

# プログラミング言語

## 第3回 4月22日

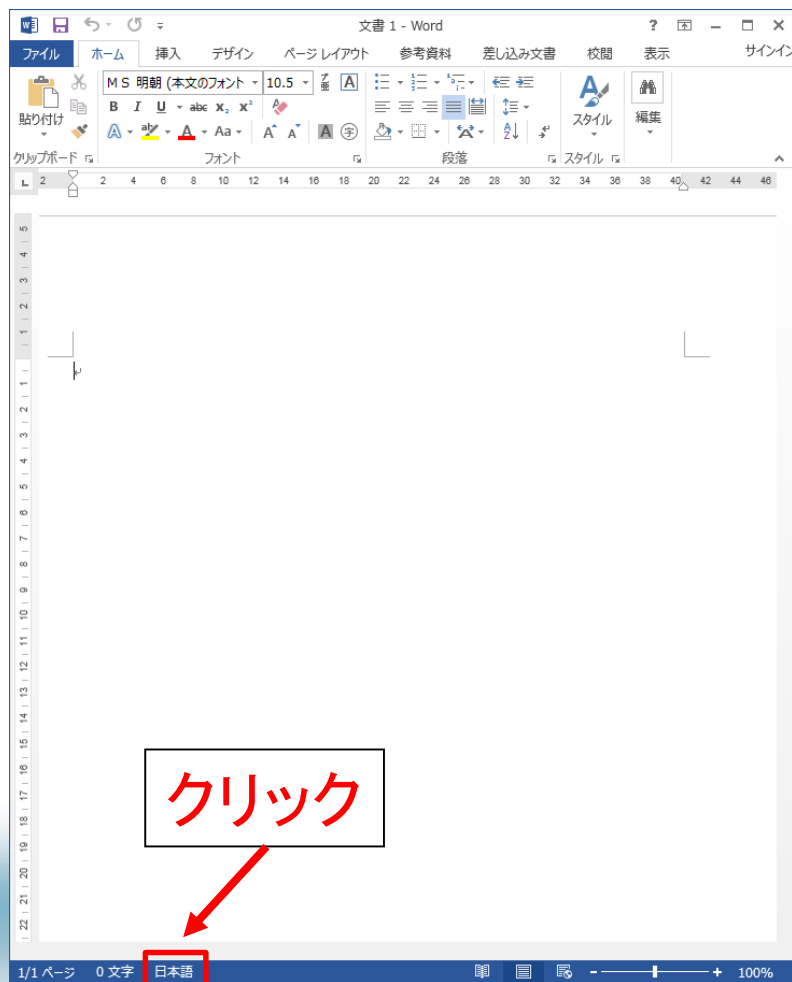
担当：篠沢 佳久  
栗原 聡

2019年度：春学期

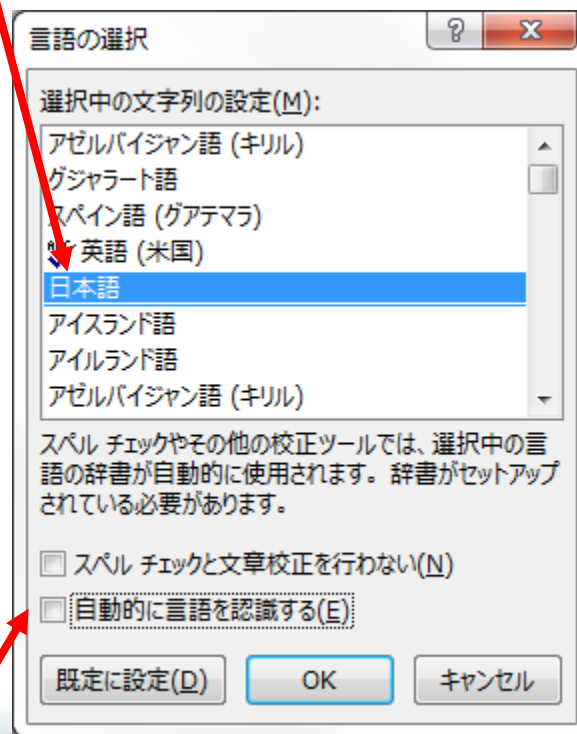
# 本日の内容

- ◆ 対話型シェルの使い方(復習)
- ◆ 変数
- ◆ 文と式
- ◆ 練習問題

# MS-Wordで日本語以外の文字が表示される場合



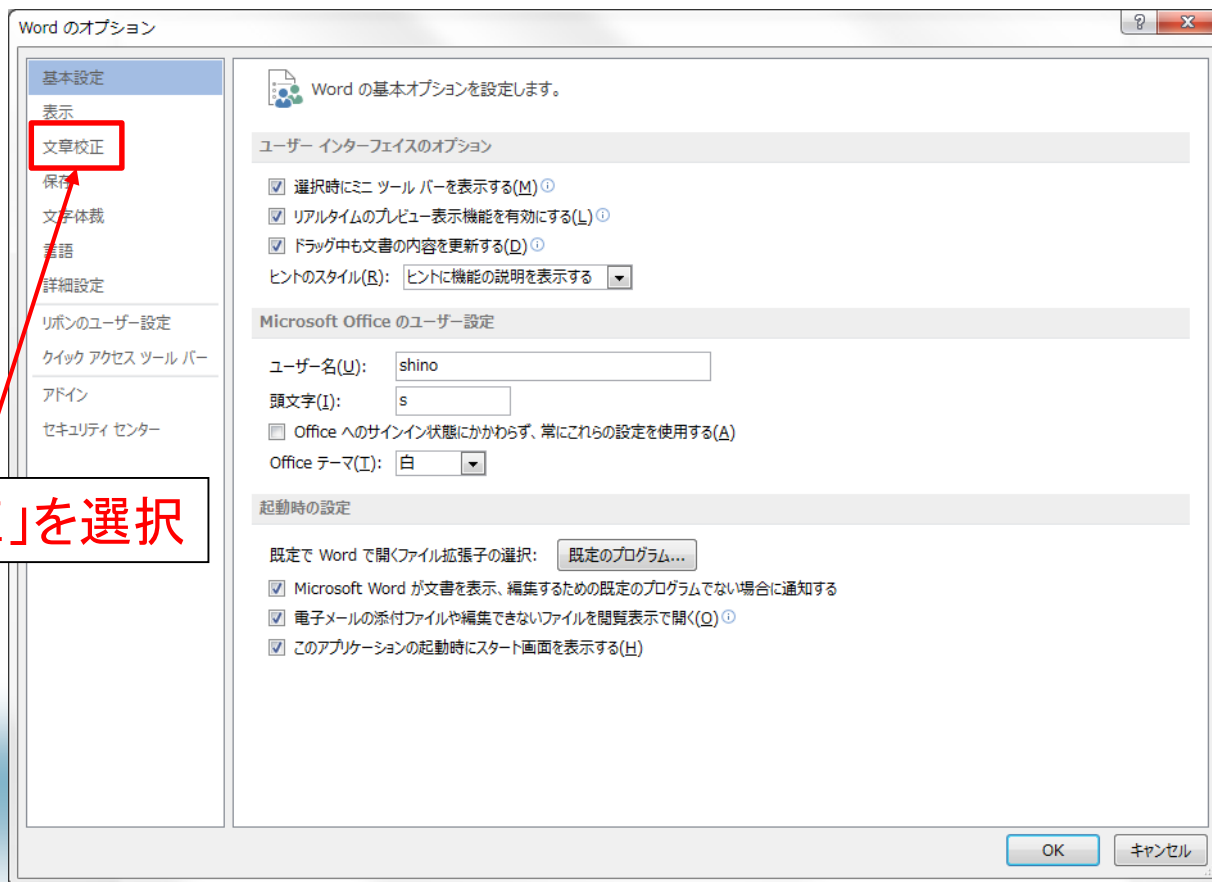
「日本語」になっているか



「自動的に言語を認識する」のチェックを外す

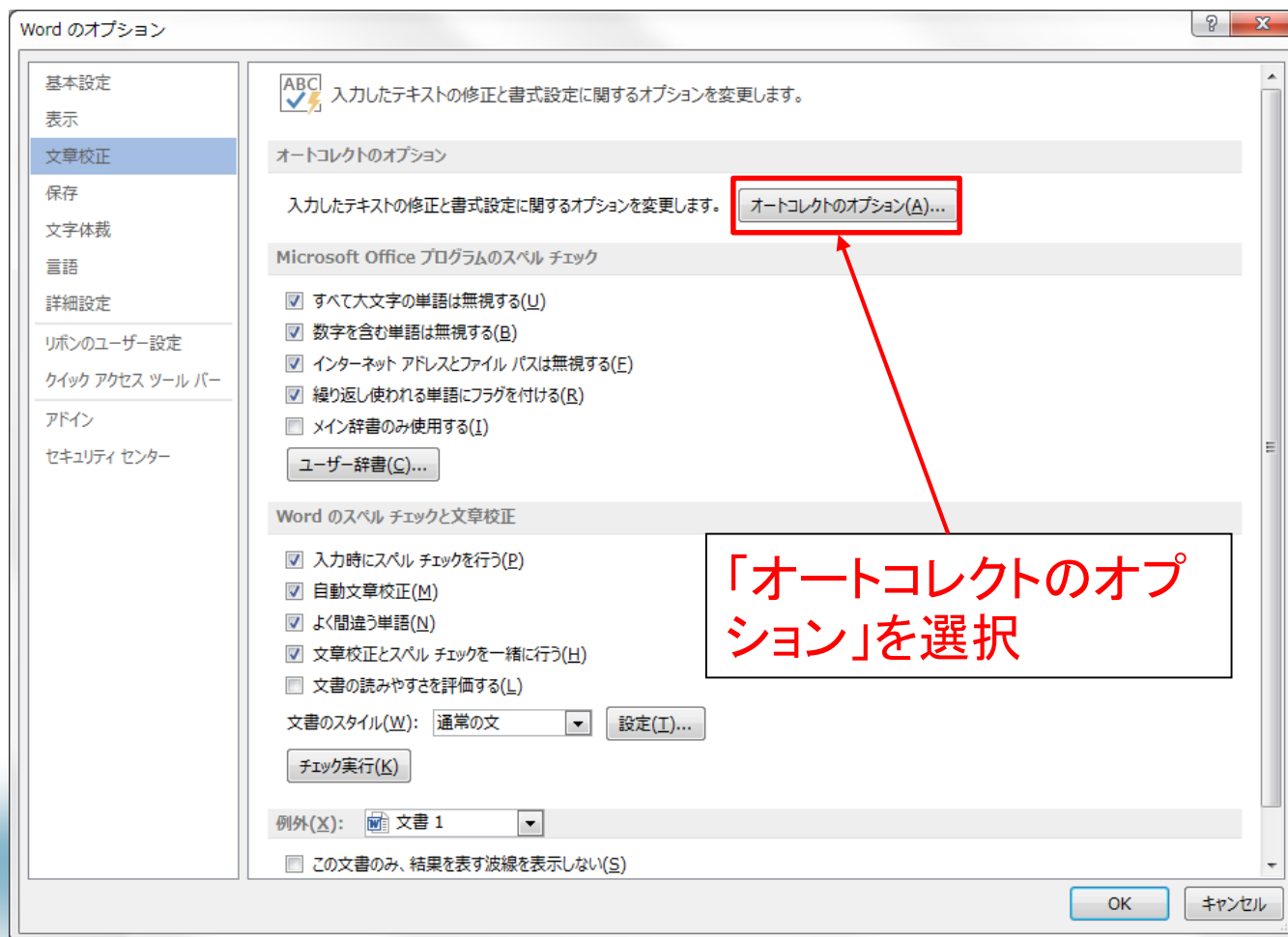
# MS-Wordでダブルクオートが自動変換される場合①

「ファイル」→「オプション」→「文章校正」



「文章校正」を選択

# MS-Wordでダブルクオートが自動変換される場合②



# MS-Wordでダブルクオートが自動変換される場合③

「入力オートフォーマット」を選択

オートコレクト

オートコレクト 数式オートコレクト **入力オートフォーマット** オートフォーマット 操作

入力中に自動で変更する項目

<input type="checkbox"/> 左右の区別がない引用符を、区別がある引用符に変更する	<input checked="" type="checkbox"/> 序数 (1st, 2nd, 3rd, ...) を上付き文字に変更する
<input type="checkbox"/> 分数 (1/2, 1/4, 3/4) を分数文字 (組み文字) に変更する	<input checked="" type="checkbox"/> ハイフンをダッシュに変更する
<input type="checkbox"/> '*'、'_' で囲んだ文字列を '太字'、'斜体' に書式設定する	<input checked="" type="checkbox"/> 長音とダッシュを正しく使い分ける
<input checked="" type="checkbox"/> インターネットとネットワークのアドレスをハイパーリンクに変更する	
<input type="checkbox"/> 行の始まりのスペースを字下げに変更する	

入力中に自動で書式設定する項目

<input checked="" type="checkbox"/> 箇条書き (行頭文字)	<input checked="" type="checkbox"/> 箇条書き (段落番号)
<input checked="" type="checkbox"/> 罫線	<input checked="" type="checkbox"/> 表
<input type="checkbox"/> 既定の見出しスタイル	<input type="checkbox"/> 日付スタイル
<input checked="" type="checkbox"/> 結語のスタイル	

入力中に自動で行う処理

<input checked="" type="checkbox"/> リストの始まりの書式を前のリストと同じにする
<input checked="" type="checkbox"/> Tab/Space/BackSpace キーでインデントとタブの設定を変更する
<input type="checkbox"/> 設定した書式を新規スタイルとして登録する
<input checked="" type="checkbox"/> かっこを正しく組み合わせる
<input type="checkbox"/> 日本語と英数字の間の不要なスペースを削除する
<input checked="" type="checkbox"/> '記' などに対応する '以上' を挿入する
<input checked="" type="checkbox"/> 頭語に対応する結語を挿入する

