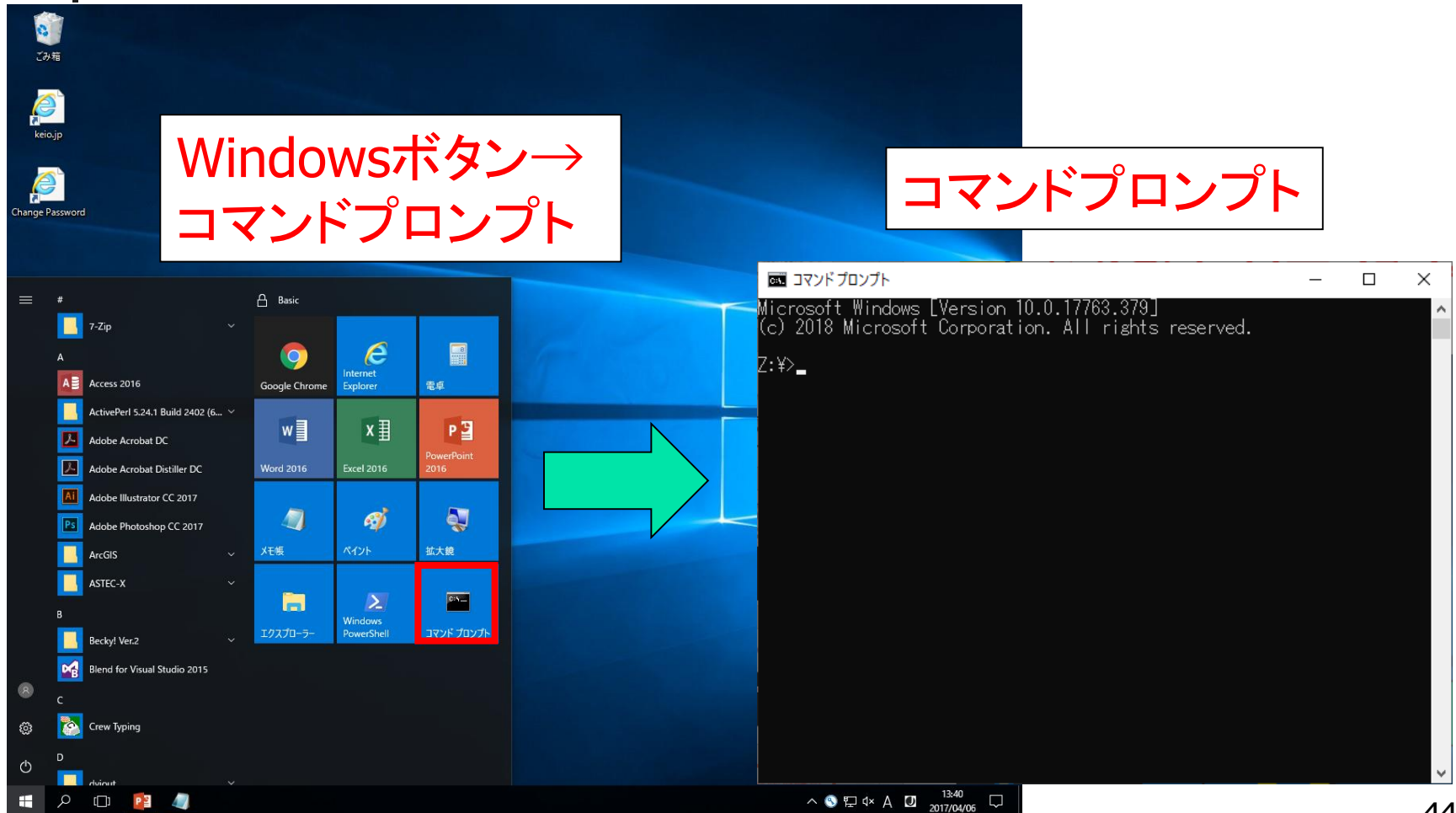


詳しい人へ:

コマンドプロンプトからpythonを実行した場合、バージョンが若干古い(3.6.8)pythonでプログラムを実行することになりますので注意して下さい。

Windowsボタン→
コマンドプロンプト

コマンドプロンプト





プログラムの書き方

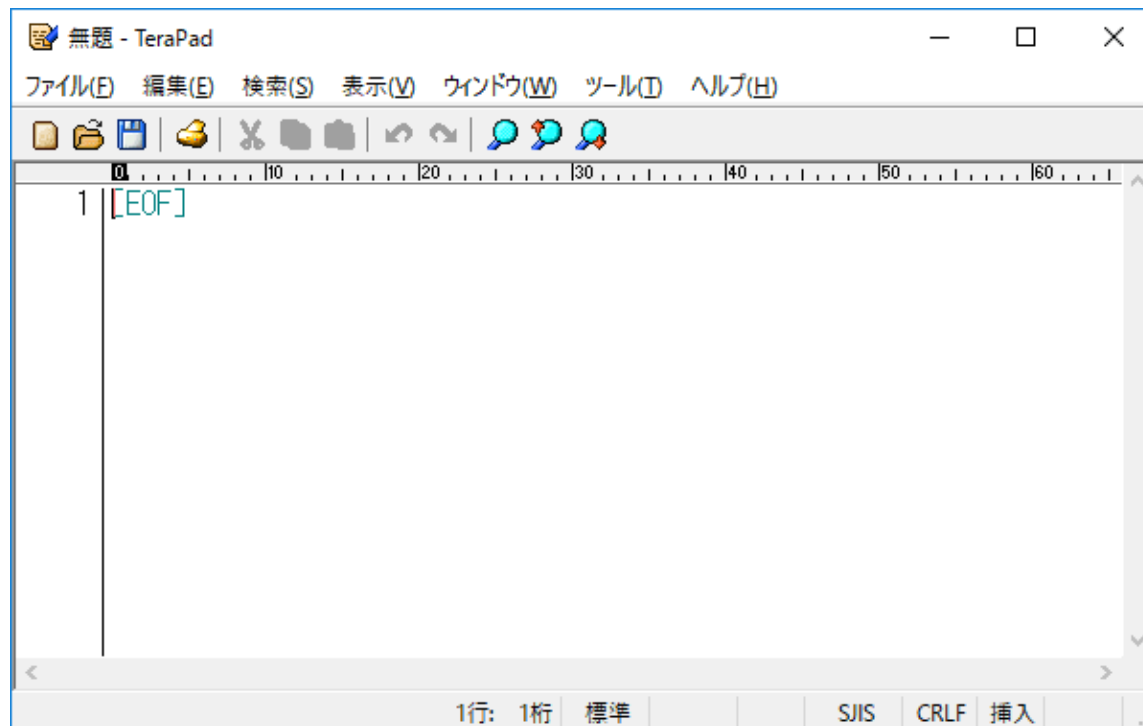
- 二つの作成手順を紹介します
- 初心者はファイルの「拡張子」で混乱します
- どちらの方法でもよいので慣れて下さい

