



プログラミング言語 第四回

担当: 篠沢 佳久
栗原 聡

2019年5月6日



本日の内容

- Pythonプログラムの作成と実行方法
 - 第一回の実習の復習
 - 68ページまでのスライドはこれまでの復習です
- 標準出力
 - ディスプレイに表示させること
- 標準入力
 - キーボードから入力すること
- 第一回レポート課題



履修者のみなさんへ

- プログラミングは簡単には身に付きません
- 一つの言語をマスターすると他の言語を覚えるのは比較的容易です
- 講義資料には例題を多くつけます
- 講義では全てを説明できませんので、練習問題、レポート問題を含めて復習して下さい



Pythonプログラムの実行

Pythonプログラムの記述と実行



Pythonプログラム①

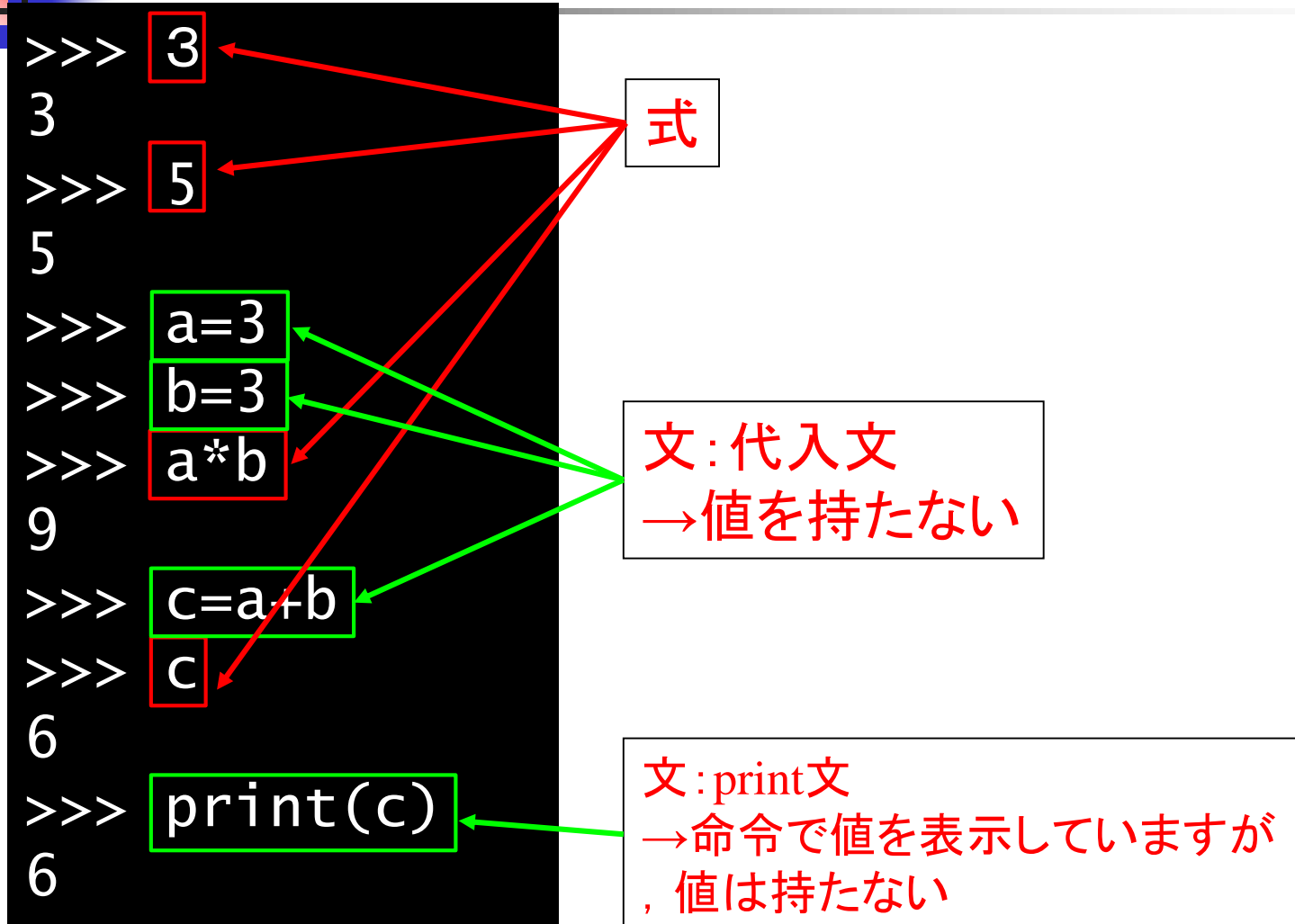
- プログラムは文のまとまりから構成されています
- 文とは、代入文, print文, 条件文, for文など実際に命令を記載する単位のことです
- 多くの文には式が含まれます
- 式とは、実行すると値をもちます
 - 「値」は数値, 文字列, 論理値など
- 式の例:
 - $2+3$ は 5 という値をもつ
 - $x-y$ は (x が5, y が2ならば) 3という値をもつ
 - $x==3$ は, x が値3を持っていれば, True という値をもつ
- 対話型シェル上で表示しているのは, 式の値です



Pythonプログラム②

- 対話型シェル上で実行
 - 一文(もしくは式)ごとに入力
 - 入力する度に実行され, 式の値は表示される
- Pythonコマンドで実行
 - プログラム(文のまとまり)として入力
 - 一行ずつ実行されるが, 式の値は表示されない
 - 表示するためには, print文で表示させなければならない

文と式(復習)





今のところのPythonプログラム①

sample41.py

```
x = 0.1
n = 1
y = 0.3
if n==1: z=x+y
if n!=1: z=x*y
print( "x=", x , "y=", y , "z=", z )
```

対話型シェル上で実行させた場合とPythonコマンドで実行させた場合の違いについて理解する



今のところのPythonプログラム②

プログラムとして実行した場合

```
>python sample41.py  
x= 0.1 y= 0.3 z= 0.4
```

↑
print文で指定された部分のみ出力される

Pythonプログラムの実行
> **python** プログラム名



今のところのPythonプログラム③

対話型シェル上で一文ずつ実行した場合

```
>>> x = 0.1
>>> n = 1
>>> y = 0.3
>>> if n==1: z=x+y

>>> if n!=1: z=x*y

>>> print( "x=", x , "y=", y , "z=", z )
x= 0.1 y= 0.3 z= 0.4
>>>
```



Pythonプログラムの作成

- 練習
- sample41.py を作成し, 実行する

sample41.py

```
x = 0.1  
n = 1  
y = 0.3  
if n==1: z=x+y  
if n!=1: z=x*y  
print( "x=", x , "y=", y , "z=", z )
```



Anaconda Prompt上での実行

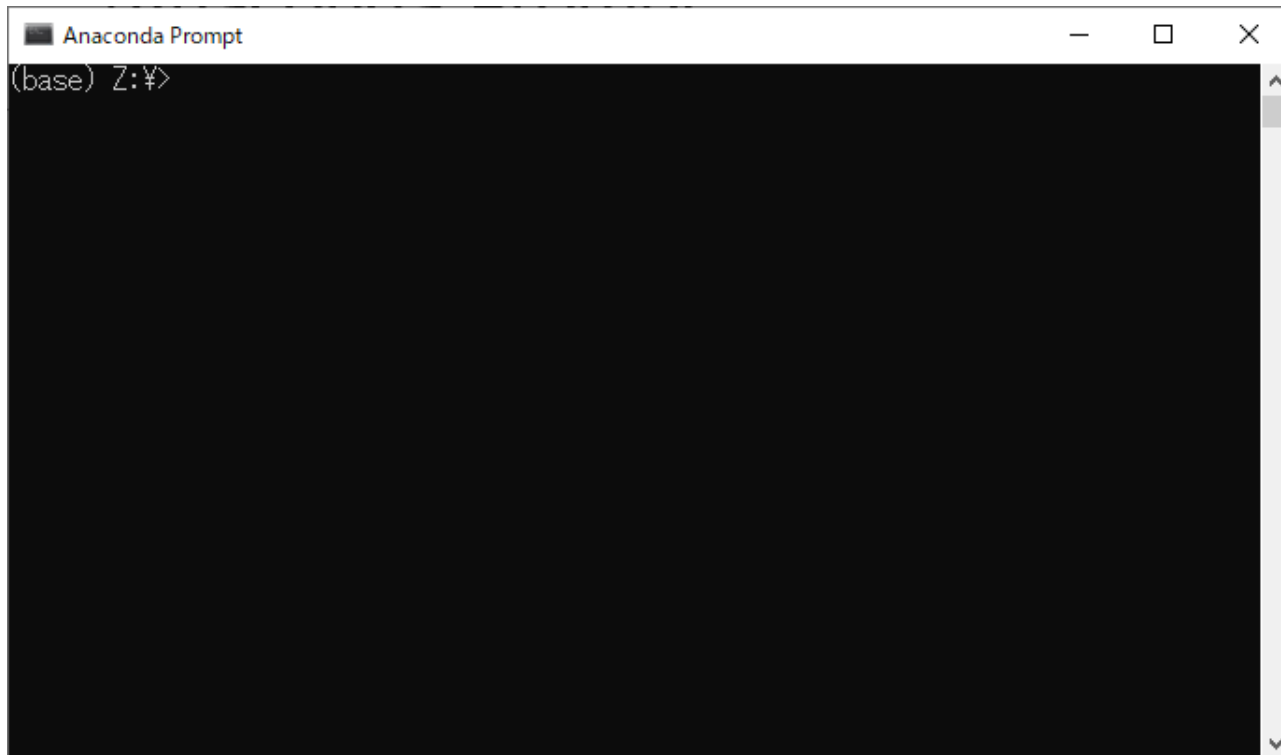
プログラムの書き方その①②

Pythonプログラムの実行

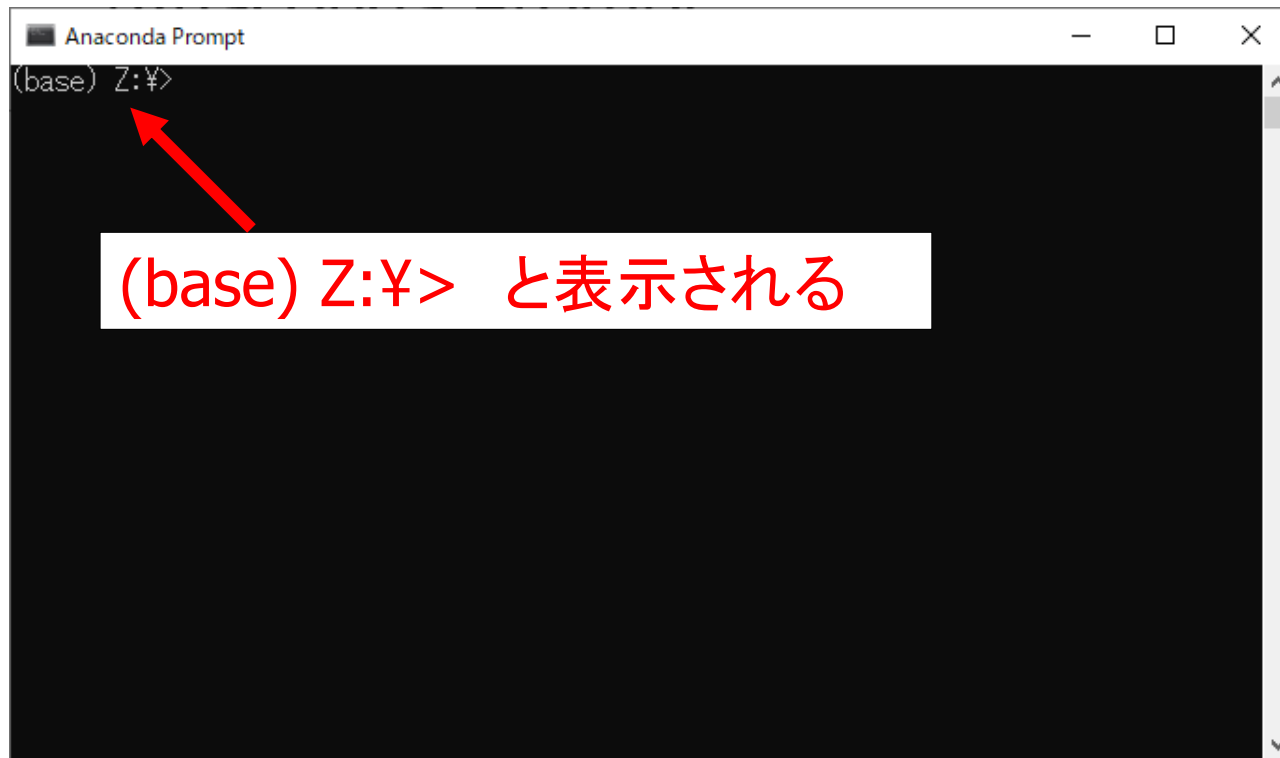
Anaconda Promptの起動①

(日吉ITCの場合)