OK キャンセル

チェックを外す

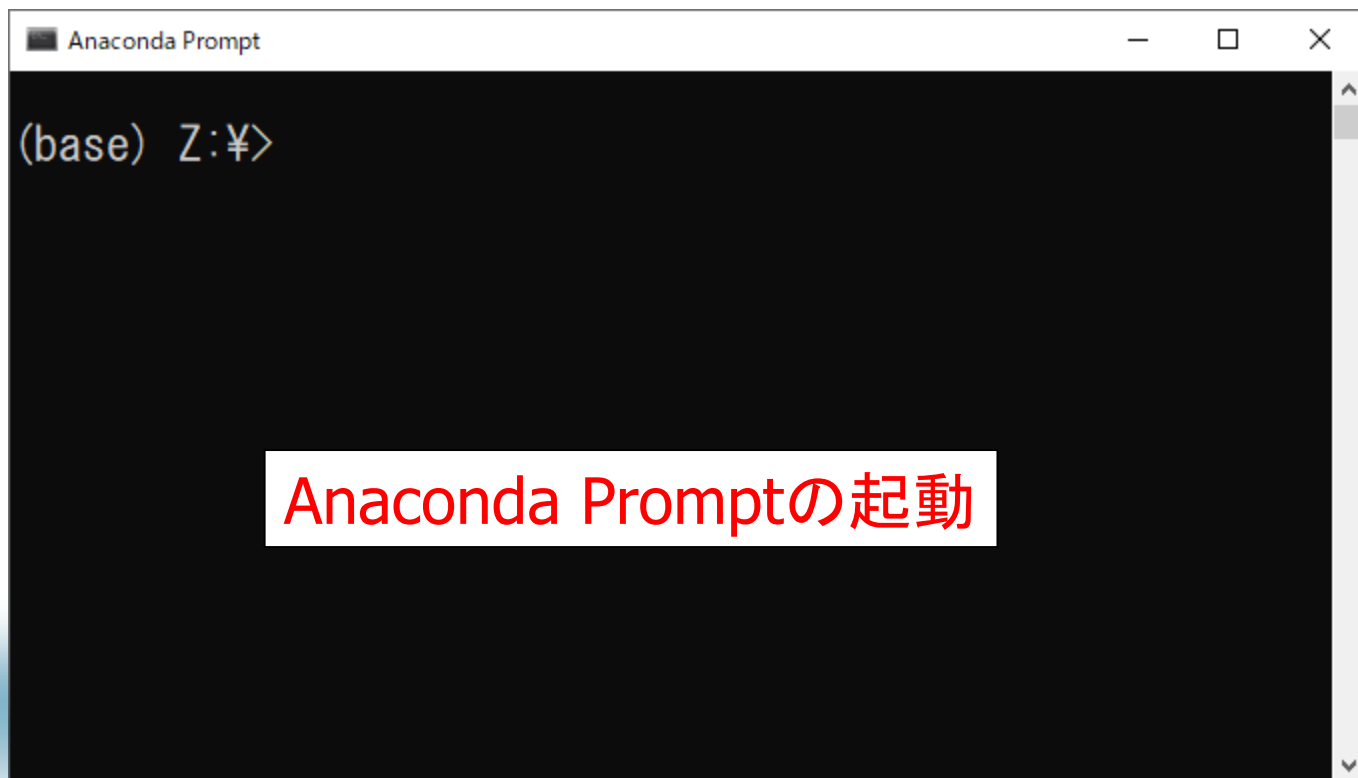
「OK」をクリック

# 対話型シェル(復習)

第二回講義資料より抜粋



# 対話型シェルの起動

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





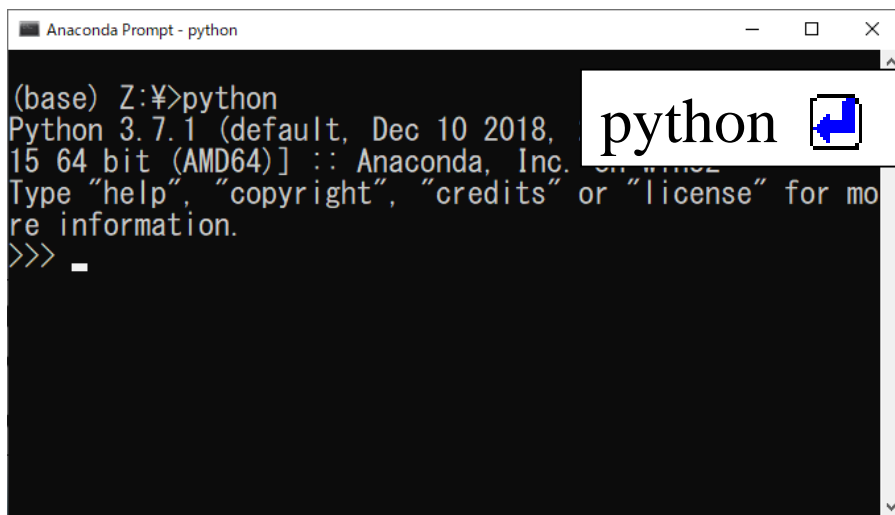
# コマンドプロンプトからの起動③

- ◆ コマンドプロンプト上で  
python   
と入力
- ◆ あとは interactive（会話的に）
  - 「コマンド」の入力
  - 実行結果の出力が無限に行われます
- ◆ 終了したいときには  
exit()   
と入力



Enterキー

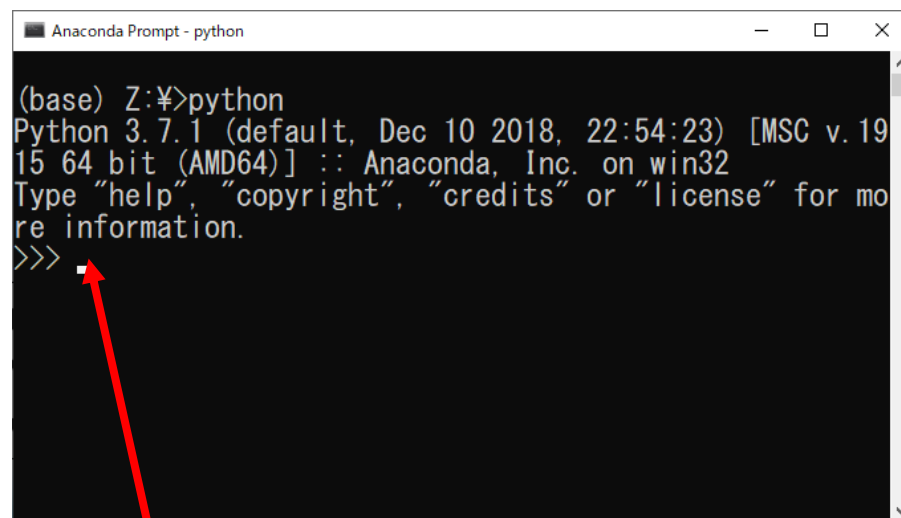
# 対話型シェルの起動②



```
Anaconda Prompt - python

(base) Z:\>python
Python 3.7.1 (default, Dec 10 2018, 15:15:15) [AMD64] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more
>>> _
```

python と入力



```
Anaconda Prompt - python

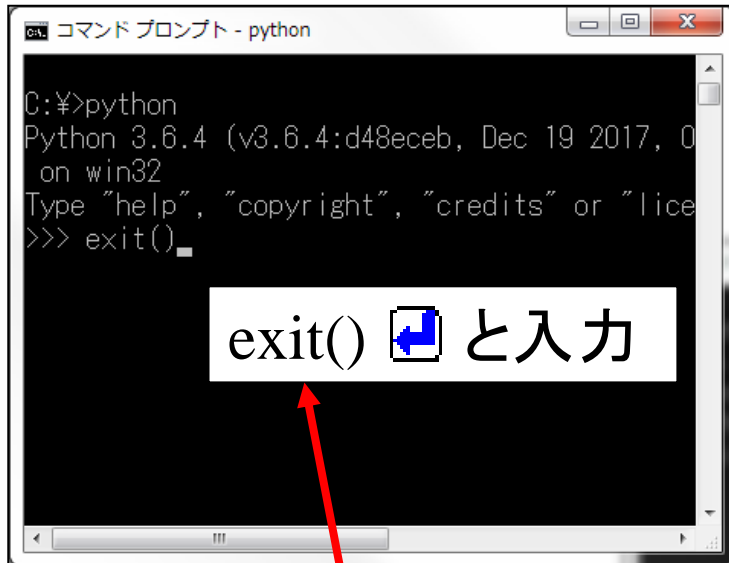
(base) Z:\>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
```

このプロンプトが表示されている時, Pythonのプログラムが実行可能

プロンプトが「>>>」と変わること  
に注目\*

\*講義資料の画面は日吉ITCのPC  
と異なる場合があります

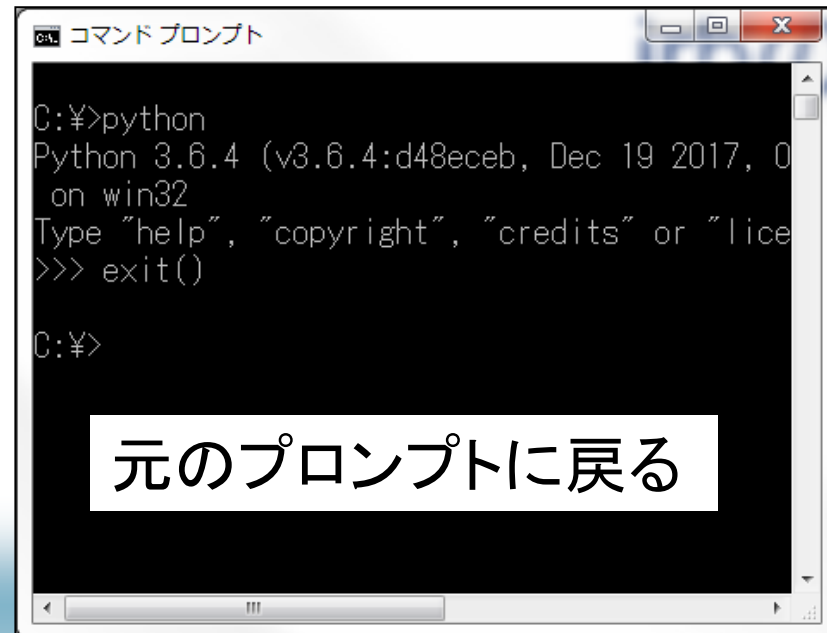
# 対話型シェルの終了



```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 0
on win32
Type "help", "copyright", "credits" or "lice
>>> exit()
```

exit() ⌨ と入力

もしくはCtrl+Z  
(Ctrlを押しながらZ)

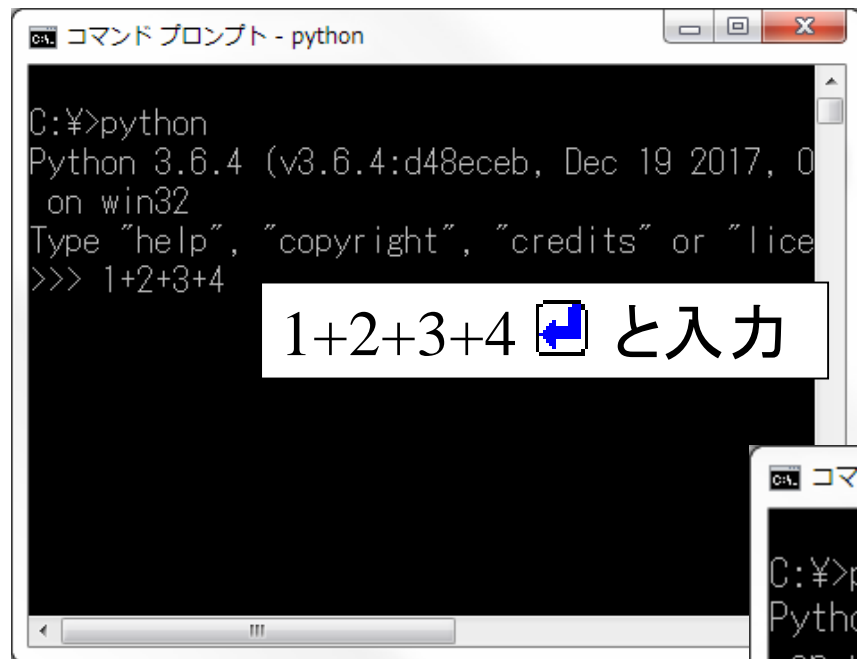


```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 0
on win32
Type "help", "copyright", "credits" or "lice
>>> exit()

C:\>
```