プログラムの書き方その①

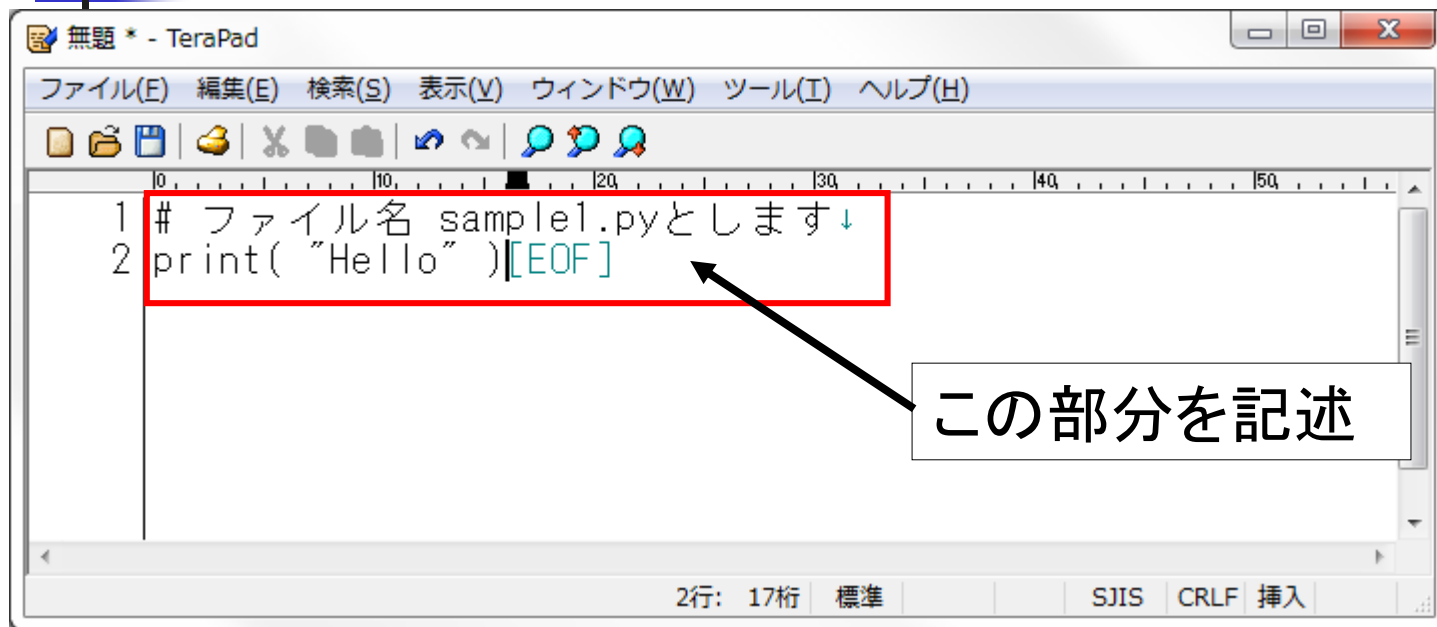
（「TeraPad」を用いる場合）

エディターの起動

- 「Windowsボタン」→「TeraPad」→「TeraPad」



プログラムの記述



```
1 # ファイル名 sample1.pyとします↓
2 print( "Hello" ) [EOF]
```

この部分を記述

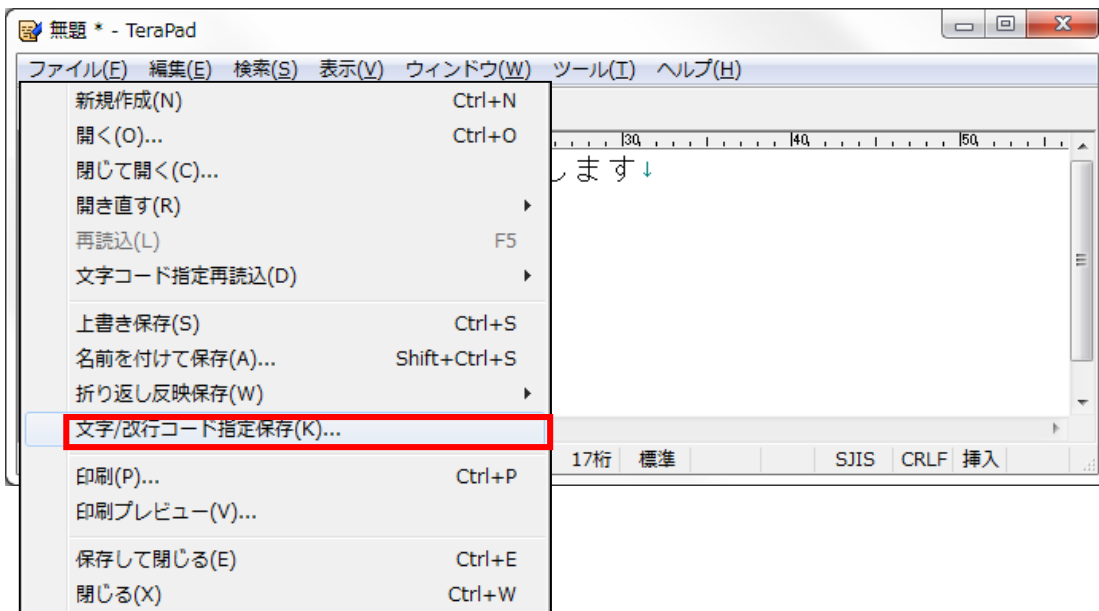
日本語以外は半角文字で書いて下さい
" " (ダブルクォート)は半角文字で書いて下さい