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



Anaconda Promptの画面

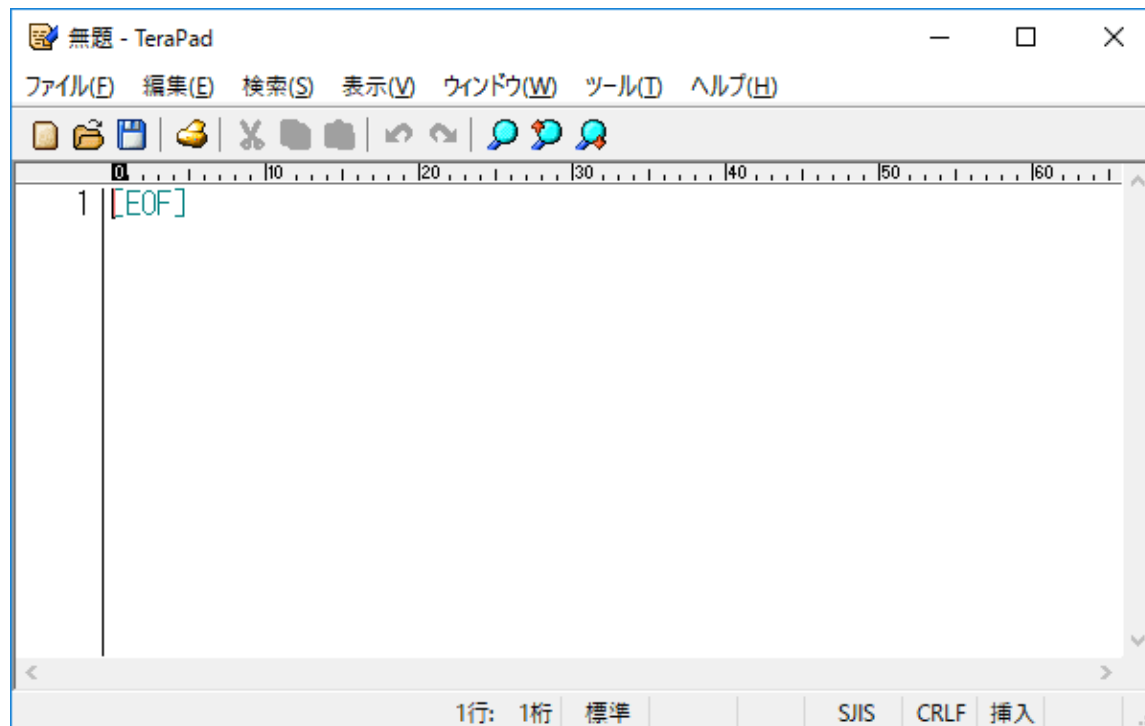


プログラムの書き方その①

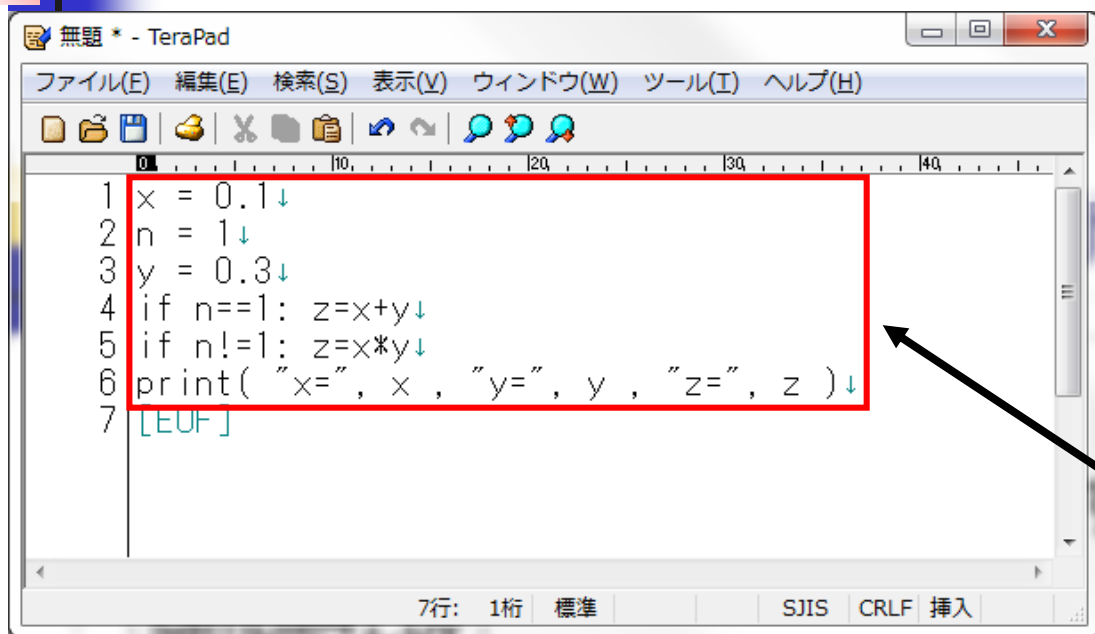
（「TeraPad」を用いる場合）

エディターの起動

- 「Windowsボタン」→「TeraPad」→「TeraPad」



プログラムの記述



```
1 x = 0.1↓
2 n = 1↓
3 y = 0.3↓
4 if n==1: z=x+y↓
5 if n!=1: z=x*y↓
6 print( 'x=', x , 'y=', y , 'z=', z )↓
7 [EOF]
```

この部分を記述

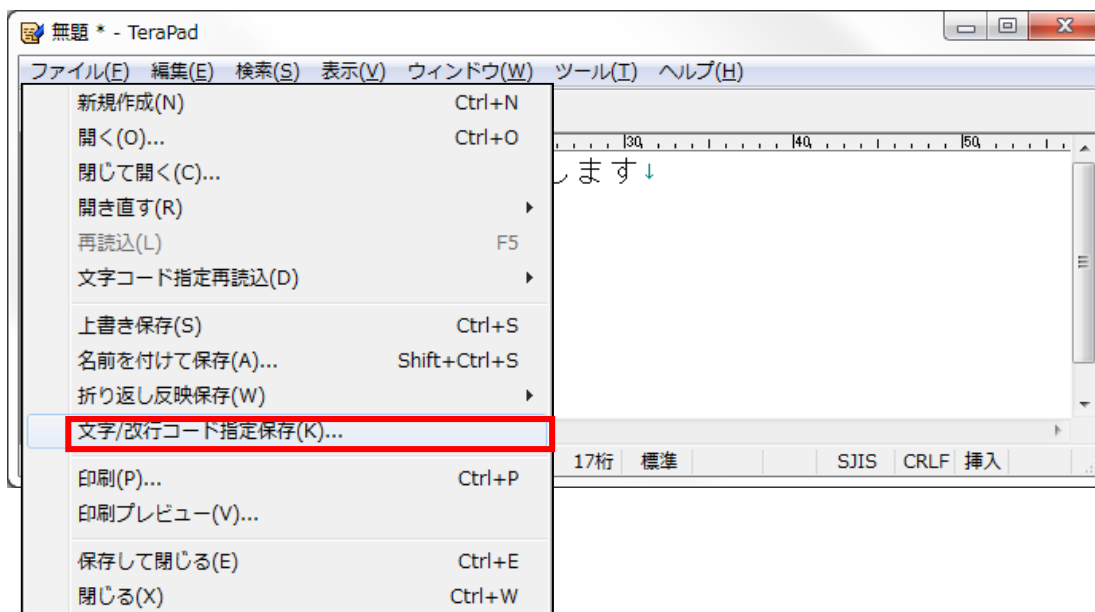
日本語以外は半角文字で書いて下さい
" " (ダブルクォート)は半角文字で書いて下さい

全角の空白は絶対に使わないで下さい

"
2 ふ

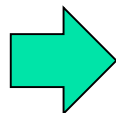
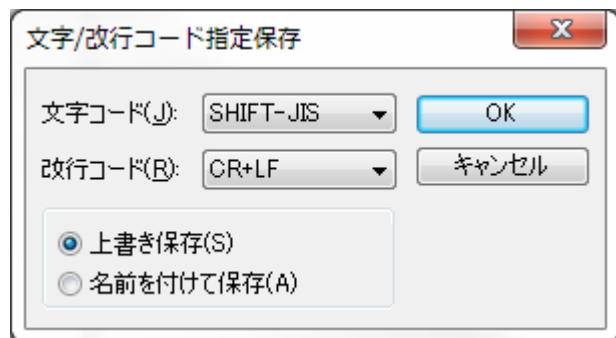
プログラムの保存①