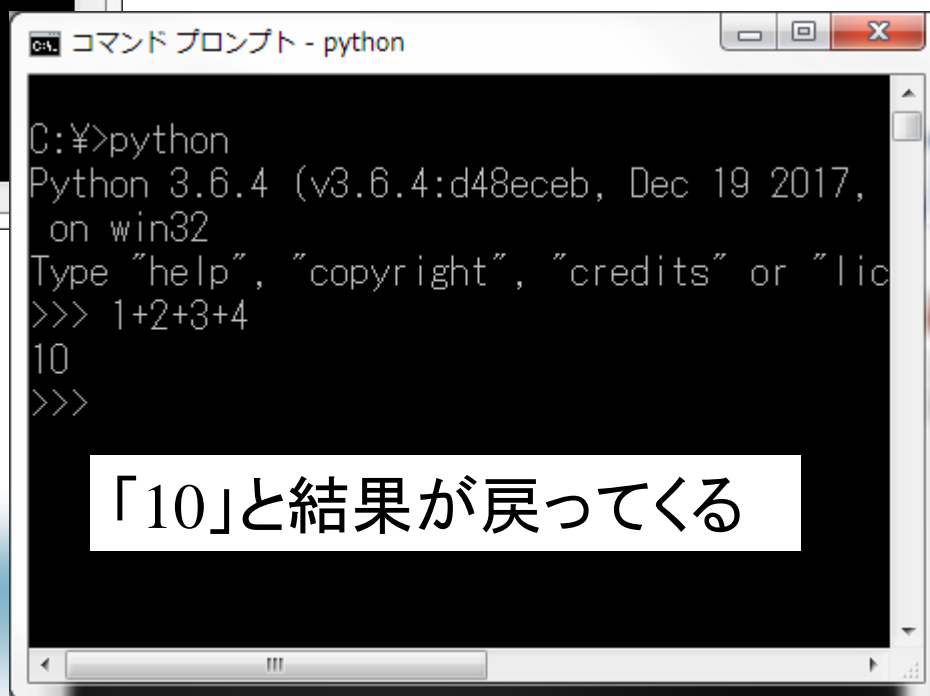
元のプロンプトに戻る

# 対話型シェル上での入力方法①



```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 0
on win32
Type "help", "copyright", "credits" or "lice
>>> 1+2+3+4
```

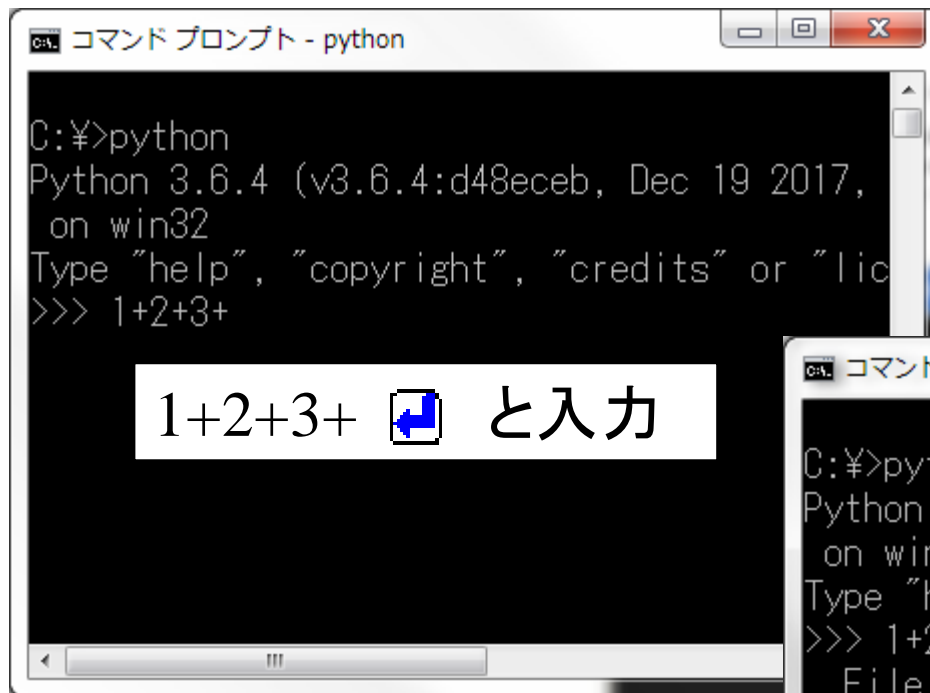
1+2+3+4 と入力



```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017,
on win32
Type "help", "copyright", "credits" or "lic
>>> 1+2+3+4
10
>>>
```

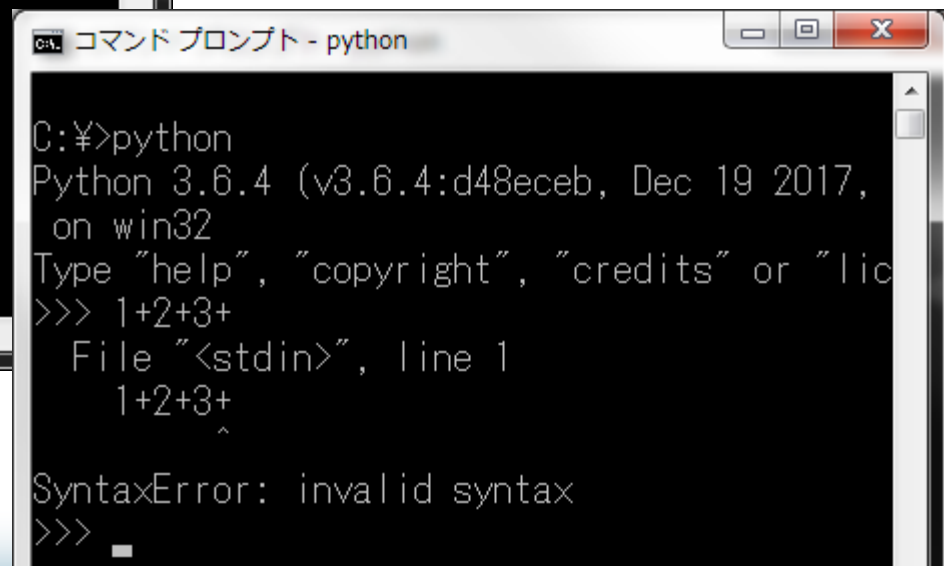
「10」と結果が戻ってくる

# 対話型シェル上での入力方法②



```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017,
on win32
Type "help", "copyright", "credits" or "lic
>>> 1+2+3+
```

1+2+3+ と入力



```
C:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017,
on win32
Type "help", "copyright", "credits" or "lic
>>> 1+2+3+
File "<stdin>", line 1
  1+2+3+
    ^
SyntaxError: invalid syntax
>>>
```

エラーメッセージ (SyntaxError) が表示される  
→ 式として誤りがあることを知らせてくれる

# 変数

変数と代入文

# 変数とは

- ◆ 今日覚える重要なことに「変数」があります
- ◆ 変数は、中学から代数で慣れ親しんだ変数とそっくりな概念です。そっくりですが、随分違いもあります。よく注意して下さい。
- ◆ 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(識別子)**をつけます。
- ◆ Pythonの**変数の型は記憶しているデータの型で決まります(重要！)**
  - Python の変数は、単に、場所の名前と思えばよい