全角の空白は絶対に使わないで下さい

"
2 ふ

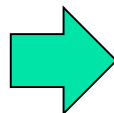
プログラムの保存①

- メニューバーの「ファイル」→「文字/改行コード指定保存(k)」



プログラムの保存②

① 文字コード:「UTF-8」に変更



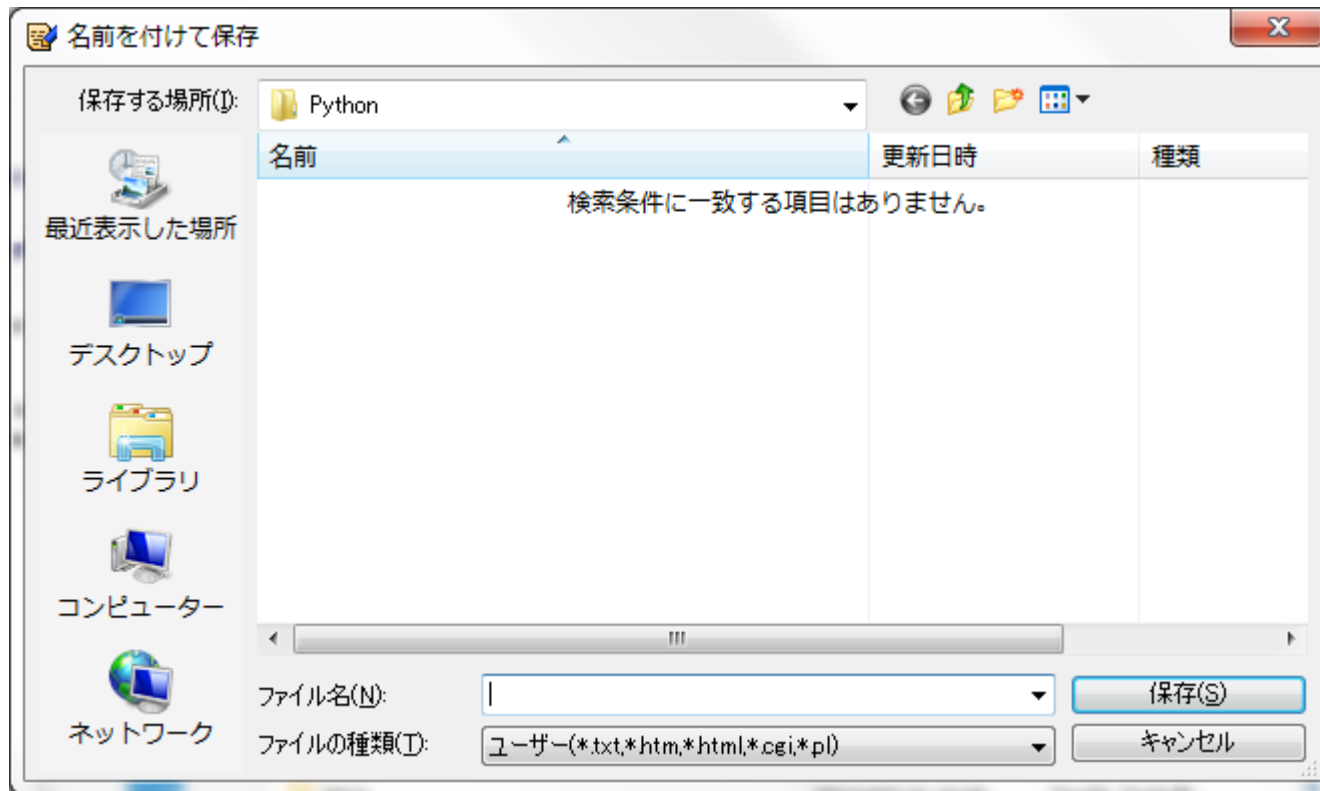
② 名前を付けて保存

③ 「OK」をクリック

文字コードの設定は一ファイルにつき一回でけっこうです

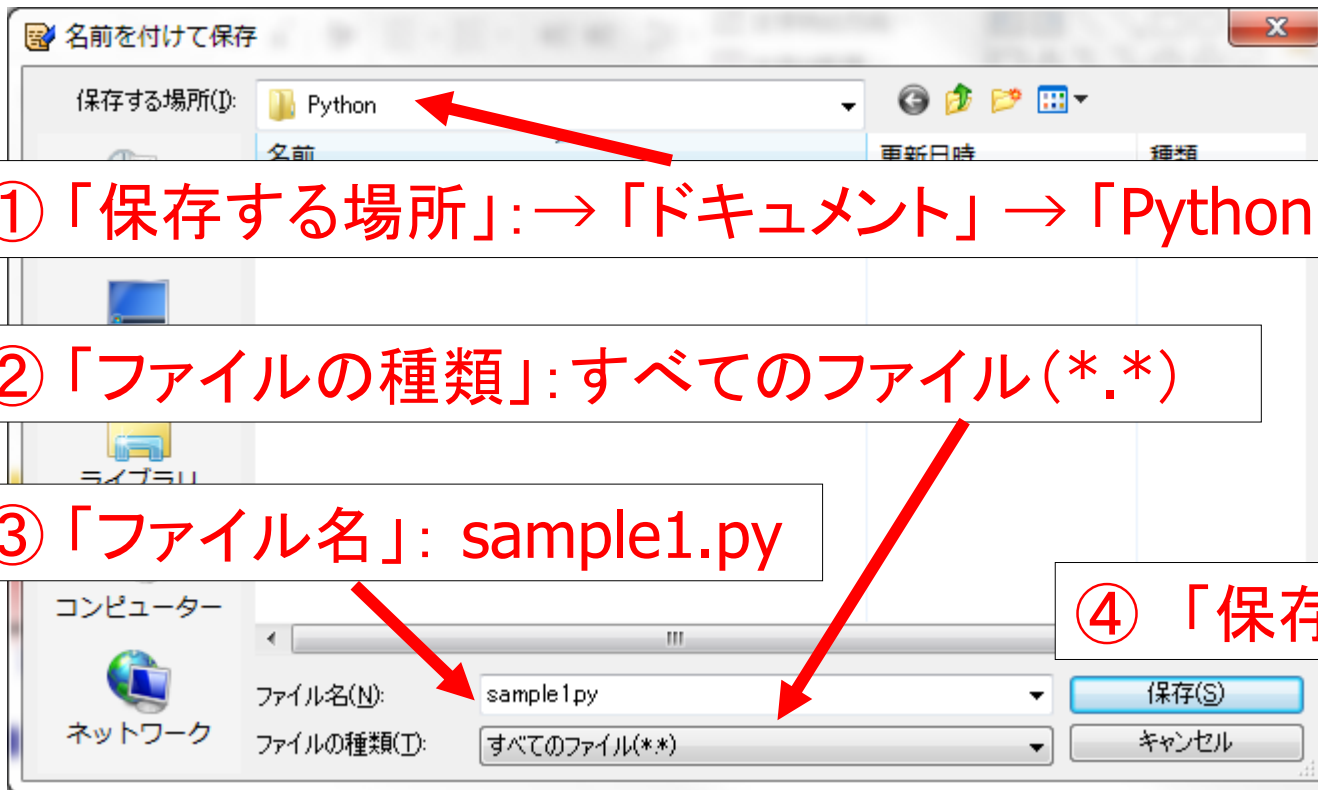
プログラムの保存③

- メニューバーの「ファイル」→「名前を付けて保存」



プログラムの保存④

- メニューバーの「ファイル」→「名前を付けて保存」



① 「保存する場所」: → 「ドキュメント」 → 「Python」

② 「ファイルの種類」: すべてのファイル (*.*)

③ 「ファイル名」: sample1.py

④ 「保存」をクリック



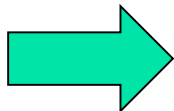
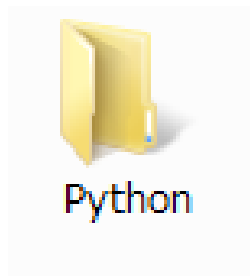
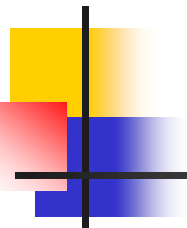
文字コード

- 文字コードとは、コンピュータ上で、文字を表現するための対応表のことです
 - 英数字の場合、asciiコード(1バイト)
- 日本語の場合、2バイト必要で、いろいろな文字コードがあります
 - JIS, Shift-JIS, EUC
- 近年は、UTF-8が利用されるようになっています
- Python(バージョン3)で日本語を用いる場合、文字コードはUTF-8を指定して下さい

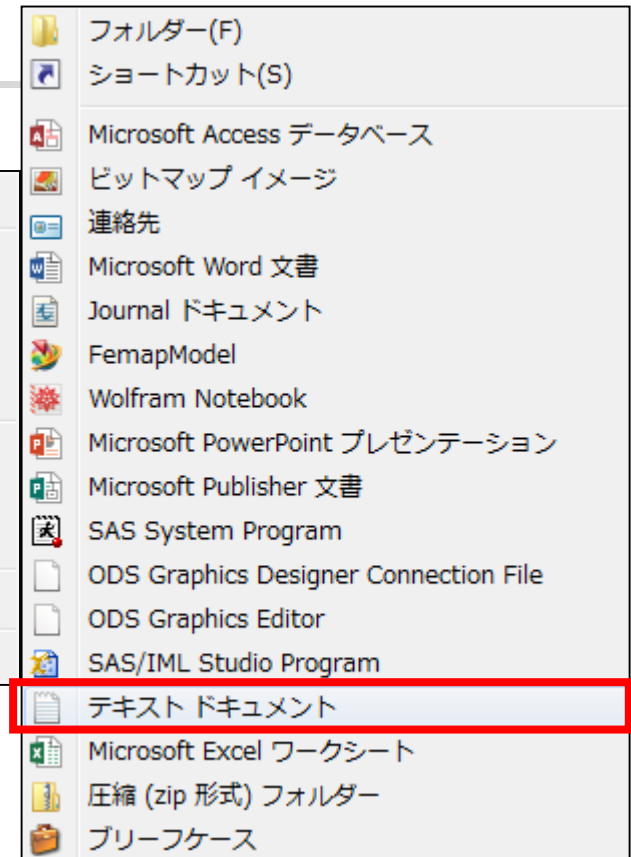
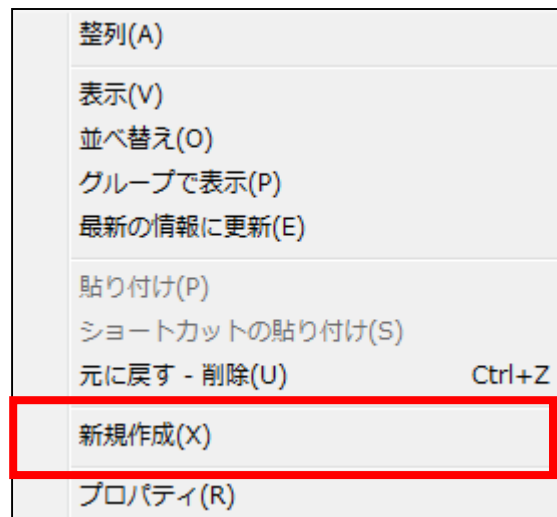
プログラムの書き方その② （「TeraPad」を用いる場合）