- 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



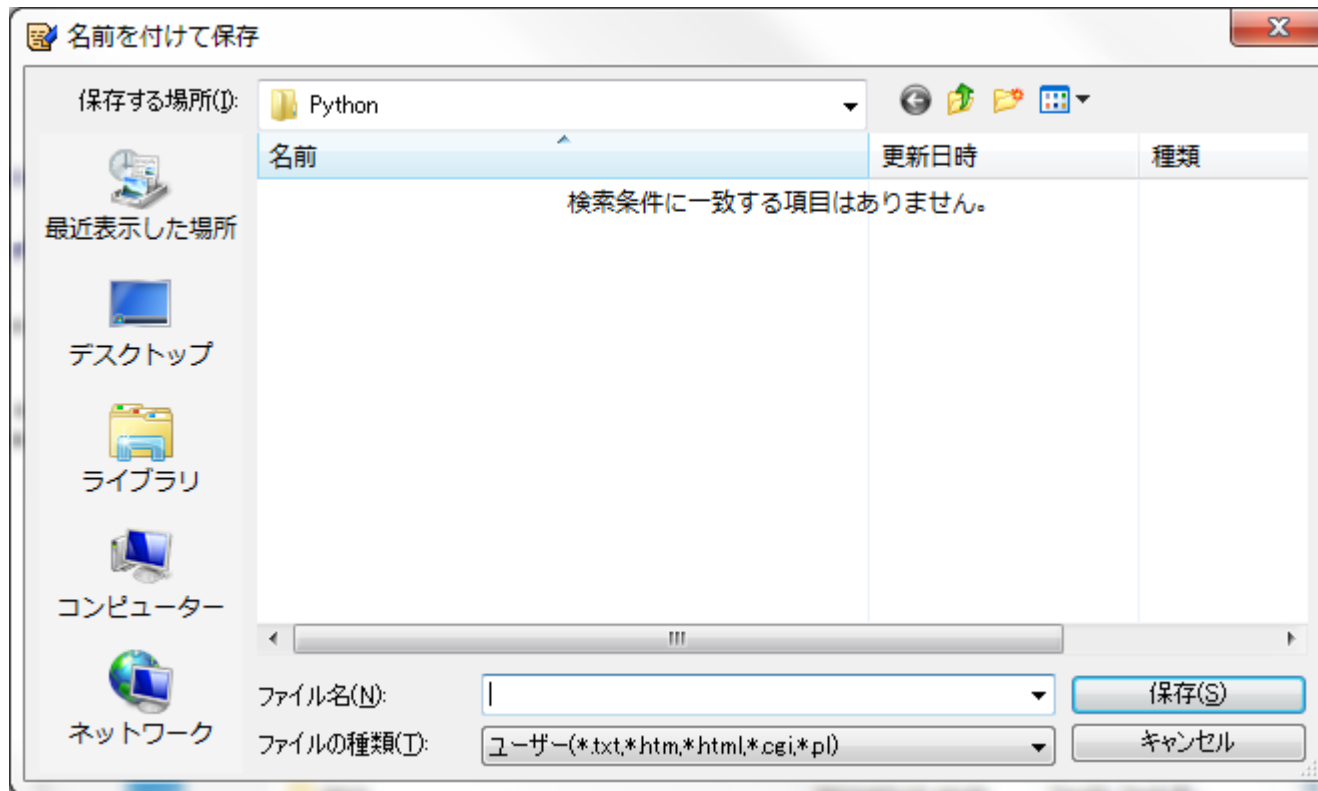
② 名前を付けて保存

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

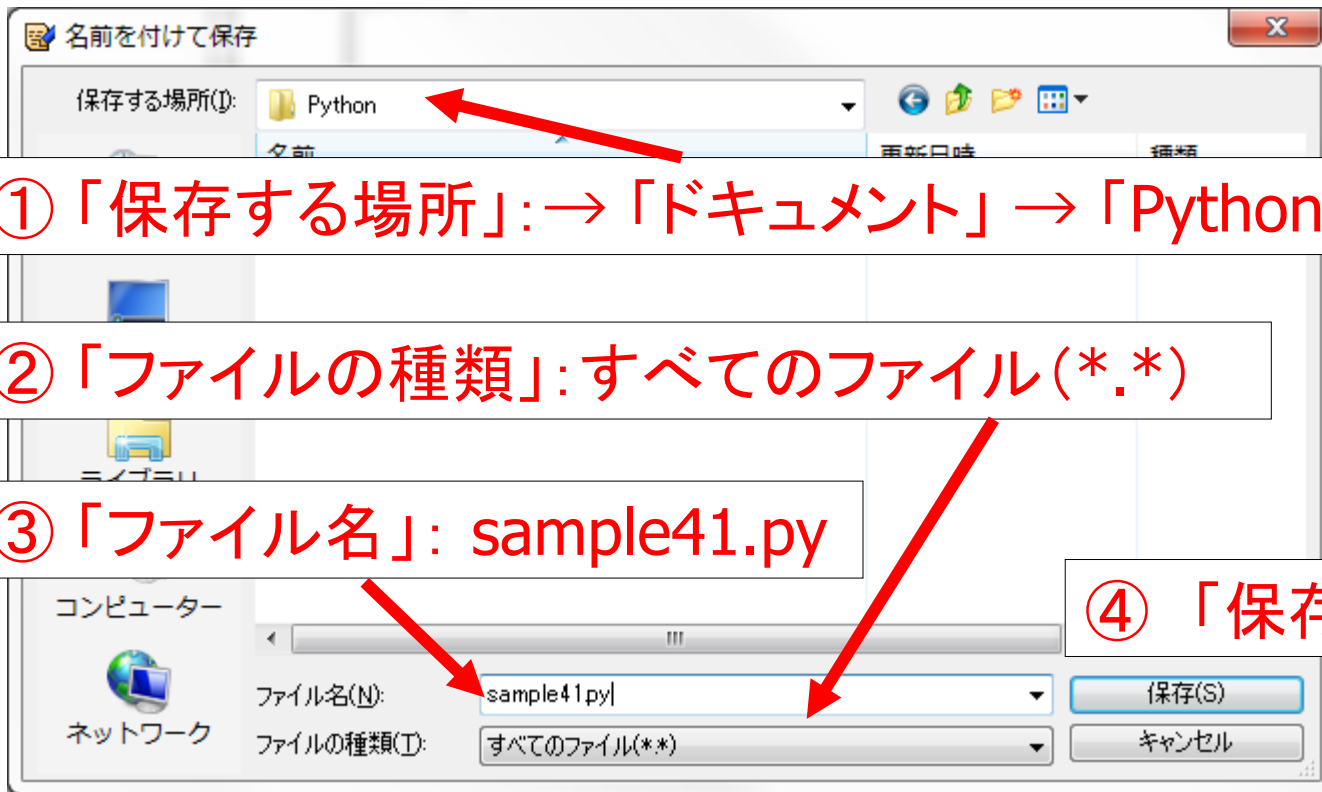
プログラムの保存③

- 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」



④ 「保存」をクリック



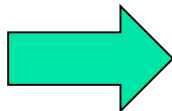
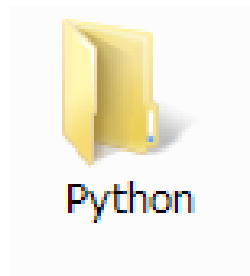
文字コード

- 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きます
- 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

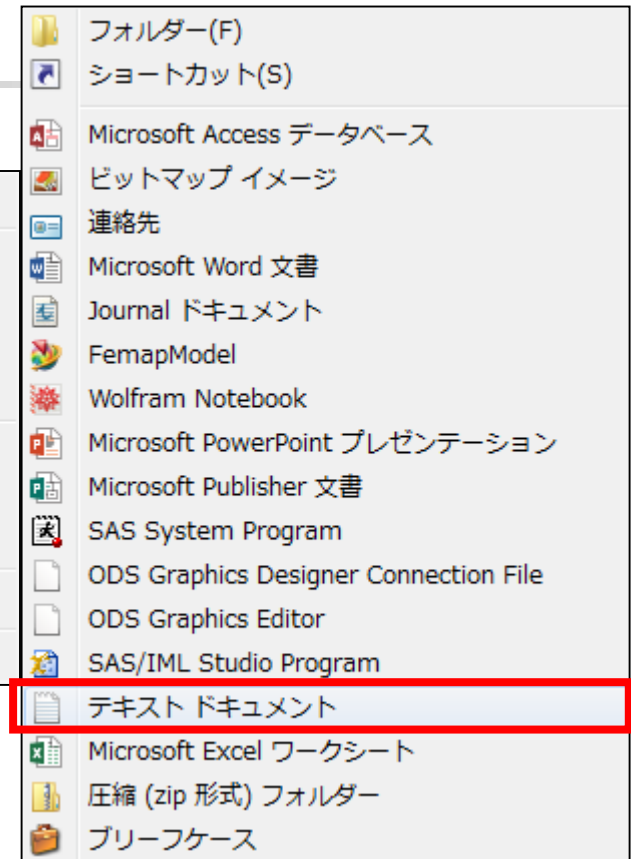
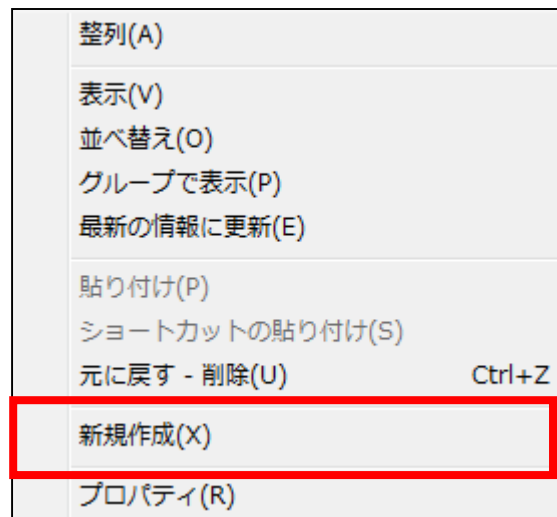
プログラムの書き方その② （「TeraPad」を用いる場合）



プログラムファイルの作成①

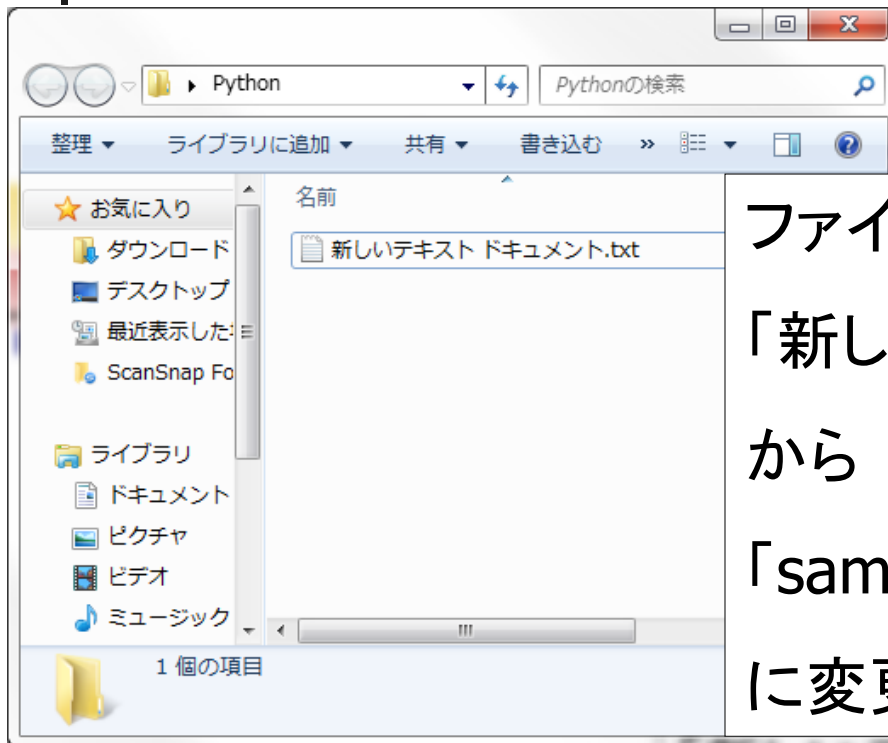


ダブルクリックして
「Python」フォルダ
を開く



「Ruby」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

「sample41.py」

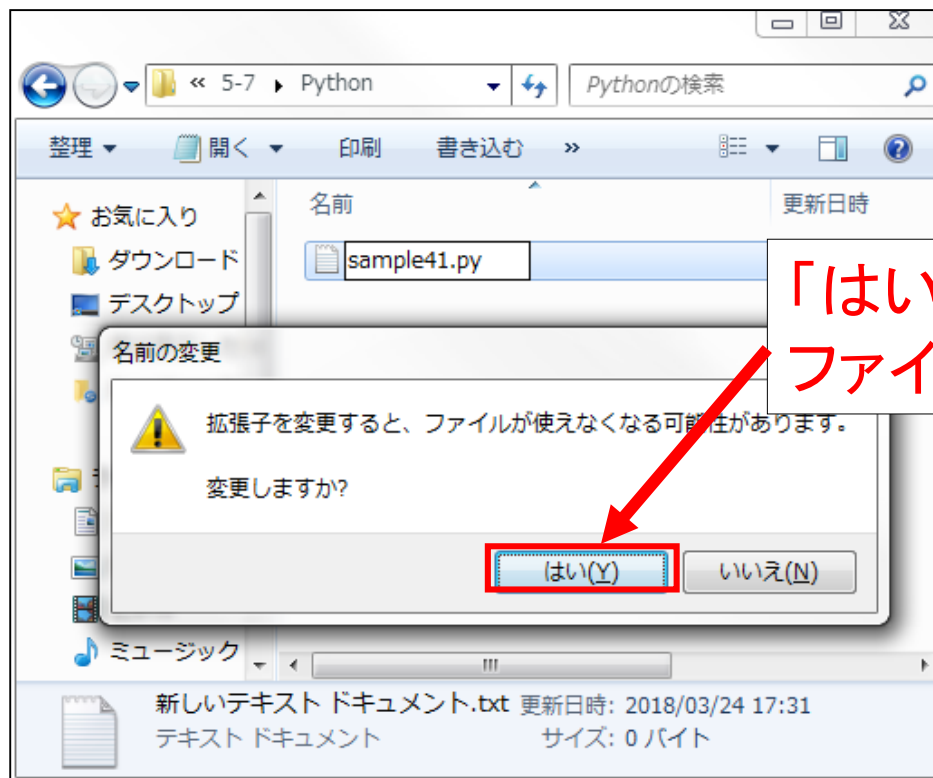
に変更する

「sample41.py」は半角文字として下さい

「sample41.py.txt」としないで下さい

プログラムファイルの作成③

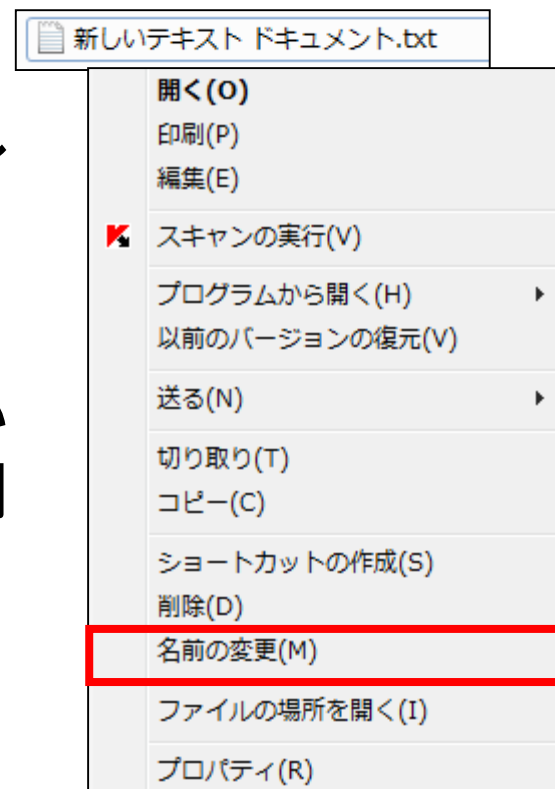
ファイル名を変更すると...