# 変数①

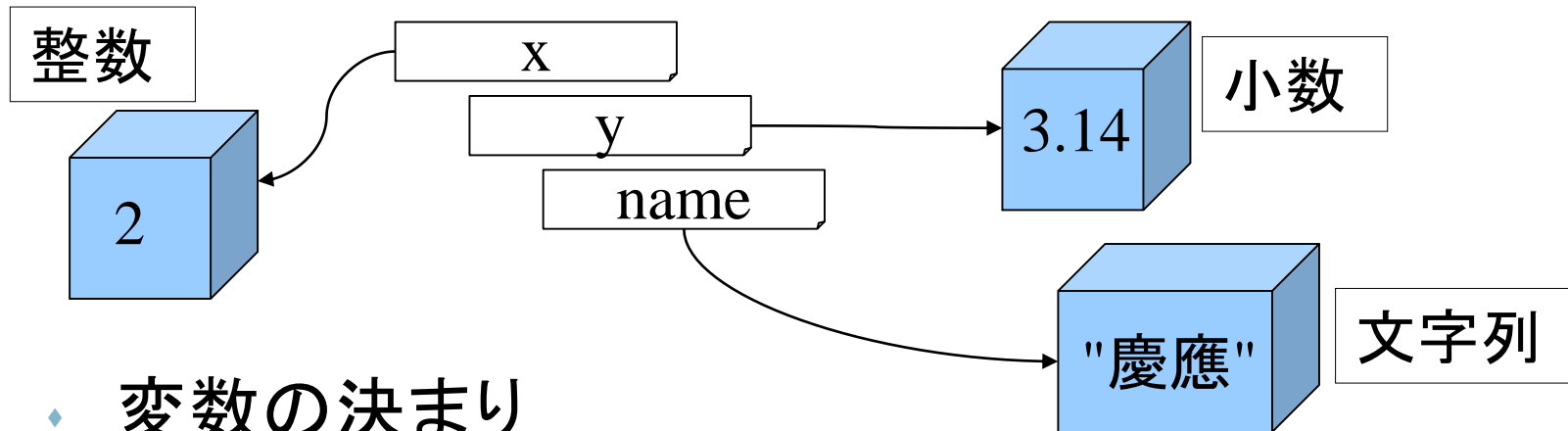
- ◆  $x = 2$ 
  - 変数 $x$  は整数型で値は2
- ◆  $y = 3.14$ 
  - 変数 $y$ は小数型で値は3.14
- ◆  $name = \text{"慶應"}$ 
  - 変数 $name$ は文字列型で値は"慶應"

変数の型は代入時に決まる



# 変数②

- ◆ コンピュータにデータを記憶させる機能のこと
  - 変数: 名前を作って, データが入った箱を区別するイメージ



- ◆ 変数の決まり
  - 変数に「**名前**」(識別子)をつける
  - 変数に値を代入する, とは, 割当てるもしくは割付けること

# 変数が使えるためには

- ◆ 変数を使うための準備
  - 変数に「名前」(識別子)をつける
    - 名前がないと, 何もできない
    - 名前は使えばよい
      - 「出生届け」はいらない
  - 変数を使えるようにする
    - 値を代入すればよい

# 識別子

- ◆ 変数につける名前のこと
  - 通常、英字・数字・アンダースコアを用いる  
例: num \_num
  - 数字で始めることはできない  
例: ~~1num~~
  - 大文字と小文字は区別される  
例: num と Num は区別される
  - Pythonで使う予約語(keyword)は使えない  
例: ~~class~~ ~~return~~ ~~break~~ ~~else~~
  - 日本語が使える(でも使わないで下さい)

# 予約語の一覧

and as assert break class continue def del  
elif else except finally for from global if  
import in is lambda nonlocal not or pass  
raise return try while with yield False  
None True

# 変数への値の代入

- ◆ 「識別子 = 値」で変数に値を代入
- ◆ 数学のイコールとは意味が違うことに注意

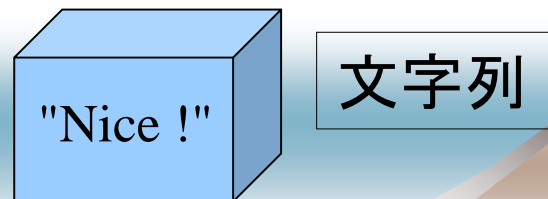
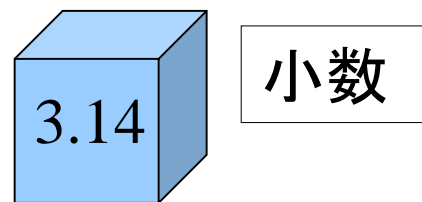
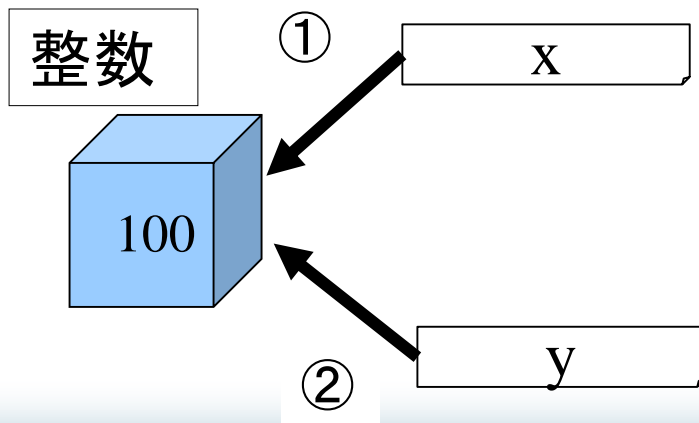
```
>>> x=100
>>> print(x)
100
>>> y=100
>>> print(y)
100
>>> x="Nice"
>>> print(x)
Nice
>>> x=3.14
>>> print(x)
3.14
```

変数の値を表示  
print(変数名)

```
>>> x=100
>>> print(x)
100
>>> y=100
>>> print(y)
100
>>> x="Nice"
>>> print(x)
Nice
>>> x=3.14
>>> print(x)
3.14
```