プログラムファイルの作成①

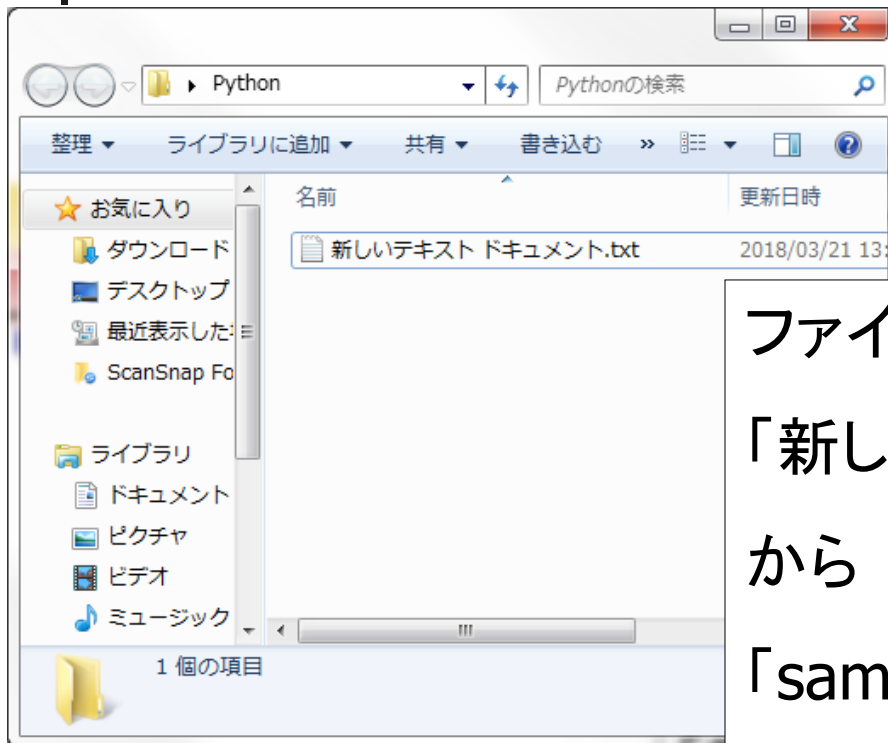


ダブルクリックして
「Python」フォルダ
を開く



「Python」のフォルダー内で右クリック→
「新規作成」→「テキストドキュメント」

プログラムファイルの作成②



ファイル名の変更

「新しいテキストドキュメント.txt」

から

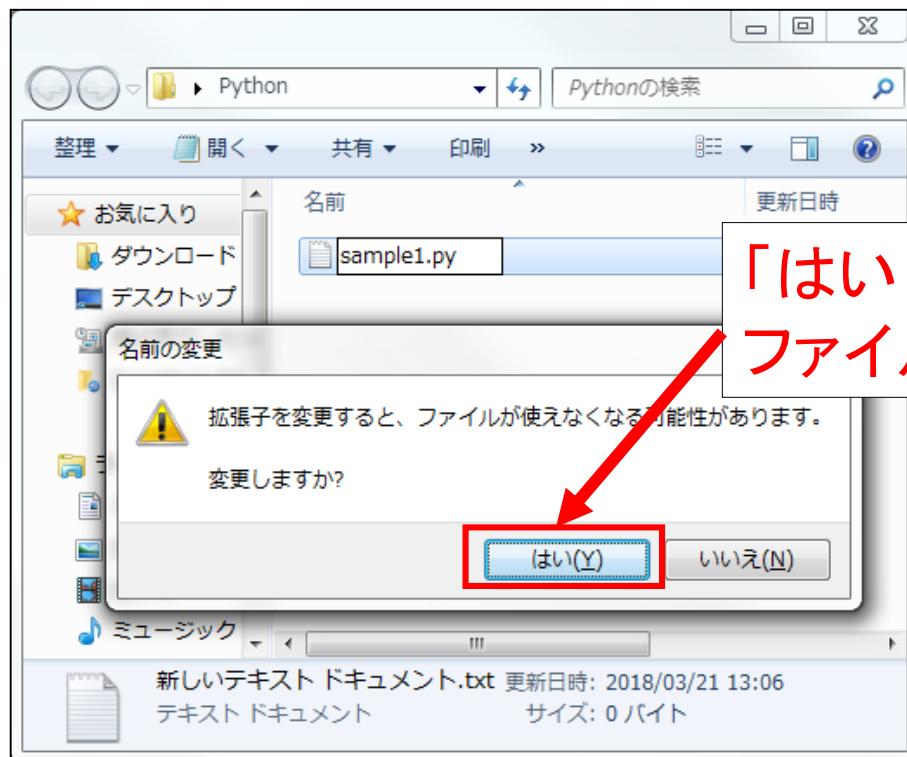
「sample1.py」

に変更する

「sample1.py」は半角文字として下さい

プログラムファイルの作成③

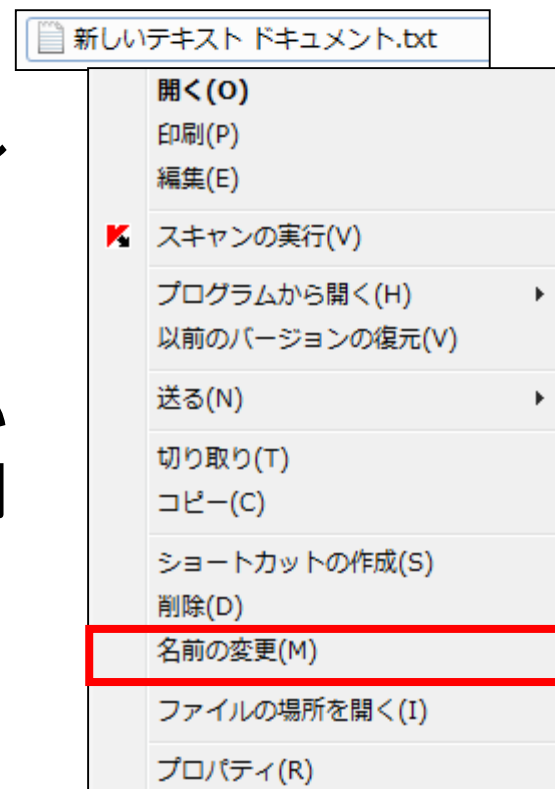
ファイル名を変更すると...



「はい(Y)」をクリック→
ファイル名が変更される

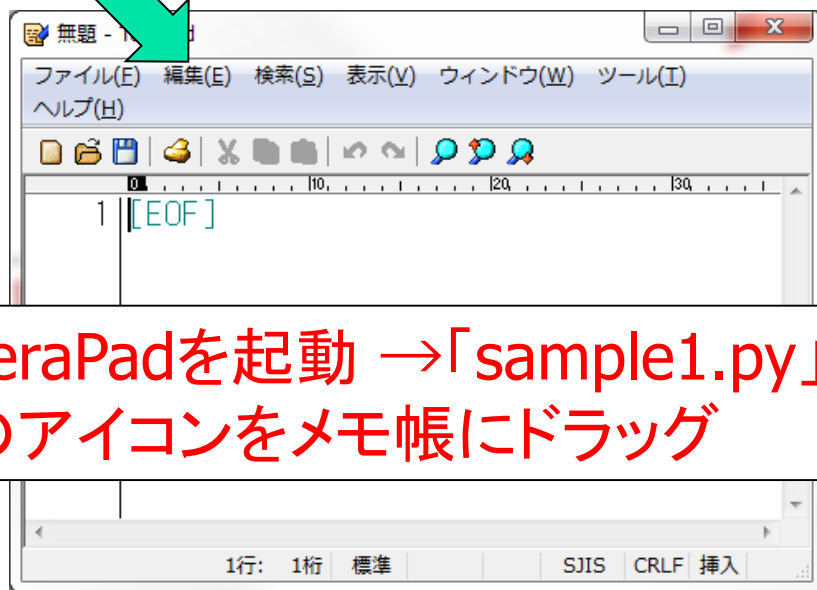
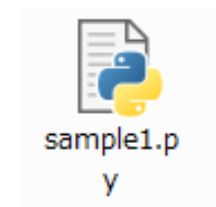
ファイル名の変更方法

- ファイルを選択→右クリック →「名前の変更(M)」
- ファイルの名前を **sample1.py** としてください
 - 半角文字
 - 今回の講義では、拡張子(この例でいえば(.py))は.pyでなくても(.txtでも)問題はおこらない(はず)

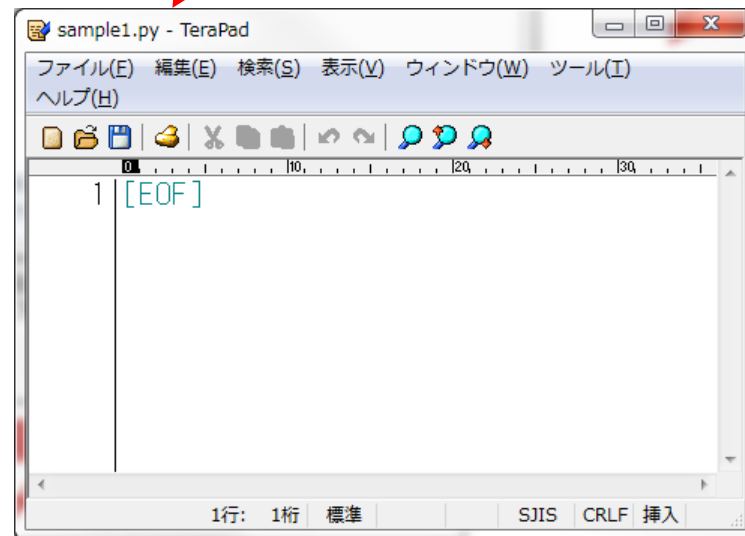


エディターの起動

タイトルが「無題」から「sample1.py」
に変わる



TeraPadを起動 →「sample1.py」
のアイコンをメモ帳にドラッグ



プログラムの記述



```
sample1.py * - TeraPad
ファイル(E) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(I) ヘルプ(H)
1 # ファイル名 sample1.pyとします↓
2 print("Hello") [EOF]
```

この部分を記述

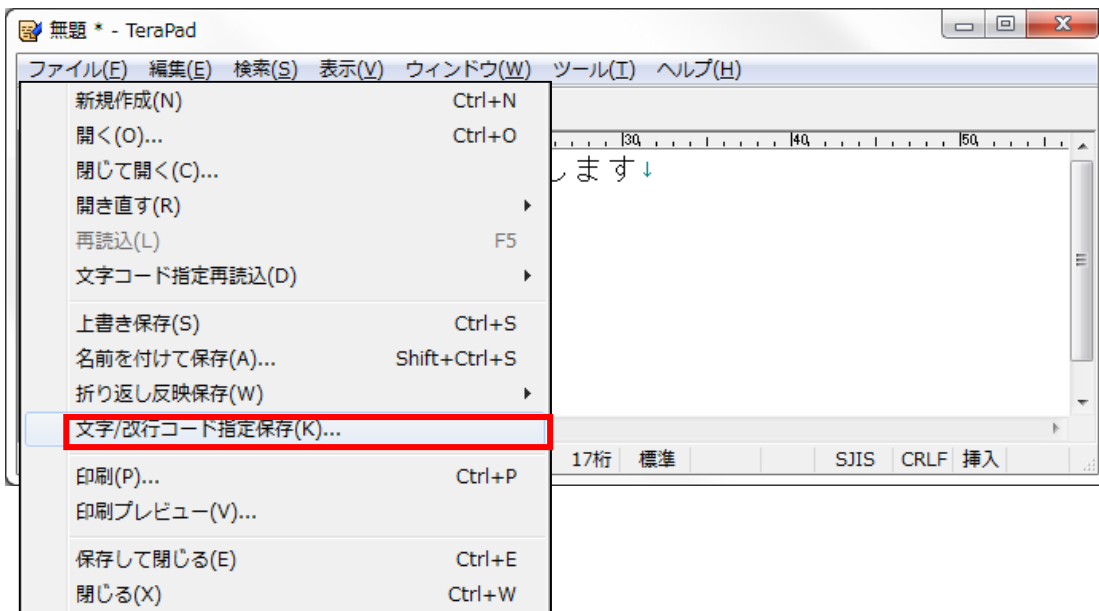
日本語以外は半角文字で書いて下さい
" " (ダブルクォート)は半角文字で書いて下さい