「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 **sample41.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)

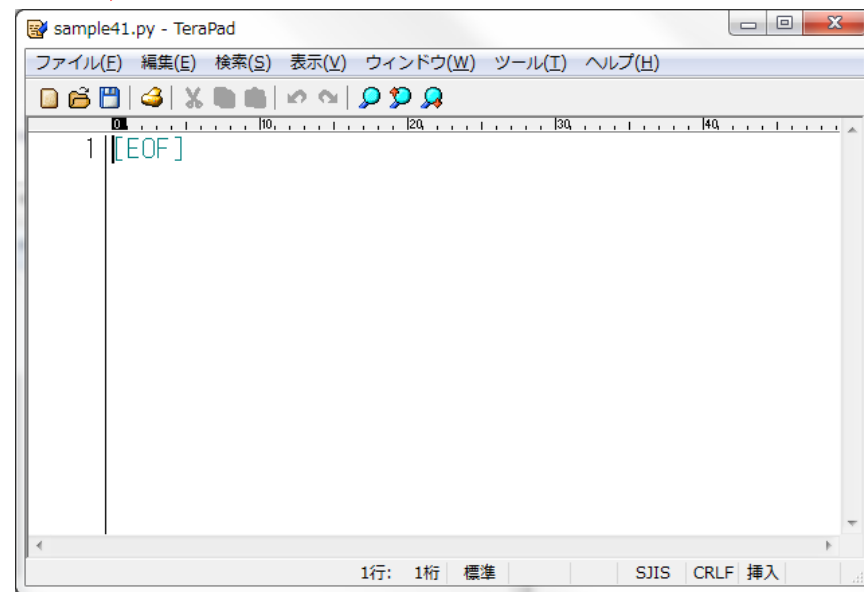


エディターの起動

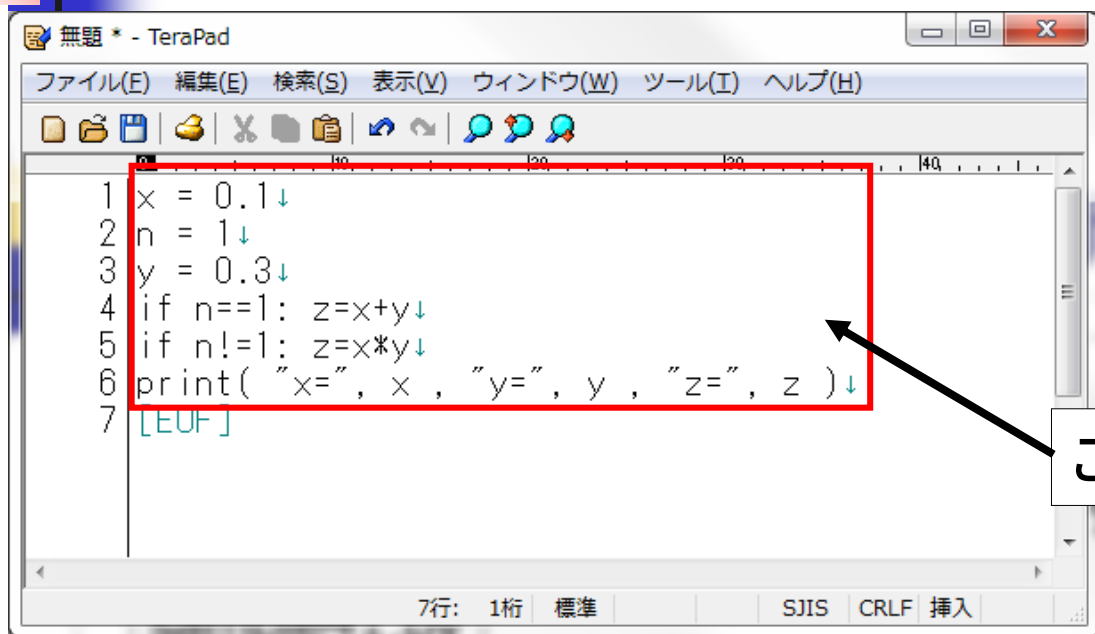
タイトルが「無題」から「sample41.py」
に変わる



TeraPadを起動 → 「sample41.py」
のアイコンをメモ帳にドラッグ



プログラムの記述



```
1 x = 0.1↓
2 n = 1↓
3 y = 0.3↓
4 if n==1: z=x+y↓
5 if n!=1: z=x*y↓
6 print( "x=", x , "y=", y , "z=", z )↓
7 [EOF]
```

この部分を記述

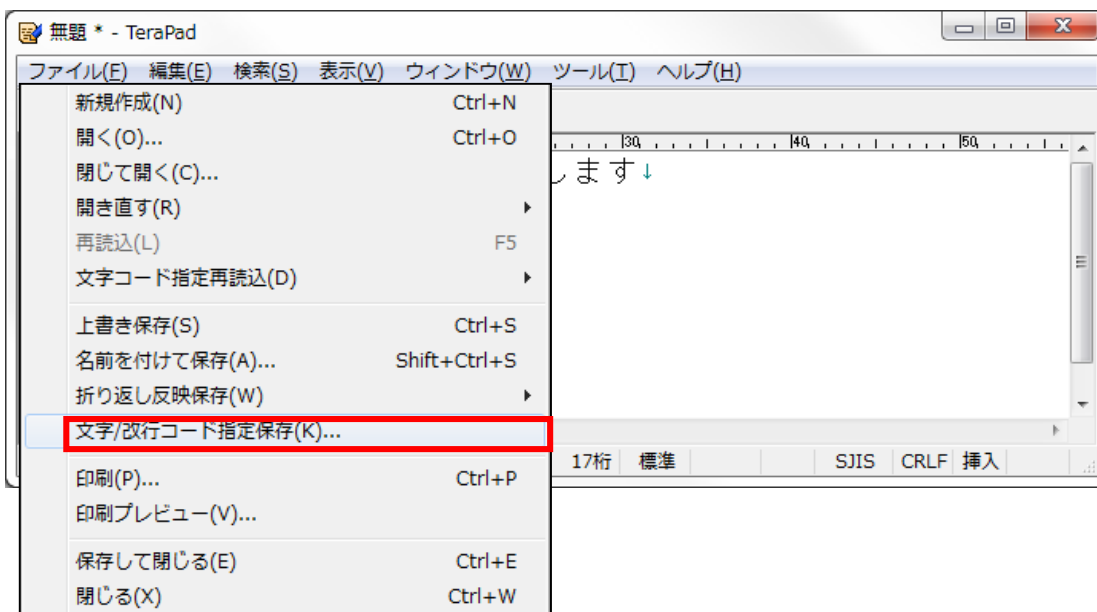
日本語以外は半角文字で書いて下さい
" " (ダブルクォート) は半角文字で書いて下さい

全角の空白は絶対に使わないで下さい

"
2 ふ

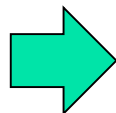
プログラムの保存①

- 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

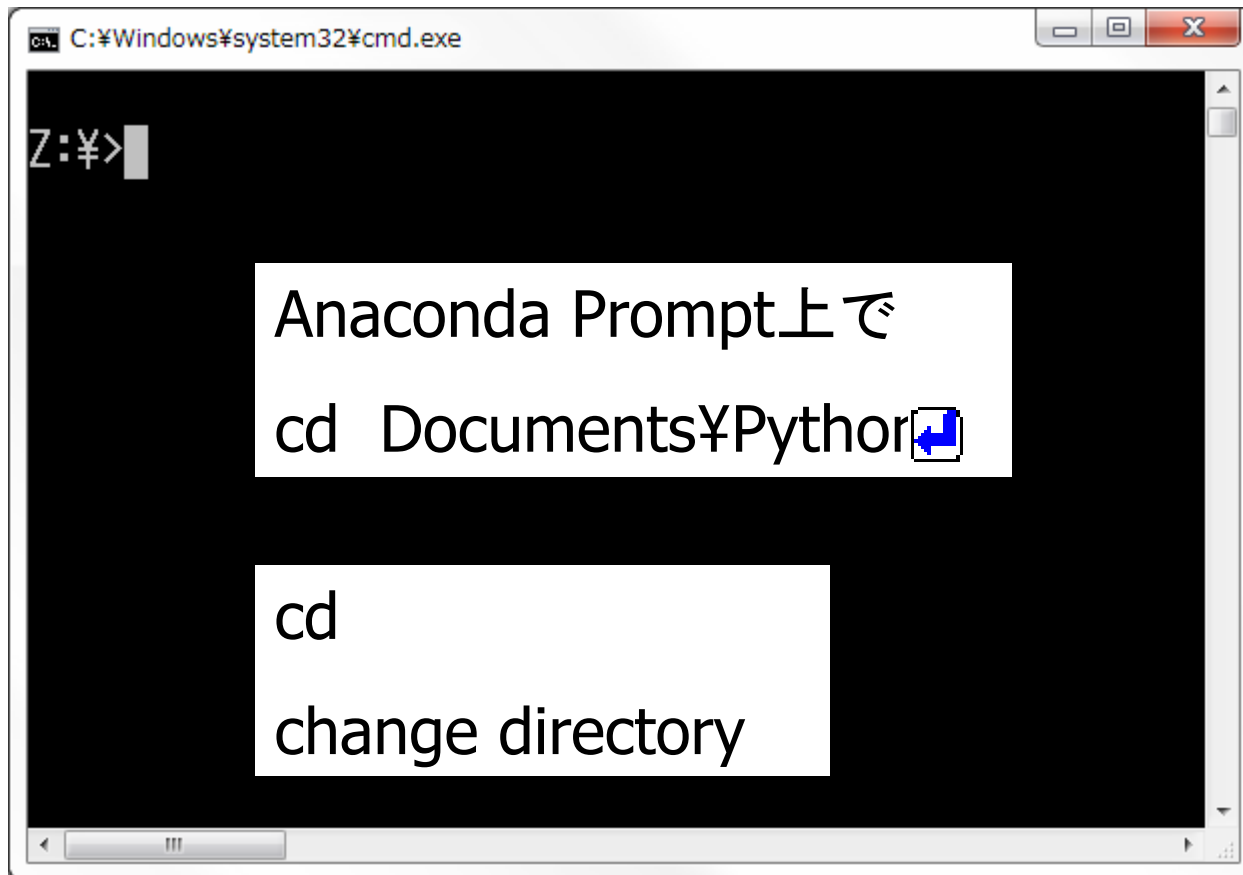
③ 「OK」をクリック

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

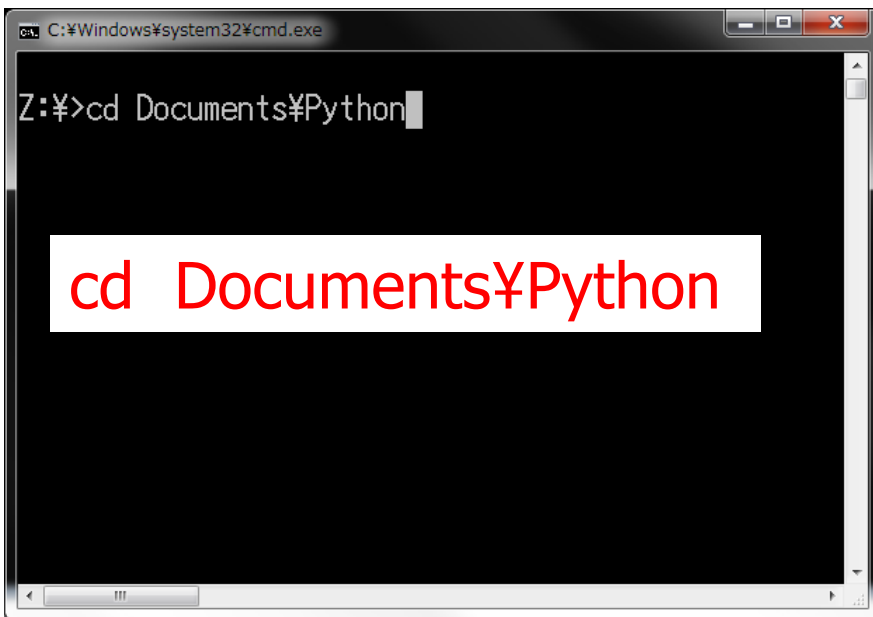


```
C:\Windows\system32\cmd.exe  
Z:\>  
cd Documents\Python  
cd  
change directory
```