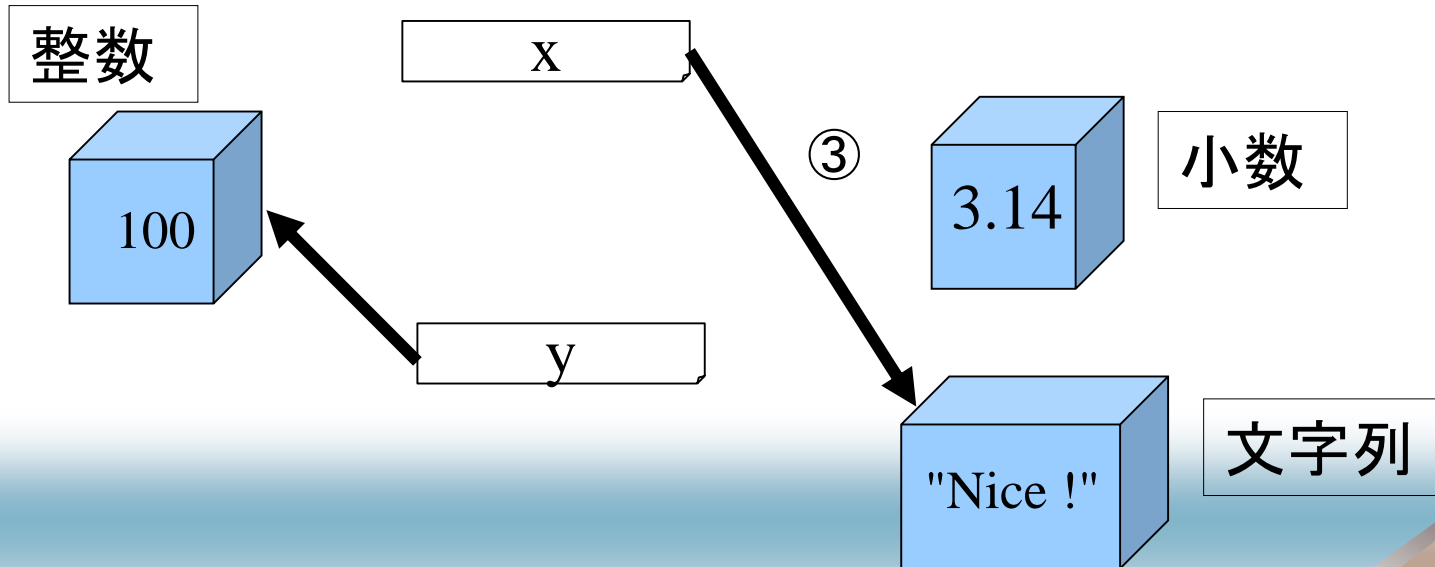
①

②



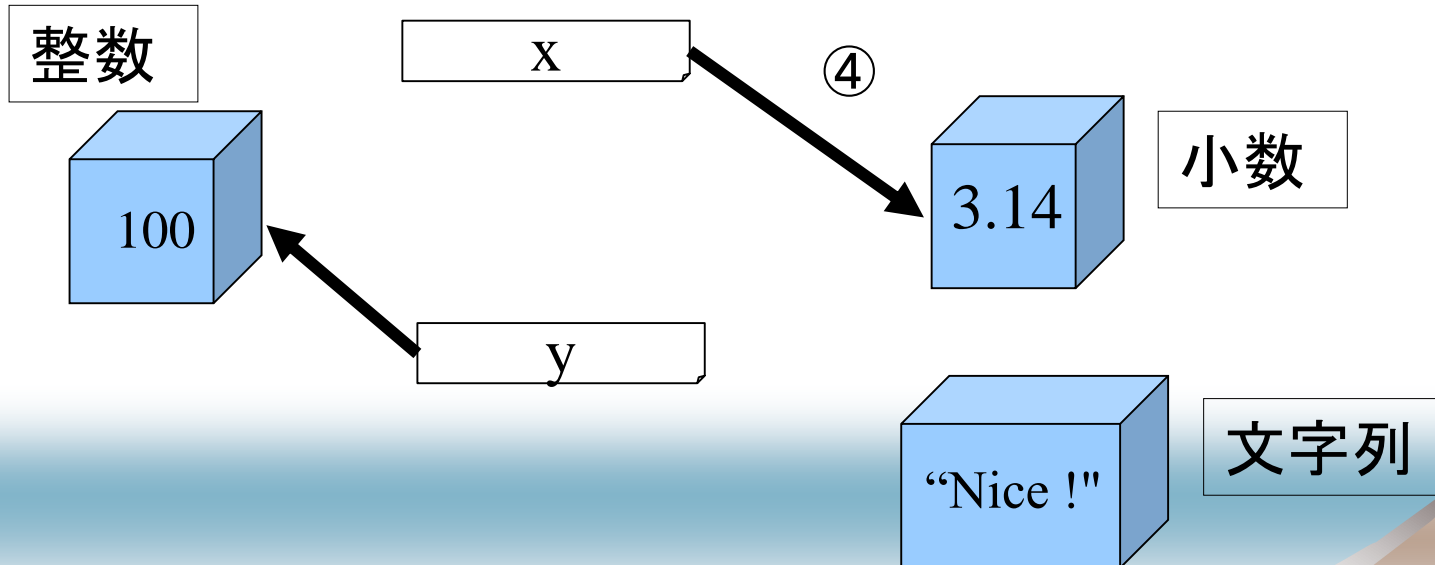
```
>>> x=100
>>> print(x)
100
>>> y=100
>>> print(y)
100
>>> x="Nice"
>>> print(x)
Nice
>>> x=3.14
>>> print(x)
3.14
```

③



```
>>> x=100
>>> print(x)
100
>>> y=100
>>> print(y)
100
>>> x="Nice"
>>> print(x)
Nice
>>> x=3.14
>>> print(x)
3.14
```

④





# 識別子の注意点①

```
>>> else = 5
SyntaxError: invalid syntax
>>> Else = 5
>>> print( Else )
5
>>> Num = 3
>>> num = 13
>>> print( Num , num , Num*num )
3 13 39
```

elseは予約  
語なので利  
用できない

Elseは予約  
語ではない  
ので利用可

Num

num

Num\*num

大文字, 小文字は区別  
される

## 識別子の注意点②

変数xは未定義

```
>>> x
Traceback (most recent call last):
  File "<pyshe11#6>", line 1, in <module>
    x
NameError: name 'x' is not defined
>>> x = 3
>>> print( x )
3
```

始めて用いる変数は代入式で値を代入することによって使用できるようになる

## 識別子の注意点③

```
>>> りんご=3  
>>> みかん=5  
>>> 値段=りんご*150+みかん*60  
>>> print(値段)  
750
```

変数名に日本語も使えますが、この授業では使わないで下さい

# 変数の表示①

- ◆ 変数に入っている値をディスプレイに表示（印字，出力とも言います）させたい場合
- ◆ `print( 変数名 )`

```
>>> x = 3
```

xに3を代入

```
>>> print(x)
```

xの値を表示

```
3
```

```
>>> x = 3.1415
```

```
>>> print(x)
```

```
3.1415
```

```
>>> x = "abcd"
```

```
>>> print(x)
```

```
abcd
```

スペースが入る

## 変数の例①

```
>>> x=25
>>> print( "xの値は" , x , "です" )
xの値は 25 です
>>> x=30
>>> print( "xの値は" , x , "に変わりました" )
xの値は 30 に変わりました
```

変数の中身は上書きされる

```
print( "コメント" )

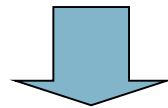
print( "コメント" , 変数名 )

print( 変数名1 , 変数名2 )

print( "コメント1" , 変数名 , "コメント2" )
```

## 変数の例②

```
>>> x=25  
>>> print( "xの値は" , x , "です" )  
xの値は 25 です
```



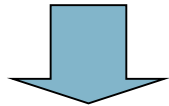
一行で書く場合

```
x=25; print( "xの値は" , x , "です" )  
xの値は 25 です
```

二つの文を一行で書きたい場合は、「;」(セミ  
コロン)で区切る

## 変数の例③

```
>>> x=30  
>>> print( "xの値は" , x , "に変わりました" )  
xの値は 30 に変わりました
```



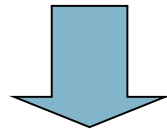
一行で書く場合

```
>>> x=30; print( "xの値は" , x , "に変わりました" )  
xの値は 30 に変わりました
```

セミコロン

## 変数の例④

```
>>> x=25
>>> print( "xの値は" , x , "です" )
xの値は 25 です
>>> x=30
>>> print( "xの値は" , x , "に変わりました" )
xの値は 30 に変わりました
```



一行で書く場合

```
>>> x=25; print( "xの値は" , x , "です" ); x=30;
print( "xの値は" , x , "に変わりました" )
xの値は 25 です
xの値は 30 に変わりました
```



# 変数の実例①

## ◆ 変数があると便利

```
# sample31.py
print( "僕の名前は、寿限無寿限無五劫の擦り切れ海砂利水魚の水行末雲来末風来末
食う寝る処に住む処やぶら小路の藪柑子パイポパイポパイポのシューリンガンシュー
リンガンのゲーリンダイゲーリンダイのポンポコピーのポンポコナーの長久命の長助
だ〜い" )
print( "え、君の名前は、寿限無寿限無五劫の擦り切れ海砂利水魚の水行末雲来末風
来末食う寝る処に住む処やぶら小路の藪柑子パイポパイポパイポのシューリンガン
シューリンガンのゲーリンダイゲーリンダイのポンポコピーのポンポコナーの長久命
の長助かい？ 長いなあ" )
```



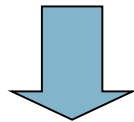
変数 **name** に値を代入しておく

```
# sample32.py
name = "寿限無寿限無五劫の擦り切れ海砂利水魚の水行末雲来末風来末食う寝る処に
住む処やぶら小路の藪柑子パイポパイポパイポのシューリンガンシューリンガンの
ゲーリンダイゲーリンダイのポンポコピーのポンポコナーの長久命の長助"
print( "僕の名前は、", name, "だ〜い" )
print( "え、君の名前は、", name, "かい？ 長いなあ" )
```

# 変数の実例②

## ◆ 変数があると便利

```
# sample33.py
print( "僕の名前は, パブロ・ディエゴ・ホセ・フランチスコ・ド・ポール・ジャン・ネボムチェーノ・クリスバン・クリスピアノ・ド・ラ・ンチシュ・トリニダット・ルイス・イ・ピカソだ〜い" )
print( "え, 君の名前は, パブロ・ディエゴ・ホセ・フランチスコ・ド・ポール・ジャン・ネボムチェーノ・クリスバン・クリスピアノ・ド・ラ・ンチシュ・トリニダット・ルイス・イ・ピカソかい? 長いなあ" )
```



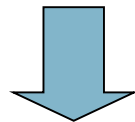
変数 **name** に値を代入しておく

```
# sample34.py
name = "パブロ・ディエゴ・ホセ・フランチスコ・ド・ポール・ジャン・ネボム  
チェーノ・クリスバン・クリスピアノ・ド・ラ・ンチシュ・トリニダット・ルイス・  
イ・ピカソ"  
print( "僕の名前は, ", name, "だ〜い" )  
print( "え, 君の名前は, ", name, "かい? 長いなあ" )
```

# 変数の実例③

- ◆ 変数があると便利

```
# sample35.py
print( "My name is Pablo Diego José Santiago Francisco de Paula Juan
Nepomuceno Crispín Crispiniano de los Remedios Cipriano de la Santísima
Trinidad Ruiz Blasco y Picasso" )
print( "What? Is your name Pablo Diego José Santiago Francisco de Paula Juan
Nepomuceno Crispín Crispiniano de los Remedios Cipriano de la Santísima
Trinidad Ruiz Blasco y Picasso ? It's too long." )
```



変数 `name` に値を代入しておく

```
# sample36.py
name = "Pablo Diego José Santiago Francisco de Paula Juan Nepomuceno Crispín
Crispiniano de los Remedios Cipriano de la Santísima Trinidad Ruiz Blasco y
Picasso"
print( "My name is " , name , "." )
print( "What? Is your name " , name , "? It's too long." )
```