全角の空白は絶対に使わないで下さい

"
2 ふ

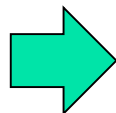
プログラムの保存①

- メニューバーの「ファイル」→「文字/改行コード指定保存(k)」



プログラムの保存②

① 文字コード:「UTF-8」に変更



② 上書き保存

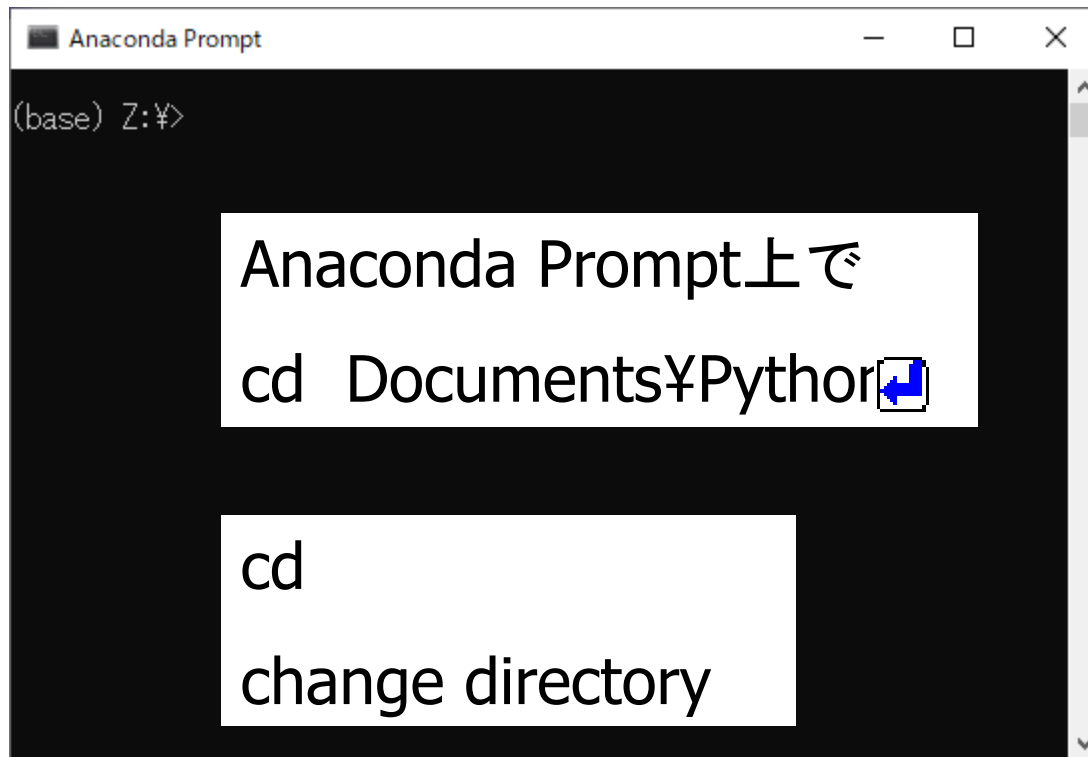
③ 「OK」をクリック

文字コードの設定は一ファイルにつき一回でけっこうです



Pythonプログラムの実行

Pythonフォルダーへの移動①



Anaconda Prompt

```
(base) Z:\>
```

Anaconda Prompt上で
cd Documents\Python

cd
change directory

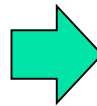


Enterキー

Pythonフォルダーへの移動②

```
Anaconda Prompt
(base) Z:\>cd Documents¥Python
```

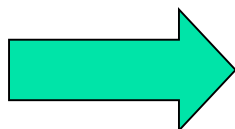
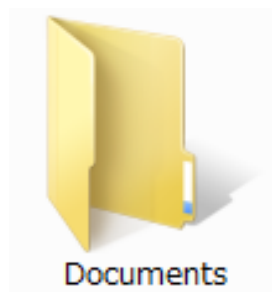
cd Documents¥Python



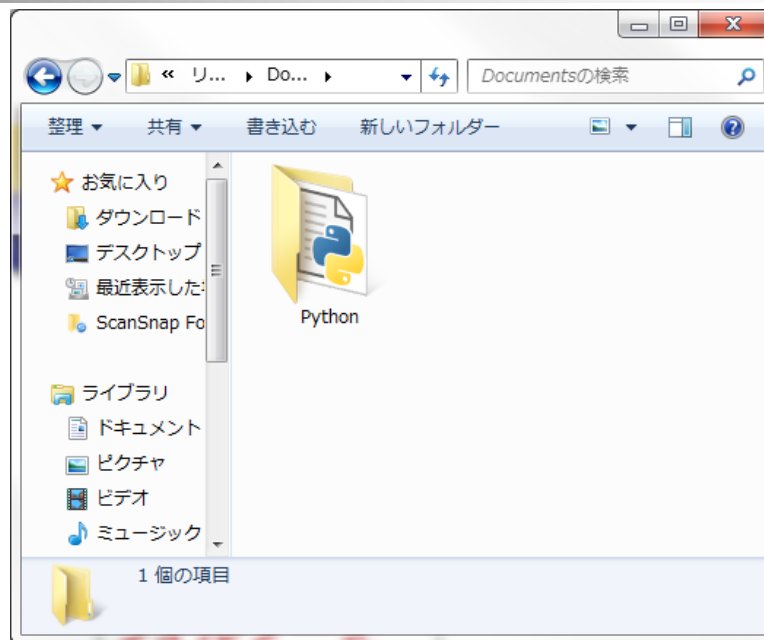
```
Anaconda Prompt
(base) Z:\>cd Documents¥Python
(base) Z:\Documents¥Python>_
```

Z:\Documents¥Python>
と変わる

コマンドプロンプト①



フォルダーを
ダブルクリック



コマンドプロンプト上で

Z:> cd Documents 

コマンドプロンプト②

(base) Z:¥Documents¥Python>dir

dir  と入力

ドライブ Z のボリューム ラベルは md201 です
ボリューム シリアル番号は 009A-9A03 です

Z:¥Documents¥Python のディレクトリ

dir

フォルダ内のファイル名を表示

```
2019/04/01  14:08    <DIR>        .
2019/04/01  13:42    <DIR>        ..
2019/03/28  12:34    <DIR>        2018
2019/04/01  13:56             61 sample1.py
2018/04/09  12:36             156 sample4.py
2018/05/14  10:23             242 sample5.py
2018/06/11  12:12             102 sample6.py
2019/03/28  12:34    <DIR>        tmp
                4 個のファイル                561 バイト
                4 個のディレクトリ 1,779,211,386,880 バイトの空き領域
```



GUI (Graphical User Interface)

- 表示として, グラフィックスを用いたユーザインタフェース. 入力は, マウスやそれと類似した装置を用いる
- パソコンでは, Macintosh が使い始めた
- 今では, これが常識



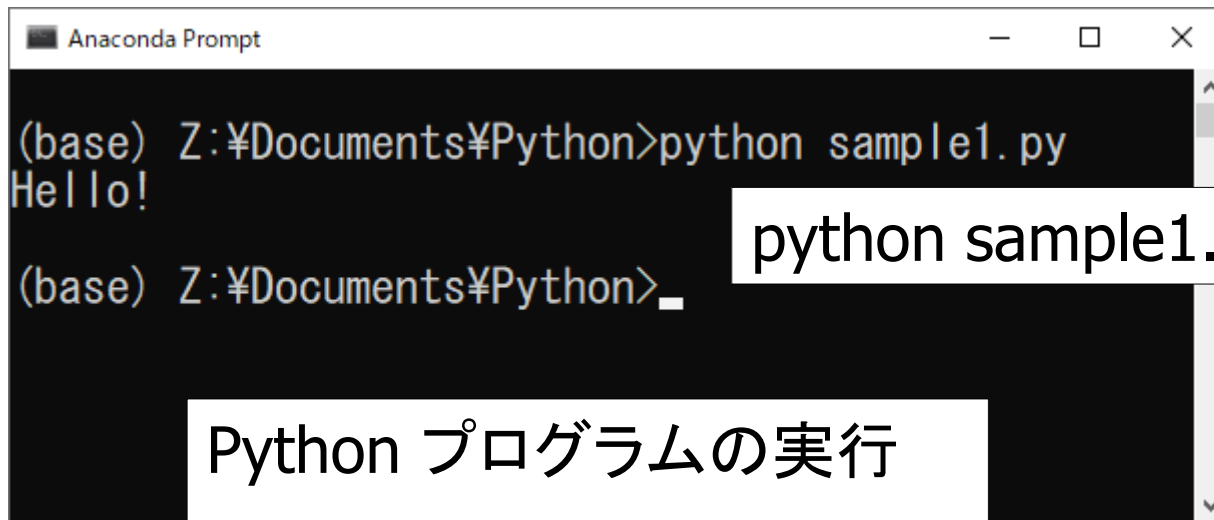
CLI or CUI

(Command Line User Interface)