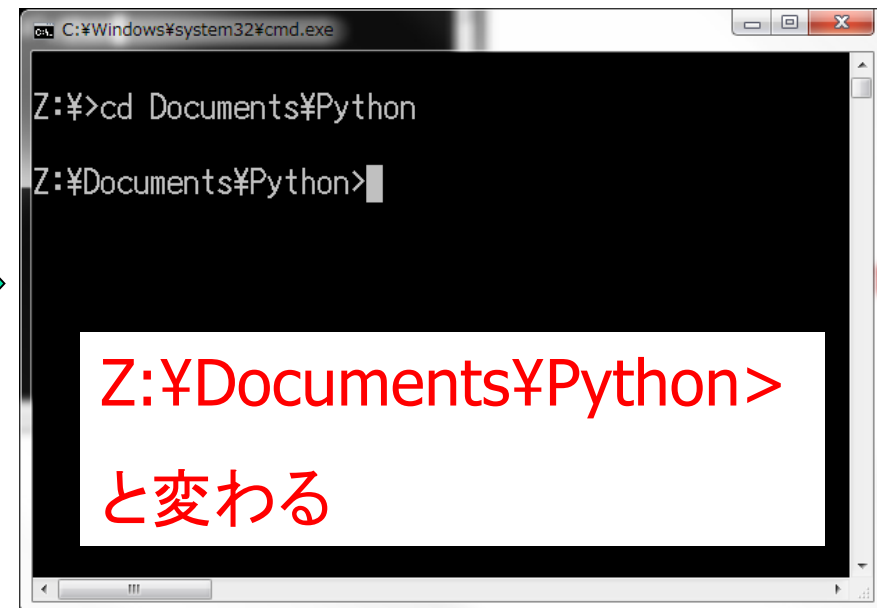
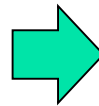
Enterキー

Pythonフォルダーへの移動②



```
C:\Windows\system32\cmd.exe
Z:\>cd Documents\Python
```

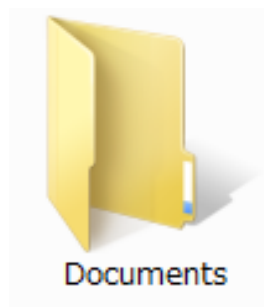
cd Documents\Python



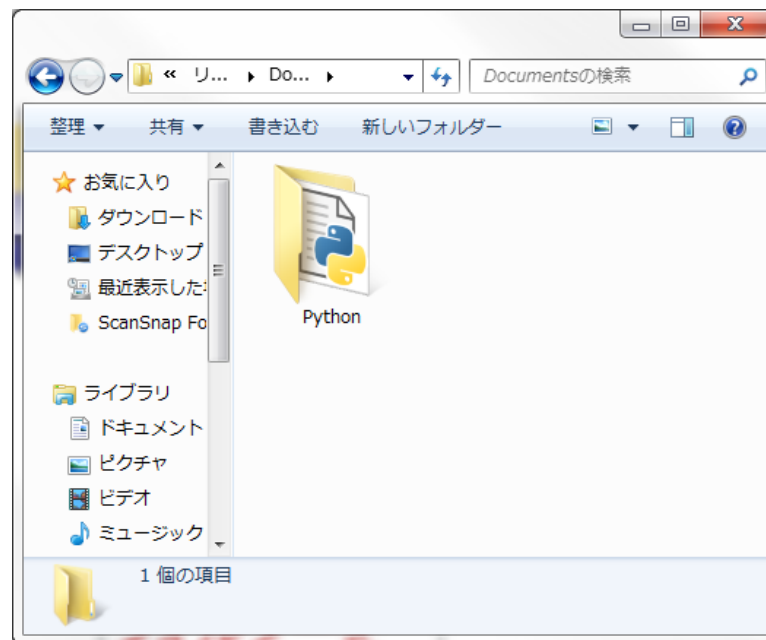
```
C:\Windows\system32\cmd.exe
Z:\>cd Documents\Python
Z:\Documents\Python>
```

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

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

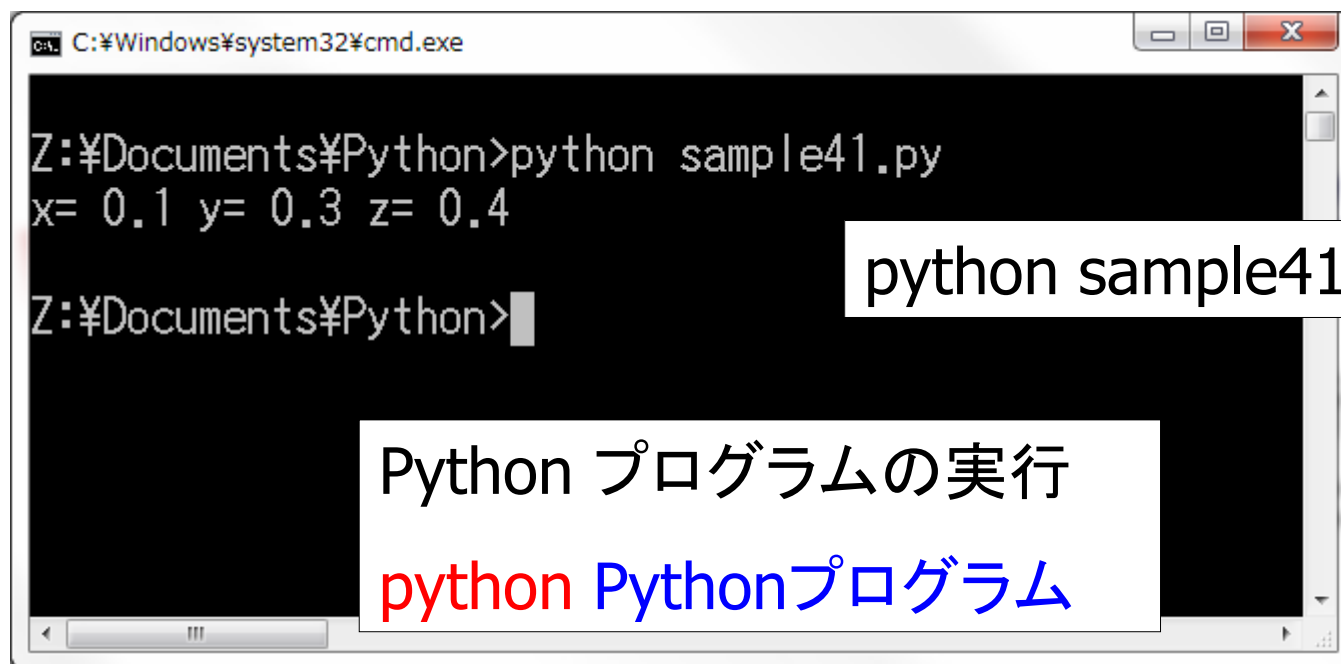
dir

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
```

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



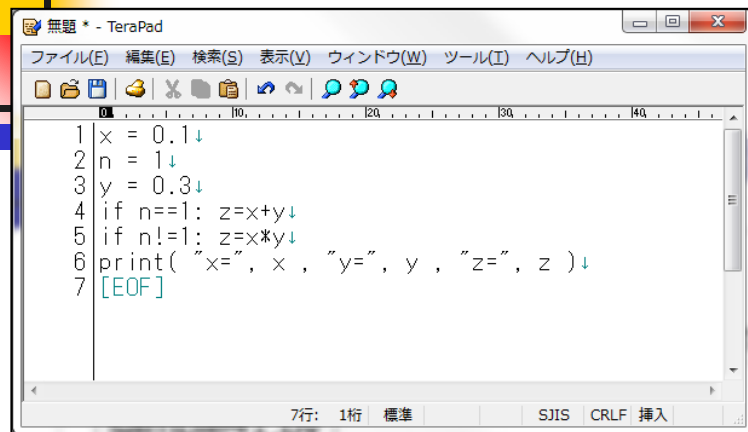
The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The command prompt shows the following text:

```
Z:\Documents\Python>python sample41.py  
x= 0.1 y= 0.3 z= 0.4  
Z:\Documents\Python>
```

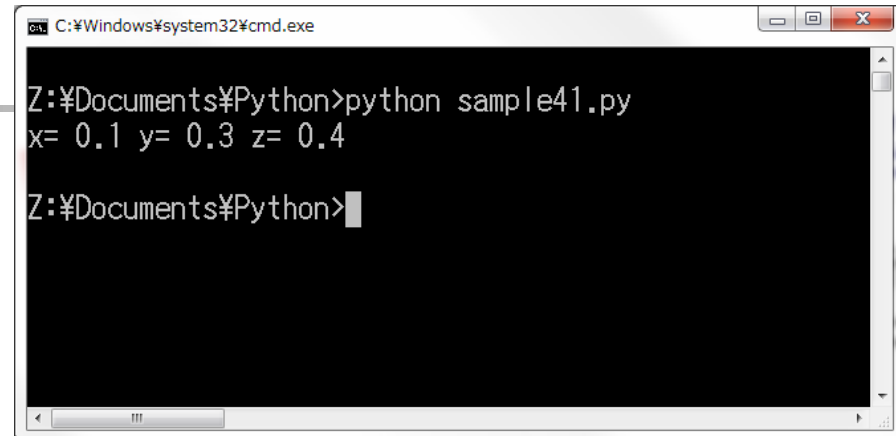
Two callout boxes are present:

- A box on the right contains the text `python sample41.py` with a blue arrow pointing to the command prompt.
- A box at the bottom contains the text "Python プログラムの実行" and "python Pythonプログラム", where "python" is red and "Pythonプログラム" is blue.

プログラミングと実行



```
1 x = 0.1↓
2 n = 1↓
3 y = 0.3↓
4 if n==1: z=x+y↓
5 if n!=1: z=x*y↓
6 print( "x=", x , "y=", y , "z=", z )↓
7 [EOF]
```



```
C:\Windows\system32\cmd.exe
Z:\Documents\Python>python sample41.py
x= 0.1 y= 0.3 z= 0.4
Z:\Documents\Python>
```

エディター(TeraPad)

プログラムを記述, 訂正

保存を忘れずに

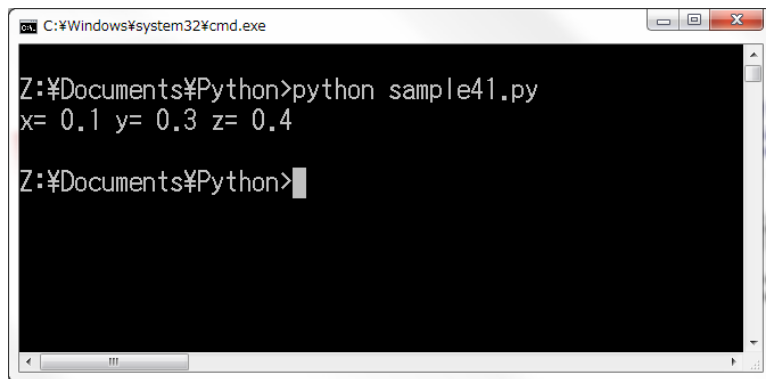
コマンドプロンプト

プログラムを実行 (python
コマンド)

エラーが
出た場合

予想通りに動かない場合
追加したい場合

コマンドプロンプトと対話型シェル



```
C:\Windows\system32\cmd.exe
Z:\Documents\Python>python sample41.py
x= 0.1 y= 0.3 z= 0.4
Z:\Documents\Python>
```

```
>>> x = 0.1
>>> n = 1
>>> y = 0.3
>>> if n==1: z=x+y

>>> if n!=1: z=x*y

>>> print( "x=", x , "y=", y , "z=", z )
x= 0.1 y= 0.3 z= 0.4
>>>
```

コマンドプロンプト

エディタで書いたプログラムを
pythonコマンドで全文実行

「python」を入力
対話型シェルの画面へ

対話型シェル

pythonのプログラムを一文ごと
に入力, 実行

「exit()」を入力
コマンドプロンプトの画面へ



プログラムが実行できない場合



プログラムが実行できない場合

- エラーメッセージを見て下さい
- 何行目にエラーがあるのか見つけて下さい
- どういう間違えであるのか, ヒントも表示されています
- 修正し, 再度実行して下さい(デバッグと言います)



うまく実行できない場合①

- うまく実行結果がでない場合
 - エラーメッセージが表示されるので見ること*