# いろいろなデータを一つの変数に

一行で書く

セミコロンでつなげる

```
>>> x=20; print( "x is now" , x );  
x = 3.14; print( "but x is now" , x );  
x = "Hi"; print( "but x is now" , x )
```

x is now 20

but x is now 3.14

but x is now Hi

小数

整数

文字列

# 再び: データ型

- ◆ コンピュータ内部ではどう表現しているのだろうか？ どう演算しているのだろうか？
- ◆ 整数型の場合
  - 整数表現. 2進数. 8/16/32/64ビットを一まとめにして記憶している

# 注：有限性

- ◆ 我々は有限なものしか認識できない(かどうかは、哲学に任せよう)
- ◆ コンピュータで取り扱えるものは、有限のみ
- ◆ 循環小数はどう表現しよう？
  - 分数で表せば有限なんだけどね。それは別の話。
- ◆ 無理数はどう表現しよう？
  - 有限桁数の近似表現で我慢しよう
- ◆ 非常に大きな整数はどう表現しよう
  - できるだけ表現しよう(Pythonの場合)

# 整数型の可能性

- ♦ Python の整数型は、非常に大きな数を表現できます
- ♦ まあ、皆さんの常識からすれば、これは当たり前なのですが、コンピュータ言語の世界では、常識ではありません
- ♦ まずは、試してみましょう

```
>>> 2**1000
10715086071862673209484250490600018105614048117055336074
43750388370351051124936122493198378815695858127594672917
55314682518714528569231404359845775746985748039345677748
24230985421074605062371141877954182153046474983581941267
39876755916554394607706291457119647768654216766042983165
2624386837205668069376
```

2 \*\* 10000 も試してみましょう

# 整数型(2進数)①

- ◆ 2進数の表し方
  - 先頭に0bをつける
  - (例) 0b101, 0b11111
- ◆ 10進数→2進数変換
  - bin(10進整数)
  - (例) bin(1000)
  - 2進数(文字列型)となる
- ◆ 2進数→10進数変換
  - int(2進数(文字列型), 2)
  - (例) int("0b1011", 2)
  - 10進数(整数型)となる



## 整数型(2進数)②

```
>>> 0b10111
```

2進数  $(10111)_2$

```
23
```

```
>>> bin(1000)
```

$(1000)_{10}$  を2進数に変換  
→結果は2進数(文字列型)

```
'0b1111101000'
```

```
>>> int("0b1111",2)
```

$(1111)_2$  を10進数に変換  
→結果は10進数(整数型)

```
15
```

文字列で指定する

# 整数型(16進数)①

- ◆ 16進数の表し方
  - 先頭に0xをつける
  - (例) 0xc42, 0xffffe
- ◆ 10進数→16進数変換
  - hex(10進整数)
  - (例) hex(1000)
  - 16進数(文字列型)となる
- ◆ 16進数→10進数変換
  - int(16進数(文字列型), 16)
  - (例) int("0x1234", 16)
  - 10進数(整数型)となる

## 整数型(16進数)②

```
>>> 0x1f
```

16進数  $(1f)_{16}$

```
31
```

```
>>> hex(125)
```

$(125)_{10}$  を16進数に変換  
→結果は16進数(文字列型)

```
'0x7d'
```

```
>>> int("0x123",16)
```

$(123)_{16}$  を10進数に変換  
→結果は10進数(整数型)

```
291
```

文字列で指定する

# 浮動小数点数型

- ◆ 浮動小数点数 floating point number
- ◆ コンピュータのなかで、整数以外の有限桁の数を表現する工夫です  
物理や化学で絶対値の大きな数や小さな数を表す工夫と同じです
  - $6.0221367 \times 10^{23}$  って何ですか？
- ◆ 整数  $\times 2^{\text{整数}}$  で表現します  

仮数部

指数部
- ◆ 仮数部は有限桁です。最近では、53ビットが多い
- ◆ 指数部も有限桁です

# 浮動小数点数型 Python の場合

- ◆ 10進数で約15桁の精度があります

小数点以下30桁まで表示

```
>>> print( "{0:.30}".format(1/3) )  
0.33333333333333333333333314829616256247  
>>> print( "{0:.30}".format(1/10) )  
0.1000000000000000000000005551115123126
```

# 浮動小数点数の指数の制限①

- ◆ 指数だって、有限の長さしかない

```
>>> 10.0**308
```

```
1e+308
```

1.0e+308

$1.0 \times 10^{308}$

```
>>> 10.0**309
```

```
Traceback (most recent call last):
```

```
  File "<pyshe11#45>", line 1, in <module>
```

```
    10.0**309
```

```
OverflowError: (34, 'Result too large')
```

$10^{309}$ は大きすぎて扱えません

# 浮動小数点数の指数の制限②

- ◆ 指数だって、有限の長さしかない

```
>>> 10.0**-323
1e-323
>>> 10.0**-324
0.0
```

$e^{-323}$

$10^{-323}$

$10^{-324}$ は0として扱われます

# 変数の型変換①

- 変数においても型変換が可能

x=3

整数型

float( x )

小数へ変換

str( x )

文字列へ変換

x=3.1415

小数型

int( x )

整数へ変換

str( x )

文字列へ変換



```
>>> x=3
>>> float(x)
3.0
>>> print(x)
3
>>> str(x)
'3'
>>> print(x)
3
```

型変換しても、変数の中身  
を変えているわけではない  
xは整数3のまま

```
>>> x=3.141
>>> int( x )
3
>>> print( x )
3.141
>>> str( x )
'3.141'
>>> print( x )
3.141
```

型変換しても、変数の中身  
を変えているわけではない  
xは小数3.141のまま

# 変数の型変換①

- 文字列においても型変換が可能

x="3"

文字列型

int( x )

整数へ変換

float( x )

小数へ変換

x="3.1415"

文字列型

float( x )

小数へ変換

int( float( x ) )

小数→整数へ変換

( × ) int( x )

(注意)一回で整数に型変換はできません

```
>>> x="3"
>>> int(x)
3
>>> float( x )
3.0
>>> x
'3'
```

型変換しても、変数の中身  
を変えているわけではない  
xは文字列3のまま

```
>>> x="3.141"
>>> float(x)
3.141
>>> int( x )
Traceback (most recent call last):
  File "<pyshe11#80>", line 1, in <module>
    int(x)
ValueError: invalid literal for int() with base 10:
'3.141'
>>> int( float( x ) )
3
>>> x
'3.141'
```

整数に変換できない

整数に変換するため  
には、小数→整数

型変換しても、変数の中身  
を変えているわけではない  
xは文字列3.141のまま

# 文と式

数式, 文字列式, 論理式  
代入文, 条件文

# 式

- ◆ 式は**演算子**（オペレータ. 演算内容を表すもの）と**オペランド**（演算の対象）の組み合わせ
- ◆ 例：  $3+4$ 
  - 「3」と「4」がオペランド, 「+」が演算子

# 文と式①

- ◆ 今まで曖昧にしてきたが、重要なこと
  - ◆ プログラムは文と式から構成されています
  - ◆ 文とは、代入文, print文, 条件文, for文(後述)など実際に命令を記載する単位のことです
  - ◆ 一方で, 式は(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