- 表示として、文字列を用いたユーザインタフェース
- 入力はキーボードを用いる
- 入力するものは、コンピュータに対するコマンドであり、行(ライン)単位に入力する。入力する場所をコマンドラインという。コマンドの実行結果はコマンド入力直後に表示する。画面を使い切ると、スクロールする
- Windows 10/8/7/Vista/XP/2000では、コマンドプロンプトという言葉が用いられる
 - コマンドプロンプトは、本来は、コマンドラインの先頭にコンピュータが書く文字である
 - Anaconda Promptもコマンドプロンプトです

Pythonプログラムの実行

- python とは Pythonプログラムを実行するコマンド
 - 指定されたファイルの中身を見て、それに従った動作をする



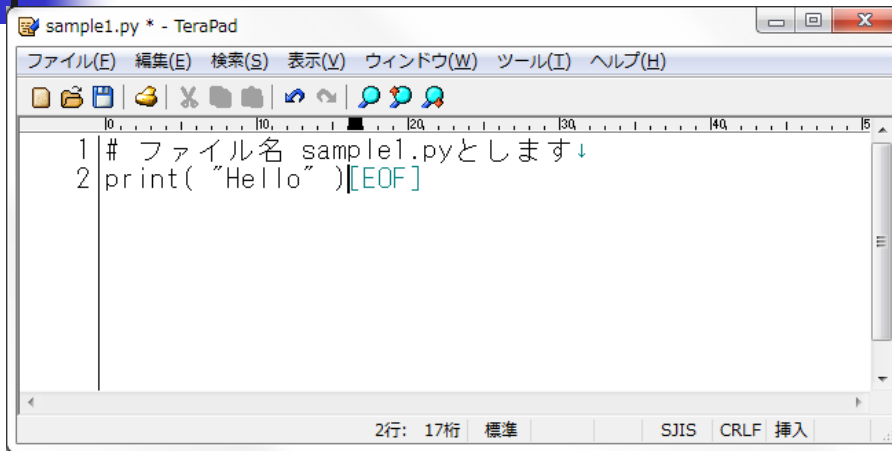
The screenshot shows an Anaconda Prompt window with a black background and white text. The prompt is `(base) Z:\¥Documents¥Python>`. The user has entered `python sample1.py` and the output is `Hello!`. The prompt is now `(base) Z:\¥Documents¥Python>_`.

python sample1.py

Python プログラムの実行

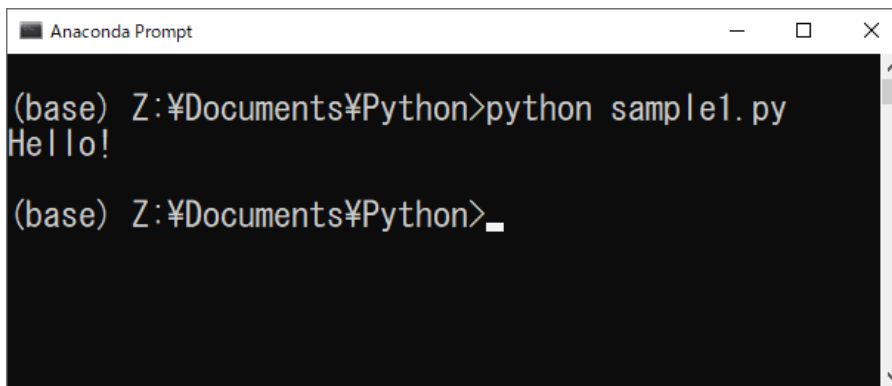
python Pythonプログラム

Pythonプログラムの実行方法のまとめ



A screenshot of the TeraPad text editor window. The title bar reads "sample1.py * - TeraPad". The menu bar includes "ファイル(F)", "編集(E)", "検索(S)", "表示(V)", "ウィンドウ(W)", "ツール(I)", and "ヘルプ(H)". The toolbar contains icons for file operations and editing. The main text area shows two lines of code: "1 # ファイル名 sample1.pyとします↓" and "2 print("Hello")". The status bar at the bottom indicates "2行: 17桁", "標準", "SJIS", "CRLF", and "挿入".

```
1 # ファイル名 sample1.pyとします↓
2 print( "Hello" )
```



A screenshot of the Anaconda Prompt terminal window. The title bar reads "Anaconda Prompt". The terminal shows the command "(base) Z:\¥Documents¥Python>python sample1.py" being executed, followed by the output "Hello!". The prompt then changes to "(base) Z:\¥Documents¥Python>_".

```
(base) Z:\¥Documents¥Python>python sample1.py
Hello!
(base) Z:\¥Documents¥Python>_
```

① プログラム
テキストエディタ(TeraPad)
で記述

② Anaconda Promptを起
動

③ Pythonプログラムの実行
python Pythonプログラム



プログラムが動かない場合

エラーメッセージについて



コンピュータは忠実である

- 言われたとおりに，実行する
- 規則通りに書かれていない場合は，実行せずに，エラーメッセージを出力する
- 書かれたように読む
 - 決して，「きっこう書きたかったのだろうなあ」と考えて読むことはしない
 - 勿論，「『きっこう書きたかったのだろうなあ』」と考えて読む」ようにプログラムを書けば，そう書いた範囲で「考えて読む」ようにはなる



プログラム構文上の大原則

- 括弧(広い意味での括弧です)は, 開いたら, 必ず閉じる
 - Pythonでの例外: 「#」で始まるコメント(プログラムと関係のない書き込み)は, 改行(そして改行のみ)が閉じる記号
- 複数種の括弧が混じるときには, 互いに交錯してはならない
 - 例: { ([]) }
 - 誤例: {] { ([)] }



空白について

- Pythonにとって、空白は区切り文字.
- 行の先頭での空白は**インデント**と呼ばれ重要です.
- しかし、Pythonが空白とみなす空白は半角(1バイトコード)の空白だけ. **全角(2バイトコード)**の空白はPythonにとっては空白ではない
 - よく読んで下さい. 決して、禅問答ではありません
- どちらの空白かは、人間がみて区別しにくいので、ちょっと目には訳の分からないこと、しかし、よく考えれば分かることが起こる



半角文字と全角文字

- プログラムは半角文字で書く
 - ただし例外もあります
 - # の後はコメントであり, この後は全角文字を使用してもよい
 - 変数名(後述)も全角文字を使用してもよい(しかし, 推奨しません. この講義では使わないで下さい)
 - “ ” の中は全角文字を使用してもよい
- ただし文字コードとして必ず「**UTF-8**」を指定すること



文字について

- 日本語 Windows が取り扱う文字には、1バイトコード(半角文字)と2バイトコード(全角文字)とがある
- 昔は、本当に、半角と全角で表示されていたので分かりやすかったが、今では、プロポーショナルフォントなどを用いるので、分かりにくい
 - 例: A A と並べれば分かるが KEIO (Iは全角)
- コンピュータはちゃんと区別するからやっかいだ



どこが間違っているでしょうか①

```
# error-1.py  
print( "Hello!" )
```

実行結果*

```
Z:\Documents\Python>python error-1.py  
File "error-1.py", line 2  
    print( "Hello!" )  
          ^  
SyntaxError: invalid character in identifier
```

*スライドは教員用のPCで作成しています。エラーメッセージは日吉ITCのPCと異なる場合があります。

どこが間違っているでしょうか①

実行結果

```
Z:¥Documents¥Python>python error-1.py
File "error-1.py", line 2
  print( "Hello!" )
        ^
SyntaxError: invalid character in identifier
```

```
# error-1.py
print(□"Hello!" )
```

2行目のここにエラーがあると知らせてくれている

半角ではなく全角の空白となっている



どこが間違っているでしょうか②

```
# error-2.py  
print( "Hello!")
```

```
Z:¥Documents¥Python>python error-2.py
```

```
File "error-2.py", line 2
```

```
    print( "Hello!" )  
           ^
```

```
SyntaxError: EOL while scanning string literal
```


どこが間違っているでしょうか②

```
Z:\Documents\Python>python error-2.py
```

```
File "error-2.py", line 2
```

```
print( "Hello!" )
```

^

```
SyntaxError: EOL while scanning string literal
```

2行目のここにエラーがあると知らせてくれている

```
# error-2.py
```

```
print( "Hello!" )
```

全角文字の「」

どこが間違っているでしょうか③

```
Z:¥Documents¥Python>python error-3.py
```

```
File "error-3.py", line 1
```

```
# error-3.py
```

```
^
```

1行目のここにエラーがあると知らせてくれている

```
SyntaxError: invalid character in identifier
```

全角文字

```
# error-3.py  
print( "Hello!" )
```

どこが間違っているでしょうか④

```
Z:¥Documents¥Python>python error-4.py
```

```
File "error-4.py", line 2
```

```
print( "Hello!" ]
```

^

2行目のここにエラーがあると知らせてくれている

```
SyntaxError: invalid syntax
```

```
# error-4.py  
print( "Hello!" ]
```

全角文字

どこが間違っているでしょうか⑤

```
Z:¥Documents¥Python>python error-5.py
Traceback (most recent call last):
  File "error-5.py", line 2, in <module>
    print( "Hello!" )
NameError: name 'print' is not defined
```

```
# error-5.py
print( "Hello!" )
```

2行目にエラーがあると知らせてくれている

「print」ではなく「print」



繰り返しになりますが...