```
Z:¥Documents¥Python>python sample41.py
```

```
python: can't open file 'sample41.py': [Errno 2] No such file  
or directory
```

想定される原因:

ファイル sample41.rb がフォルダZ:¥Documents¥Pythonに存在しない

さらにその推定原因:

ファイル名または拡張子が違っている

別のフォルダにセーブした

→「dir」でファイルがあるかどうか確認してみる

*Pythonのバージョンによってエラーメッセージに違いがあります



うまく実行できない場合②

6行目に「invalid character」があるとエラーが表示

```
Z:¥Documents¥Python>python sample41.py
File "sample41.py", line 6
  print( "x=", x , "y=", y , "z=", z )
      ^
SyntaxError: invalid character in identifier
```

ここにエラーがあると指摘している

→ 半角空白なのに全角空白を使用してしまった

うまく実行できない場合③

5行目に「invalid syntax」とあるとエラーが表示

```
Z:¥Documents¥Python>python sample41.py
File "sample41.py", line 5
  if n!=1; z=x*y
      ^
SyntaxError: invalid syntax
```

ここにエラーがあると指摘している

→ 「:」なのに「;」を使用してしまった

うまく実行できない場合④

6行目に「invalid character」があるとエラーが表示

```
Z:¥Documents¥Python>python sample41.py
File "sample41.py", line 6
  print( "x=", x , "y=", y , "z=", z )
```

^
SyntaxError: invalid character in identifier

ここにエラーがあると指摘している

想定される原因:

実は1番目のダブルクオートが全角になっている

文字列は半角のダブルクオートで始まり、半角のダブルクオートで閉じる必要があります



うまく実行できない場合⑤

4行目に「invalid syntax」とあるとエラーが表示

```
Z:¥Documents¥Python>python sample41.py
File "sample41.py", line 4
    if n==1 z=x+y
           ^
SyntaxError: invalid syntax
```

ここにエラーがあると指摘している

想定される原因:
zの前に「:」が抜けている

うまく実行できない場合⑥

6行目に「invalid syntax」とあるとエラーが表示

```
Z:¥Documents¥Python>python sample41.py
```

```
File "sample41.py", line 6
```

```
    print( "x=", x "y=", y , "z=", z )
```

```
SyntaxError: invalid syntax
```

ここにエラーがあると指摘している

想定される原因:

x の後に「,」が抜けている



プログラムが実行できない場合

- エラーメッセージが表示されるので注目
 - 何行目にエラーが表示されているのか
 - invalid character → 不正な文字が含まれていないか
 - invalid syntax → 文法的に誤っていないか
- プログラムを修正した後, テキストエディタ上で上書き保存を行ない, 再実行する



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: {] { ([]] }
- ダブルクオートも括弧の一種です. 文字列を表します. 普通の英語でもそうですが, 開いたら必ず閉じる必要があります
 - 例: "this is a book", "これはOK) } { "



文と式の復習

sample41.py の説明



sample41.py の説明

```
x = 0.1
n = 1
y = 0.3
if n==1: z=x+y
if n!=1: z=x*y
print( "x=", x , "y=", y , "z=", z )
```

代入文

条件文

print文(標準出力)



変数と型

■ 変数:

- コンピュータにおける変数とは、まず第一に、データを一時的に記憶しておく**場所**です
- そして、場所を区別するために**名前**をつけます
- そして、データには**型**があります

■ 型:

- 許される演算によって決まります
- これまでに出てきたのは、整数、小数(浮動小数点数)、文字列、論理型(True, False)です



データ型の変換①

- 整数に変換

`int(3.1415)`

`int("3")`

`int("3") + 5`

整数へ変換

`int(値)`



データ型の変換②

- 小数に変換

`float(3)`

`float("3.1415")`

`float("3")`

`float("3.1415") * 2.5`

小数へ変換

`float(値)`



データ型の変換③

- 文字列に変換

`str(3)`

`str(3.1415)`

`str(3) + "5"`

`str(3.1415) * 2`

文字列へ変換

`str(値)`



自動型変換

- 整数型と小数が混在しているときには、小数に変換される.
 - これは、演算ごとに行われる. 演算は左から順番に行われるため、すべての整数型データが小数に変換されるわけではない
 - 小数から整数型への変換はintを用いる必要がある

```
>>> 3*2.5+3//4
7.5
>>> 3*int(2.5)+3//4
6
```

- 文字列との自動変換は行われない

```
>>> "2"+3
Traceback (most recent call last):
  File "<pyshe11#20>", line 1, in <module>
    "2"+3
TypeError: must be str, not int
```

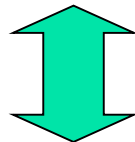



文に関する注意①

- python において「文」の意味する範囲は広い
 - $y = 2 * x + 3$
 - $x = y + x$ #注意 $y+x$ を x に代入すること
 - `print(x)`
 - `if x==3: y = x ; print(y)` #「;」で文をつなげる
- さらには, 複数行にまたがる(またいで書いた方が分かりやすい)場合も, よく, ある.

文に関する注意②

```
if x==3: y = x; print( y )
```



同じ条件文

```
if x==3:
```

```
    y = x  
    print( x )
```

複数行にまたいで書いた場合

この空白をインデントと呼びます. pythonのプログラムではこのインデントが非常に重要です



算術演算子

演算子	用途	例	演算結果
+	加減	3+2	5
-	減算	4-2	2
*	乗算	2*2	4
/	除算	4/2	2.0
//	除算	4//2	2
%	剰余	5%2	1
**	冪	5**3	125

「/」は小数点以下を切り捨てない 「//」は小数点以下を切り捨てる



浮動小数点数型の算術演算子

演算子	用途	例	演算結果
+	加減	3.1+2.2	5.3
-	減算	4.2-2.1	2.1
*	乗算	2.1*2.1	4.41
/	除算	4.5/3.0	1.5
//	除算	4.5//3.0	1.0
%	剰余	5.0%2.1	0.8
**	冪	2.1**0.5	1.44913



代入演算子

- 「a = a 演算子 b」を「a 演算子 = b」と記述することができる
- これを代入演算子という
 - 注意 =と演算子の間にスペースはおけない

a=20; b=10

a=a+b ⇔ a+=b # aは30

a=a-b ⇔ a-=b # aは20

a=a*b ⇔ a*=b # aは200



代入演算子

演算子	用途	例	演算結果
=	代入	$x = 4.3$	$x \leftarrow 4.3$
+=	加減後代入	$x += 3.1$	$x \leftarrow x + 3.1$
-=	減算後代入	$x -= 3$	$x \leftarrow x - 3$
*=	乗算後代入	$x *= 3$	$x \leftarrow x * 3$
/=	除算後代入	$x /= 3$	$x \leftarrow x / 3$
//=	除算後代入	$x //= 3$	$x \leftarrow x // 3$
%=	剰余の代入	$x \% = 3$	$x \leftarrow x \% 3$
**=	冪の代入	$x ** = 3$	$x \leftarrow x ** 3$

演算子の優先順位

高
↑
↓
低

(): 括弧, (): 引数

** : 冪乗

+, - : 符号 (符号同一, 符号反転)

* : 乗算 / : 除算 % : 剰余

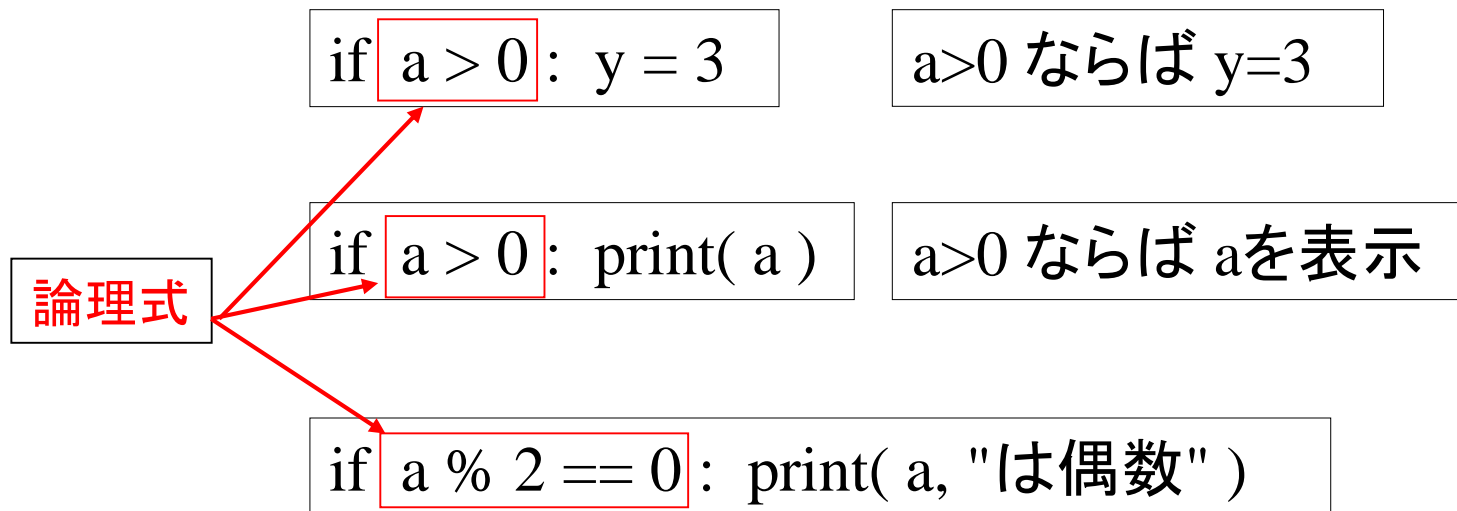
+ : 加算 - : 減算

= : 代入

- 同じレベルの演算子は左から順に計算する. 従って,
 - $2-4*3$ は $2-(4*3)$, $3/4*6$ は $(3/4)*6$
 - なお, $-a**-b$ は $-(a**(-b))$ (**が優先)
 - $a--+-b$ は $a-(-(+(-b)))$ (-と+は同じ)

条件文①

- 「if 論理式: 文」という文がある
- 論理式がTrueならば文を実行, Falseならば文を実行しない



aが2で割り切れる場合, 偶数と表示



比較演算子

演算子	用途	例	演算結果
==	等	3==2	False
>	大	4 > 2	True
<	小	4 < 2	False
>=	大 or 等	4>=2	True
<=	小 or 等	4<=2	False
!=	非等	3!=2	True
in	含まれる	"x" in "xyz"	True



論理演算子

演算子	用途	例	演算結果
not	否定	not 3==2	True
and	かつ	2==2 and 4>2	True
or	または	2==3 or 4>2	True



条件文②

```
if n==1 : z=x+y
```

nが1の場合

論理式

```
if n!=1 : z=x*y
```

nが1でない場合

```
x=5; y=3; n=1
```

```
>python sample.py  
x= 5 y= 3 z= 8
```

```
x=5; y=3; n=10
```



```
>python sample.py  
x= 5 y= 3 z= 15
```



条件文③

nが2で割り切れる, かつnが3で割り切れるかどうか

```
n=9
if n % 2 == 0 and n % 3 == 0 : n+=2
if n % 2 == 0 or n % 3 == 0 : n*=2
print( " n=" , n )
```



nが2で割り切れる, またはnが3で割り切れるかどうか

```
>python sample.py
n= 18
```



標準出力

print文を用いた出力



標準出力 (print文)

■ `x=3`

① `print("x=" , x , "です")`

② `print("x=" + str(x) + "です")`

③ `print("x=は{0}です".format(x))`



出力文①

- `print(変数)`
- `print("コメント")`
 - コメント(文字列)を表示する場合は" "で囲む
- `print("コメント", 変数)`
- `print(変数1, 変数2, ..., 変数n)`
 - 変数, コメントを一行で表示したい場合は,「,」で区切る

出力文②

```
>>> x=3.1415
>>> print( "x=" , x )
x= 3.1415
```

文字列

変数

改行される

「,」で区切った変数と変数,
文字列の間は空白が入る

変数と変数, 文字列は「,」で区切る

出力文③

```
>>> x=3.1415
>>> y=2
>>> print( "x=" , x , "y=" , y )
x= 3.1415 y= 2
```

文字列

文字列

変数

変数

改行される

変数と変数, 文字列は「,」で区切る

「,」で区切った変数と変数,
文字列の間は空白が入る

出力文④

```
>>> print( "x" x )  
SyntaxError: invalid syntax
```

「,」がないとエラーが出力

```
>>> print(x)  
Traceback (most recent call last):  
  File "<pyshell#0>", line 1, in <module>  
    print(x)  
NameError: name 'x' is not defined
```

x の値を設定(宣言)していないため
エラーが生じた

出力文⑤(区切り文字の変更)