- 言われたとおりに, 実行する
- 規則通りに書かれていない場合は, 実行せずに, エラーメッセージを出力する
- 動かなかった場合は...
 - エラーメッセージを見てどこの行にエラーがあるかを見つける
 - どうして間違っているかを考える
 - エラーを修正し, 再度実行する



練習問題

他の例題①

(同じようにプログラミングしてみてください. ファイル名は自由につけても結構です)

四則演算を行なうPythonプログラム

四則演算

a=5

b=4

print("a+b=" , a+b)

print("a-b=" , a-b)

print("a*b=" , a*b)

print("a/b=" , a/b)

以降はコメントです

sample2.py

Z:\Documents\Python>python sample2.py

a+b= 9

a-b= 1

a*b= 20

a/b= 1.25



他の例題②

```
# 日本語の表示  
print( "春の" )  
print( "うららの" )  
print( "隅田川" )
```

sample3.py

```
Z:¥Documents¥Python>python sample3.py  
春の  
うららの  
隅田川
```




他の例題③

```
# 何が起こるでしょうか
from turtle import *
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
```

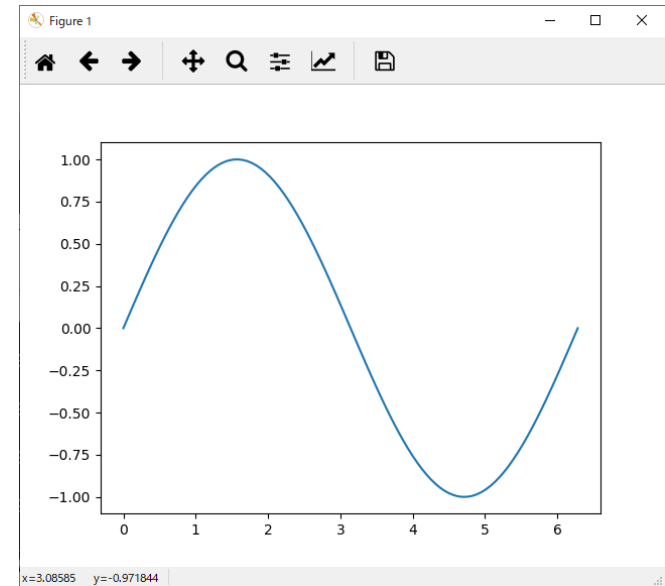
```
Z:¥Documents¥Python>python sample4.py
```

sample4.py

他の例題④

```
# サインカーブ
import math
import numpy as np
from matplotlib import pyplot
x = np.linspace(0, 2*math.pi, 100)
y = np.sin(x)
pyplot.plot(x, y)
pyplot.show()
```

Z:\Documents\Python>python sample5.py



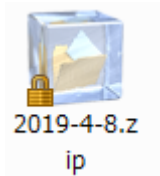
sample5.py



圧縮ファイルの展開方法

圧縮ファイルの展開方法①

① 右クリック*

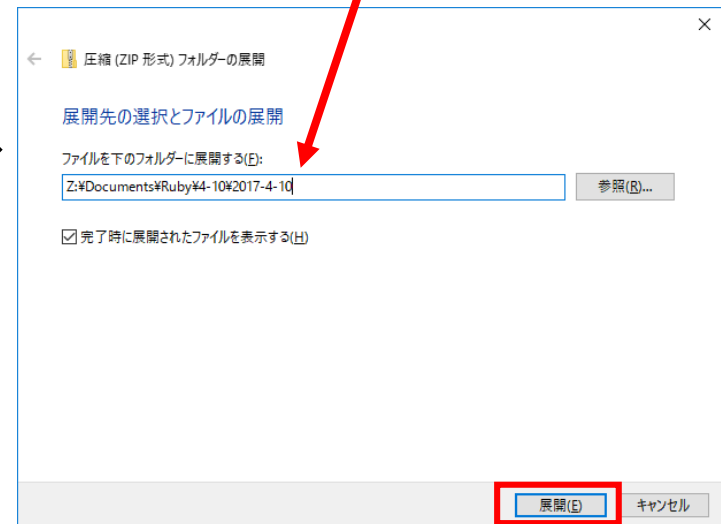


圧縮ファイル
(zip形式)

② 「すべて展開」を選択



展開したいフォルダーを指定

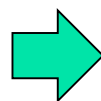
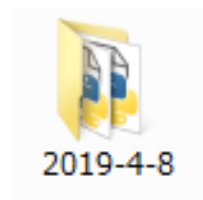


③ 「展開」をクリック

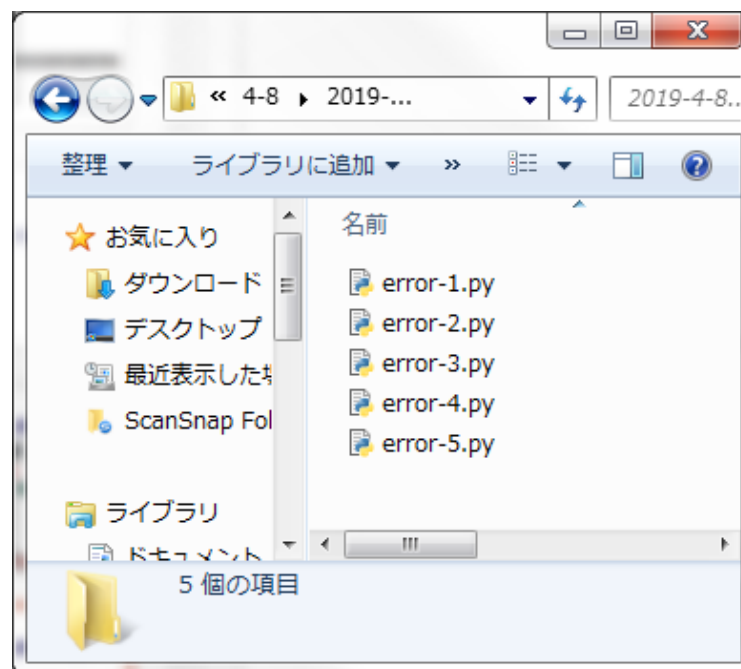
*ダブルクリックではファイルは解凍されないので注意

圧縮ファイルの展開方法②

展開されたフォルダー



展開されたファイル





Pythonに関する情報



バージョンでの違い

- 現在, 用いられているPythonにはバージョン2とバージョン3があります
- 日吉ITCのPCにはどちらのバージョンもインストールされています
- 講義ではバージョン3を用います



Anaconda版Python

- Pythonは多くの便利なパッケージがあります.
- こうしたパッケージは, 使いたくなかった際, 自分でインストールする必要があります.
- こうした多数のパッケージが最初からインストールされているpythonがAnacondaです.
- プログラムの基本的な書き方, 実行方法はどちらも変わりません. 講義ではAnaconda版のpythonを用います.

Python 関連サイト(2019年3月現在)

- Official site:
<https://www.python.org/>
- マニュアル
 - 「Documentation」→「Python3.X Docs」
- Pythonのインストール(最新版は3.7.2)
 - 「Downloads」→「Release Version」→「Python 3.7.2」→「Files」→「Windows x86-64 executable installer」(Windows OS 64bit版)をダウンロード



Python 関連サイト(2019年3月現在)

- Anaconda
 - <https://www.anaconda.com/>
 - python, 科学技術計算パッケージ
- Anacondaのインストール
 - 「Download Now」→「Windows」→「Python 3.7 version Download」
- Anacondaのインストールをお薦めします. ただし, 通常のpythonより, 多くのディスク容量が必要です.





参考書

- 各種出ています. 自分の気に入ったものでよいと思います. on-line文書もあります
- Pythonチュートリアル
 - <https://docs.python.jp/3/tutorial/>
- Python入門(バージョン2)
 - <http://www.tohoho-web.com/python/index.html>

Python (Anaconda版) のインストール



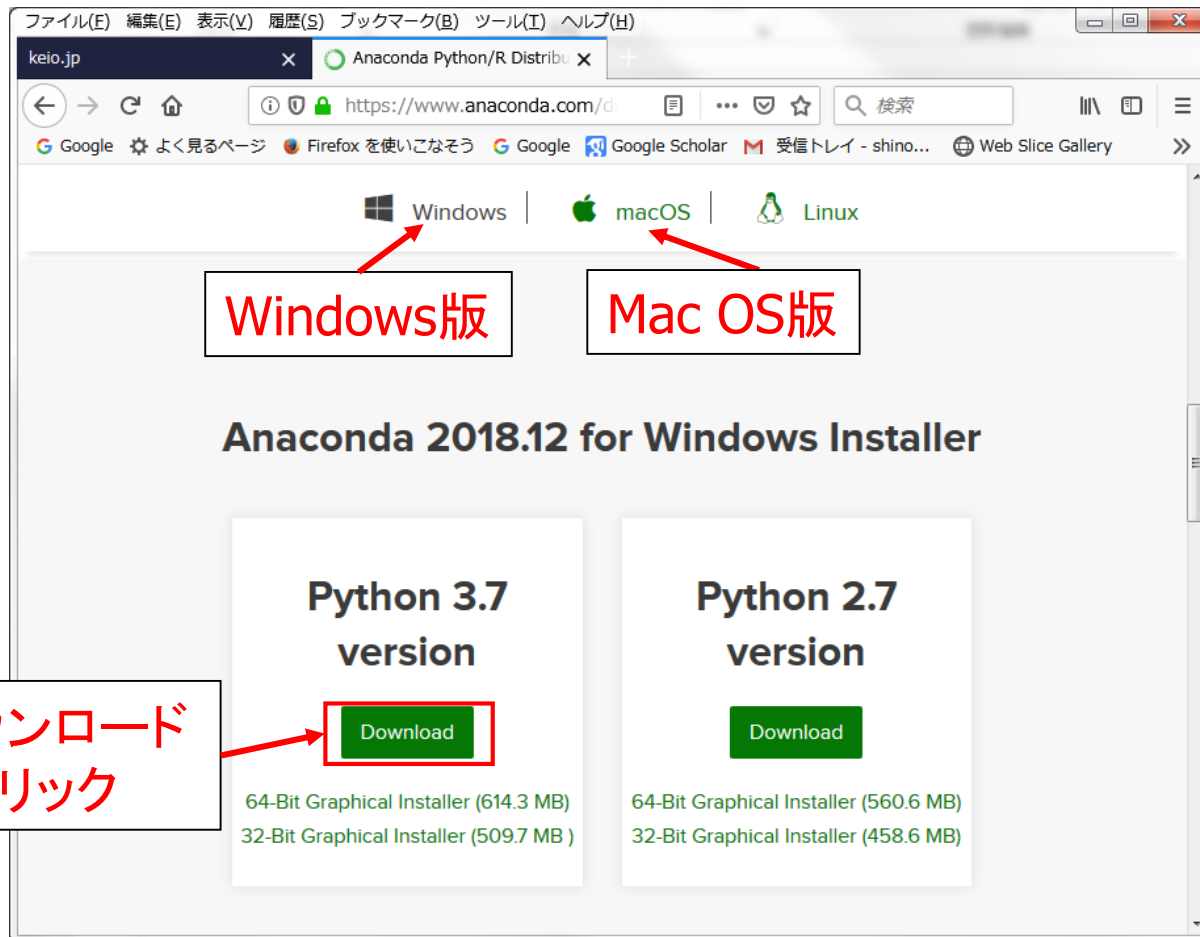
個人PCへのPythonのインストール



Python (Anaconda版) のインストール①

- Anacondaのインストール(2019年4月現在)
 - <https://www.anaconda.com/>
 - 「Download」→次頁

Pythonのインストール②

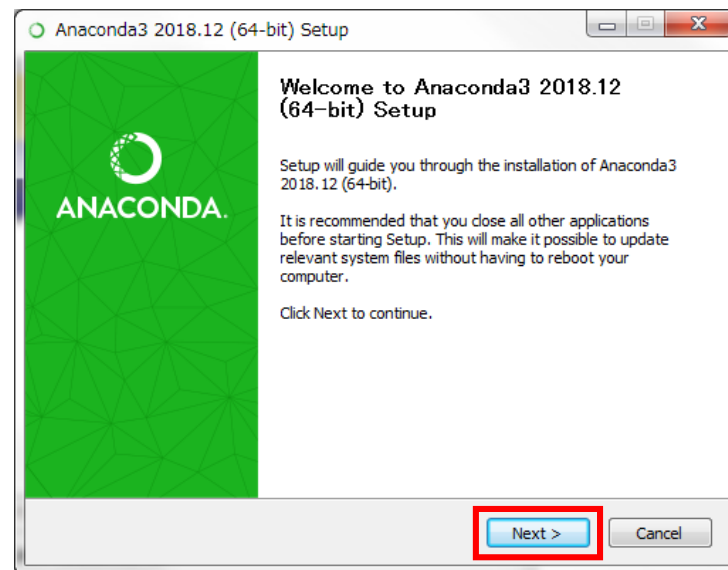
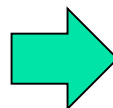
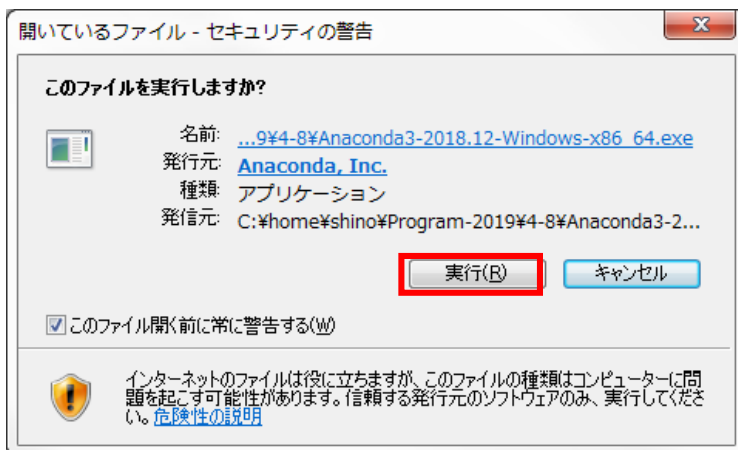


Pythonのインストール③



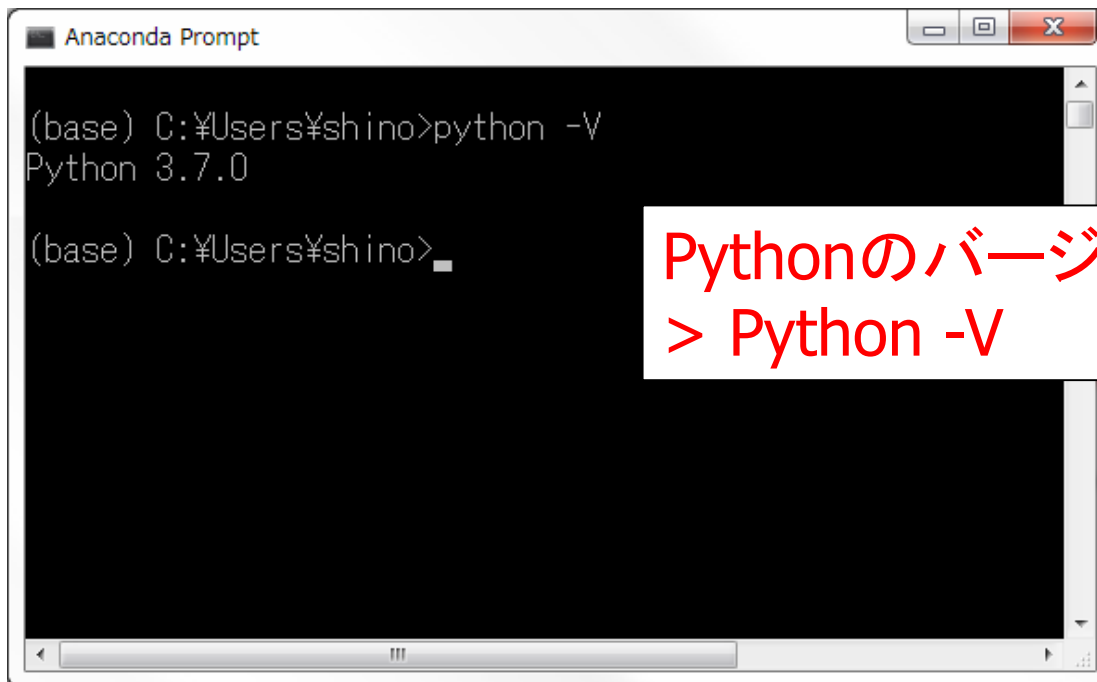
Anaconda3-2018.12-Windows-x86_64.exe

ダブルクリック



Pythonがインストールできたかの確認

すべてのプログラム→「Anaconda3(64-bit)」→
「Anaconda Prompt」



```
Anaconda Prompt
(base) C:\Users\shino>python -V
Python 3.7.0
(base) C:\Users\shino>_
```

Pythonのバージョンの確認
> Python -V