```
>>> x="abc"  
>>> y="xyz"  
>>> z="1234"  
>>> print( x , y , z )  
abc xyz 1234
```

「,」で区切った変数と変数, 文字列の間は空白が入る

```
>>> print( x , y , z , sep="-" )  
abc-xyz-1234
```

区切り文字を「-」に変更

出力文⑥(区切り文字の変更)

```
>>> print( x , y , z , sep="" )  
abcxyz1234
```

区切り文字を「なし」に変更

```
>>> print( x , y , z , sep="¥n" )  
abc  
xyz  
1234
```

¥n
改行コード

区切り文字を「改行」に変更

出力文⑦(文末改行の変更)

```
>>> x=3.141
>>> print( "x=" , x )
x=3.141
```

改行し, 次の行に移動

```
>>> print( "x=" , x , end="¥n¥n" )
x=3.141
```

¥n
改行コード

二回改行される

出力文⑧(文末改行の変更)

```
>>> x=3.141
>>> y=2
>>> print( "x=" , x ) ; print( "y=" , y )
x=3.141
y=2
```

改行し、次の行に移動

```
>>> x=3.141
>>> y=2
>>> print( "x=" , x , end="" ) ; print( "y=" , y )
x=3.141y=2
```

改行せずに、次のprint文を出力

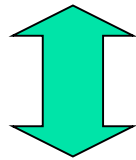


文字列連結演算子による出力

- `print("コメント")`
- "コメント"は文字列型の値
- 文字列連結演算子
 - 「+」は, 文字列を連結する役割がある
 - 「*」は, 文字列を繰り返す役割がある

文字列連結演算子①

- `i=3`
- `print("iは ", i, " です")`



- `print("iは " + str(i) + " です")`
 - " "で囲まれた文字列または変数の文字列値を「+」演算子を用いて連結し、一つの文字列として出力する
 - 整数(小数)を文字列とするには、strを用いて変換する
- `print("にわ" *4 + "とりがいる")`

文字列連結演算子②

```
>>> x=3.1415
>>> print( "x=" , x )
x= 3.1415
```

文字列連結演算子を用いた場合

```
>>> x=3.1415
>>> print( "x=" + str( x ) )
x= 3.1415
```

空白あり

"x=" + str(x)

小数 x を文字列に変換し + で連結



文字列連結演算子③

```
>>> x=3.1415
>>> y=2
>>> print( "x=" , x , "y=" , y )
x= 3.1415 y= 2
```

文字列連結演算子を用いた場合

```
>>> x=3.1415
>>> y=2
>>> print( "x= " + str( x ) + " y= " + str(y) )
x= 3.1415 y= 2
```



文字列連結演算子④

```
>>> x="abc"
>>> y="xyz"
>>> print( "x=" , x , "y=" , y )
x= abc y= xyz
```

文字列連結演算子を用いた場合

```
>>> x="abc"
>>> y="xyz"
>>> print( "x= " + x + " y= " + y )
x= abc y= xyz
```

変数x, yともに文字列型

文字列連結演算子⑤

```
>>> x="3.1415"
>>> print( "x=" , x )
x= 3.1415
>>> print( "x= " + x )
x= 3.1415
>>> print( "x=" , float(x) )
x= 3.1415
>>> print( "x= " + float(x) )
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    print( "x= " + float(x) )
TypeError: must be str, not float
```

+演算子
次の変数は文字列型のみ

format文①

```
>>> n=123
```

```
>>> print( "nの値は{0}です".format(n) )
```

```
nの値は123です
```

```
>>> m=456
```

```
>>> print( "nの値は{0}, mの値は{1}です".format(n,m) )
```

```
nの値は123, mの値は456です
```

{0}の位置に変数nを出力

{0}の位置に変数n, {1}の位置に変数m

nは{0}, mは{1}に対応

- format文では文字列中に, 出力したい変数を指定できる
- 上記のように, 文字列中に出力したい変数の位置を{}で指定する
- format文中に, 出力したい変数を並べて書く

format文②

```
>>> n=123
>>> m=456
>>> print( "nの値は{0}, mの値は{1}です".format(n,m) )
nの値は123, mの値は456です
```



nは{0}, mは{1}に対応

```
>>> n=123
>>> m=456
>>> print( "mの値は{1}, nの値は{0}です".format(n,m) )
mの値は456, nの値は123です
```



format文③

```
>>> n=123
>>> msg = "{0}の2倍は{1}, {0}の3倍は{2}".format( n , 2*n , 3*n )
>>> print( msg )
123の2倍は246, 123の3倍は369
```



n は{0}, $2*n$ は{1}, $3*n$ は{2}に対応

```
>>> n=123
>>> msg = "{0}の3倍は{2}, {0}の2倍は{1}".format( n , 2*n , 3*n )
>>> print( msg )
123の3倍は369, 123の2倍は246
```



format文④

```
>>> n=123
>>> m=456
>>> print( "nの値は{0}, mの値は{1}です".format(n,m) )
nの値は123, mの値は456です
```



```
>>> print( "nの値は{a}, mの値は{b}です".format(a=n,b=m))
nの値は123, mの値は456です
```

{0}{1}の代わりに変数名もつけられます
→{a}はn, {b}はmに対応



format文⑤

```
>>> n=123
>>> msg="{a}の2倍は{b}, {a}の3倍は{c}".format(a=n,b=2*n,c=3*n)
>>> print(msg)
123の2倍は246, 123の3倍は369
```



{a}はn, {b}は2*n, {c}は3*nに対応

```
>>> n=123
>>> msg="{a}の3倍は{c}, {a}の2倍は{b}".format(a=n,b=2*n,c=3*n)
>>> print( msg )
123の3倍は369, 123の2倍は246
```



改行①

- `print("x=" , x)`
- `print("x=" , x , "¥n")`

```
>>> x=2
>>> print( "x=" , x )
x= 2
>>> print( "x=" , x , "¥n" )
x= 2
>>>
```

一回改行

二回改行



改行②

- 「¥n」
 - 改行文字
 - 一行改行される

```
>>> x=2; y=3
>>> print( "x=" , x , "y=" , y )
x= 2 y= 3
>>> print( "x=" , x , "¥n y=" , y )
x= 2
y= 3
```

改行される



標準入力

input文による入力

キーボードから入力できれば...

```
x=5  
y=3  
print( "x={0}, y={1}, x+y={2}".format(x,y,x+y) )
```

x=15 , y=30にしたい



```
> python sample.py  
x=5, y=3, x+y=8
```

x=15

y=30

書き直して再度実行

```
print( "x={0}, y={1}, x+y={2}".format(x,y,x+y) )
```

```
> python sample.py  
x=15, y=30, x+y=35
```



キーボードから入力できれば...

x=???

y=???

```
print( "x={0}, y={1}, x+y={2}".format(x,y,x+y) )
```

実行時に、キーボードから値を入力したい

標準入力①

- キーボードからの入力
- input()

```
>>> input()  
34  
'34'
```

入力

```
> input()  
abcd  
'abcd'
```

- ① input()と打つ
- ② キーボードから入力
(最後に改行する)

入力した値は文字列として処理される



標準入力②

入力

```
>>> a=input()  
3.1415  
>>> a  
'3.1415'
```

```
>>> x=input()  
abcd  
>>> a  
'abcd'
```

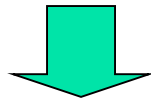
変数a に入力した値を代入
変数a は文字列型

変数x に入力した値を代入
変数x は文字列型



標準入力③

- input()による標準入力
- キーボードより入力した値は文字列型として扱われる



- 整数値(小数値)として利用したい場合
- 文字列型から整数(小数)へ変換する必要がある

標準入力④

入力

```
>>> a=input()
```

```
3.1415
```

aは文字列

```
>>> a
```

```
'3.1415'
```

```
>>> float( a )
```

文字列型を小数に変換

```
3.1415
```

```
>>> int( float( a ) )
```

```
3
```

文字列型を整数に変換
(文字列→小数→整数)



プログラム中で入力する

- 2個数値を入力し, 和を求めるプログラムです

sample42.py

```
line = input( "一番目の数値を入力してください: " )  
x1 = float( line )  
line = input( "二番目の数値を入力してください: " )  
x2 = float( line )  
print( "{0} + {1} = {2}".format( x1 ,x2 ,x1+x2 ) )
```

```
Z:¥Documets¥Python>python sample42.py  
一番目の数値を入力してください: 123  
二番目の数値を入力してください: 456  
123.0 + 456.0 = 579.0
```

input文の工夫

```
>>> print( "aの値を入力して下さい" )
```

aの値を入力して下さい

```
>>> a = input()
```

1234

```
>>>
```



```
>>> a = input("aの値を入力して下さい")
```

aの値を入力して下さい1234

```
>>>
```

"コメント"が表示され、入力待ち状態となる



標準入力⑤

```
line = input( "整数を入力して下さい" )  
x = int( line )  
print( x )
```

整数に変換し, 表示

```
line = input( "小数を入力して下さい" )  
x = float( line )  
print( x )
```

小数に変換し, 表示

```
line = input( "文字列を入力して下さい" )  
print( line )
```

文字列をそのまま表示



キーボードからの入力例①

```
a = int( input( "整数a? " ) )  
b = int( input( "整数b? " ) )  
c = int( input( "整数c? " ) )  
average = (a+b+c)/3  
print( "平均は" , average , "です" )
```

3個の整数を読み込んで
平均を出力

```
>python sample.py  
整数a? 34  
整数b? 5  
整数c? 56  
平均は 31.666666666666668 です
```



キーボードからの入力例②

```
a = input( "文字列a? " )  
b = input( "文字列b? " )
```

2個の文字列を読み込んで比較

```
if a == b: print( a , "と" , b , "は同じです" )  
if a != b: print( a , "と" , b , "は違います" )
```

```
>python sample.py  
文字列a? abc  
文字列b? xyz  
abc と xyz は違います
```



キーボードからの入力例③

```
x = int( input( " x?¥n " ) )
y = int( input( " y?¥n " ) )
if x == y: print( " x と y は同じ値です " )
if x != y: print( " x と y は異なる値です " )
```

整数型で入力

```
>python sample.py
x?
25
y?
15
x と y は異なる値です
```




キーボードからの入力例④

長方形の面積を求めるプログラム

```
x = int( input( "横の長さ?¥n" ) )  
y = int( input( "縦の長さ?¥n" ) )  
s = x * y  
print( "横{0}, 縦{1}の長さの面積は{2}".format( x , y , s ) )
```

整数型で入力

```
>python sample.py  
横の長さ?  
10  
縦の長さ?  
4  
横10, 縦4の長さの面積は40
```



キーボードからの入力例⑤

消費税を求めるプログラム

```
price = int( input( "商品の値段?¥n" ) )
tax = int( price * 0.08 )
total = price + tax
print( "{0}円の消費税は{1}円, 税込み価格は{2}円".format(
price , tax , total ) )
```

整数型で入力

整数型で計算

```
>python sample.py
```

```
商品の値段?
```

```
2000
```

```
2000円の消費税は160円, 税込み価格は2160円
```



練習問題

練習①～④を行なって下さい
(時間があれば⑤⑥も行なって下さい)

練習問題①

- 以下, print文を用いて, 下記の実行結果のように表示しなさい.

```
x=10  
y=5
```

以下print文を用いて, 右の
実行結果のように表示しなさい

```
> python 4-1.py  
x= 10 y= 5  
x+y= 15  
x-y= 5 x*y= 50  
x/y=  
2.0  
x%y=0
```

「-」が10回



練習問題②

- 半径20の円において、円周および面積を小数値として求め、表示するプログラムを書きなさい。

```
>python 4-2.py
```

```
半径 20 の円周は 125.66370614359172 で面積は 1256.6370614359173 です
```

以下の問題について、画面への表示はこの通り
でなくてもけっこうです



練習問題③

- 円の半径を整数値としてキーボードから入力し, 円周および面積を小数値として求め, 表示するプログラムを書きなさい.

```
> python 4-3.py
```

```
円の半径は？
```

```
20
```

```
半径 20 の円周は 125.66370614359172 で面積は 1256.6370614359173 です
```



練習問題④

- 整数 x, y をキーボードから入力し, 大小を判定するプログラムを書きなさい

```
>python 4-4.py  
x? 20  
y? 10  
20 は 10 以上です
```

```
>python 4-4.py  
x? 20  
y? 100  
20 は 100 より小さい
```



練習問題⑤

- x円を年利r%でn年借りた場合の金額を印字するプログラムを書きなさい. x, r, nは整数でキーボードから入力しなさい.

```
>python 4-5.py  
金額? > 10000  
年利? > 5  
年? > 5  
12762 円
```




練習問題⑥

- 小数xをキーボードから入力しなさい。小数xの整数部，小数部を別々に表示しなさい。

```
>python 4-6.py  
小数? > 3.4444  
整数部 3  
小数部 0.444399999999999999
```



気にしないで下さい

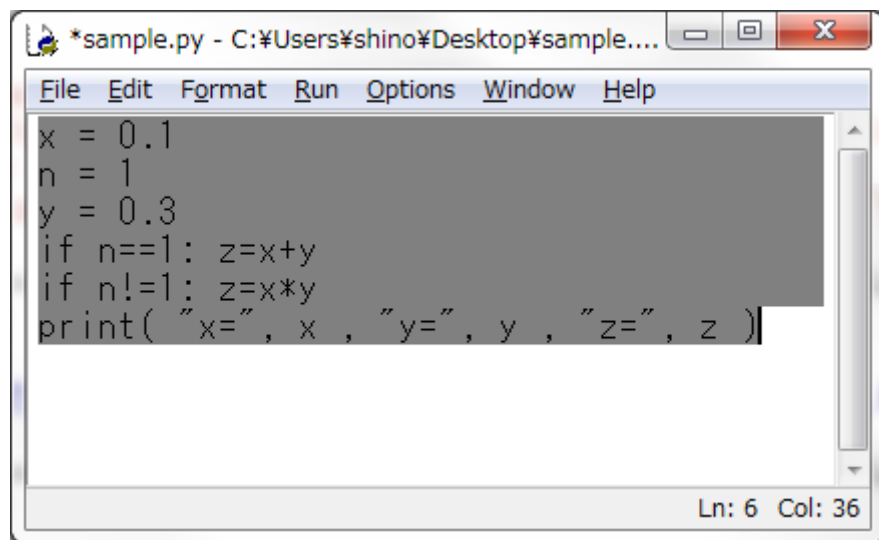


提出方法

- プログラムと実行結果の画面をMS-Wordファイルにまとめて下さい
- keio.jp 上から作成したファイルを提出して下さい

プログラムと実行結果をMS-Word への貼り付け方①

① エディター上にてプログラムを選択



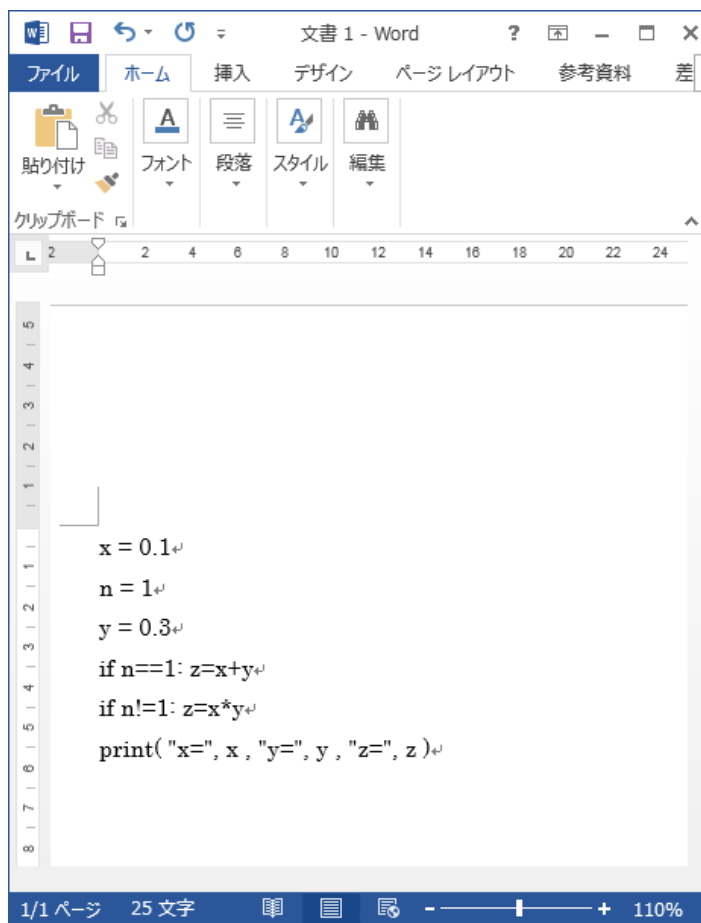
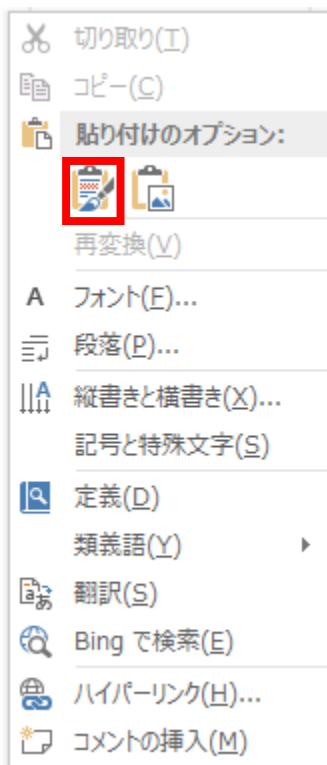
```
x = 0.1
n = 1
y = 0.3
if n==1: z=x+y
if n!=1: z=x*y
print("x=", x, "y=", y, "z=", z )
```

② 右クリック→「コピー」

元に戻す(U)
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
すべて選択(A)
右から左に読む(R)
Unicode 制御文字の表示(S)
Unicode 制御文字の挿入(I) ▶
IME を開く(O)
再変換(B)

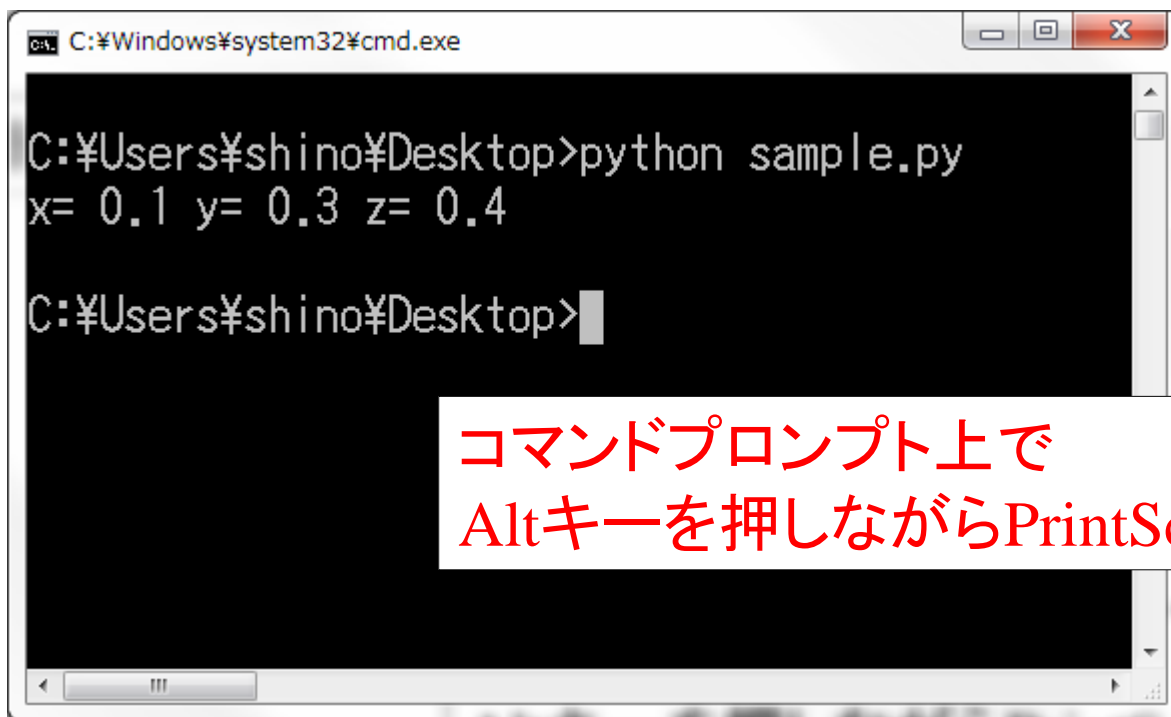
プログラムと実行結果をMS-Wordへの貼り付け方②

③ MS-Word上で右クリック
→「貼り付け」



プログラムと実行結果をMS-Word への貼り付け方③

実行結果

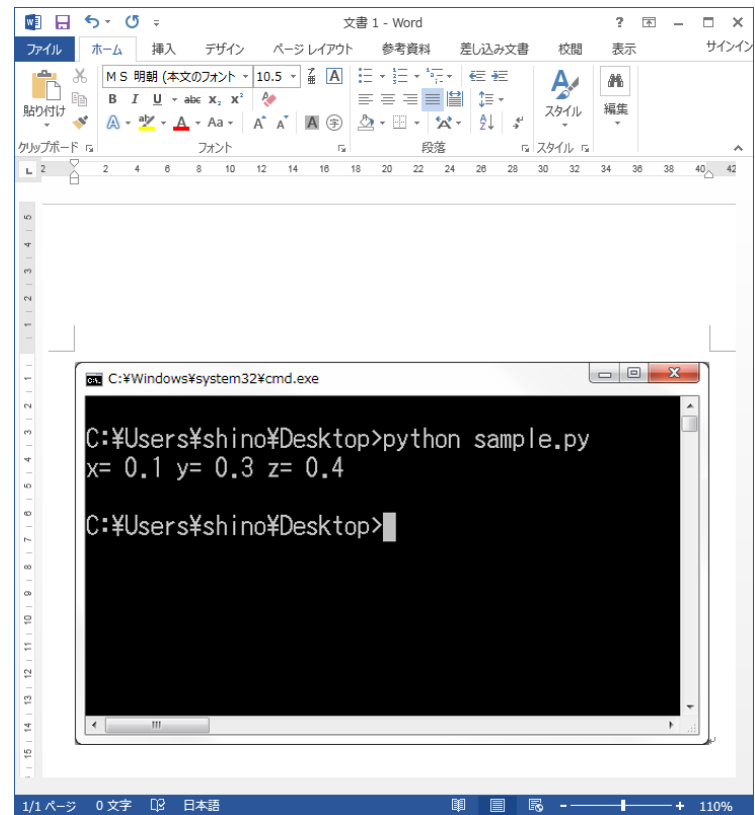
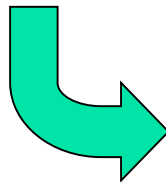
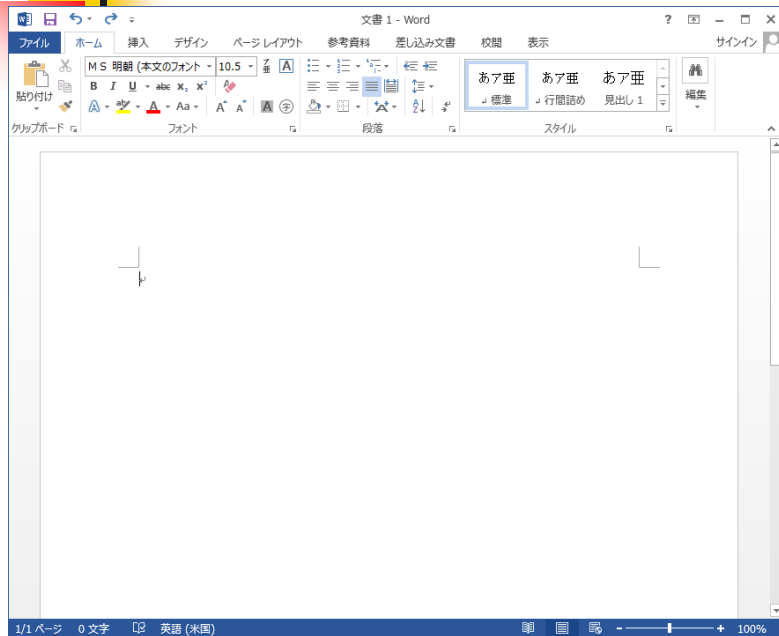


The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The command prompt displays the following text:

```
C:\Users\shino\Desktop>python sample.py  
x= 0.1 y= 0.3 z= 0.4  
  
C:\Users\shino\Desktop>
```

コマンドプロンプト上で
Altキーを押しながらPrintScrn

MS-Word上で右クリック
→「貼り付け」



コマンドプロンプトの画面が貼り付けられる