## 文と式②

```
>>> 3
3
>>> 5
5
>>> a=3
>>> b=3
>>> a*b
9
>>> c=a+b
>>> c
6
>>> print(c)
6
```

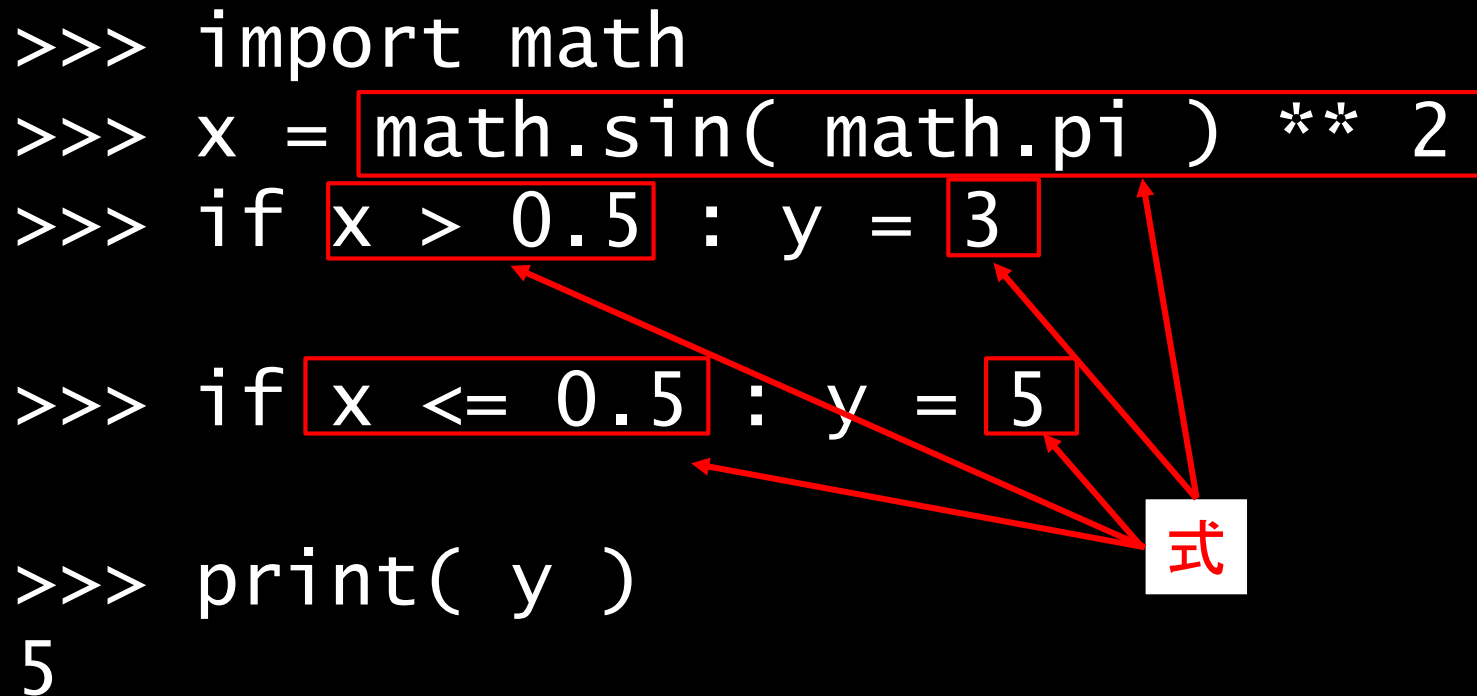
式

文: 代入文  
→ 値を持たない

文: print文  
→ 命令で値を表示していますが  
、値は持たない

## 文と式③

```
>>> import math
>>> x = math.sin( math.pi ) ** 2
>>> if x > 0.5 : y = 3
>>> if x <= 0.5 : y = 5
>>> print( y )
5
```





## 文と式④

```
>>> import math
>>> x = math.sin( math.pi ) ** 2
>>> if x > 0.5 : y = 3

>>> if x <= 0.5 : y = 5

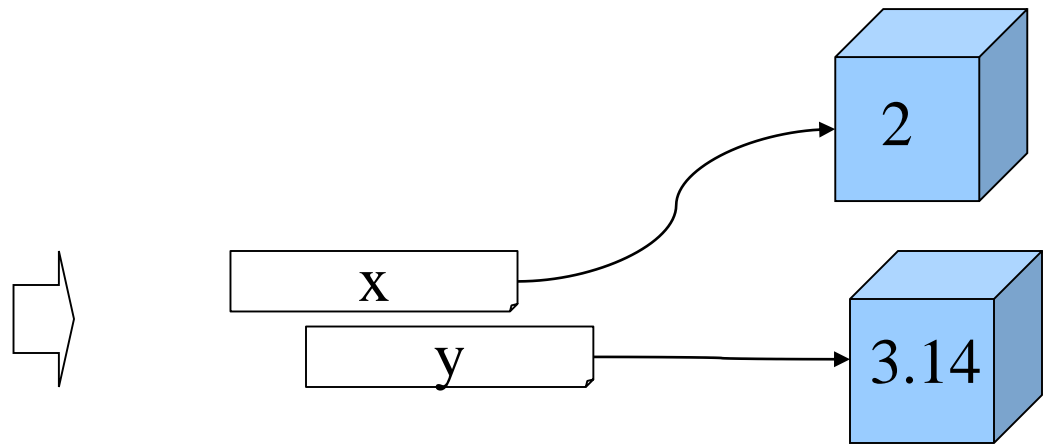
>>> print( y )
5
```

これらは全て文

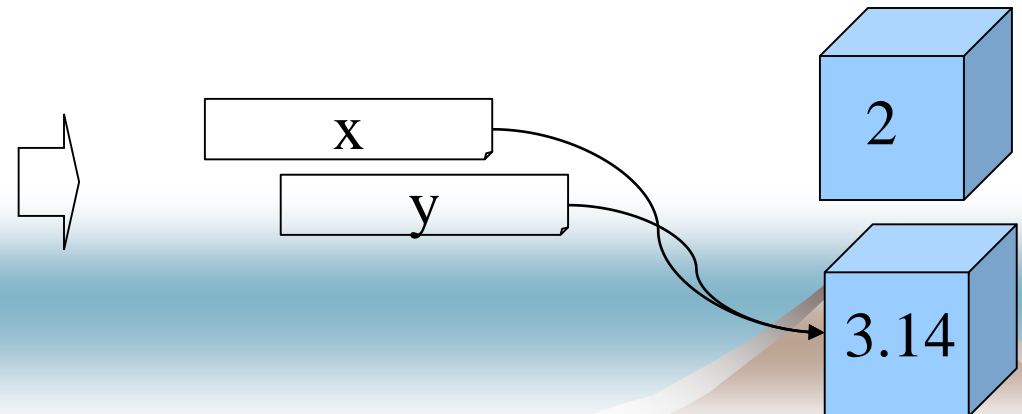
# 代入文

- 変数に他の式, 変数の値を代入する
  - 等式ではない!
- イメージでは,

```
>>> x=2  
>>> print(x)  
2  
>>> y=3.14  
>>> print(y)  
3.14
```



```
>>> x=y  
>>> print(x)  
3.14
```



# 式の種類

- ◆ 式は基本的に代入文の右辺に表れます
  - 後で学ぶ条件文, for文などではそうではありません
- ◆ これまでのように, 数値の計算を行わせる式を数式と呼ぶことにします
- ◆ 文字列の計算を行わせる式を文字列式と呼ぶことにします
- ◆ 論理値(真偽値)を計算する式を論理式と呼ぶことにします
- ◆ 論理式の基本は, 数式または文字列式(の値)と数式または文字列式(の値)とを比較演算子を用いて比較する式です. これを論理演算子(and/or/not)で組み合わせます
- ◆ 他に三項演算子(後述)を用いた条件式もあります

# 複雑な式

- ◆ 四則演算と剰余演算と括弧とを自由に(勿論, 正しく)組み合わせたものは正しい式になる
  - 但し, 型の混合があると, ちょっと, 面倒
- ◆ そのほか,
  - 三角関数 `math.pi`, `math.sin`, `math.cos`, `math.tan`
  - 対数・指数関数 `math.e`, `math.log`, `math.exp`

```
>>> import math
>>> x=math.sin(math.pi*0.1) +
math.exp(math.log(math.pi+0.1))
>>> print(x)
3.5506096479647407
```

import mathを忘れずに

# 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

# 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

- 実は、代入記号(「=」など)も演算子なのです。そして、種類がたくさんあります

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

# 代入演算子の例

```
>>> x=10
>>> x+=5 ← x=x+5
>>> print(x)
15
>>> x=10
>>> x*=5 ← x=x*5
>>> print(x)
50
>>> x=10
>>> x+=x ← x+=x にてxは20
>>> print(x)
20
>>> x*=x ← x*=x にてxは400
>>> print(x)
400
```



# 論理式

## 比較演算子

◆  $3 == 3$

- 3と3は等しい

◆  $3 > 2$  **and**  $4 > 3$

- $3 > 2$  かつ  $4 > 3$

◆  $"abc" == "abc"$  **or**  $"xyz" == "XYZ"$

- $"abc" == "abc"$  または  $"xyz" == "XYZ"$

## 論理演算子

# 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

# 論理式の例①

```
>>> 3 > 2
True
>>> 3 != 3
False
>>> 3 > 2 and 3 == 3
True
>>> 3 > 2 and 3 != 2
True
>>> 3 > 2 and 3 != 3
False
>>> 3 > 2 or 2 != 2
True
>>> 3 > 2 and 3 == 2
False
>>> 3 > 2 or 3 == 2
True
```

演算式が正しい場合  
→「True」を返す

演算式が正しい場合  
→「False」を返す

## 論理式の例②

```
>>> True == True
```

```
True
```

```
>>> True == False
```

```
False
```

```
>>> True and True
```

```
True
```

```
>>> True and False
```

```
False
```

```
>>> True or False
```

```
True
```

```
>>> True and False or True
```

```
True
```

演算式が正しい場合  
→「True」を返す

演算式が正しい場合  
→「False」を返す

# 複雑な論理式

- 比較演算子を用いて、論理式を作り、さらに、論理演算子と組み合わせて、複雑な論理式を書くことができる
- 括弧を用いて優先順位も決められる

```
import math
>>> math.pi > 3.14 and ( math.e > 2.7 or 3 != 2
or math.sin(3.4) > 0.0 ) and "pro" in
( "python"+" programmig" )
True
```



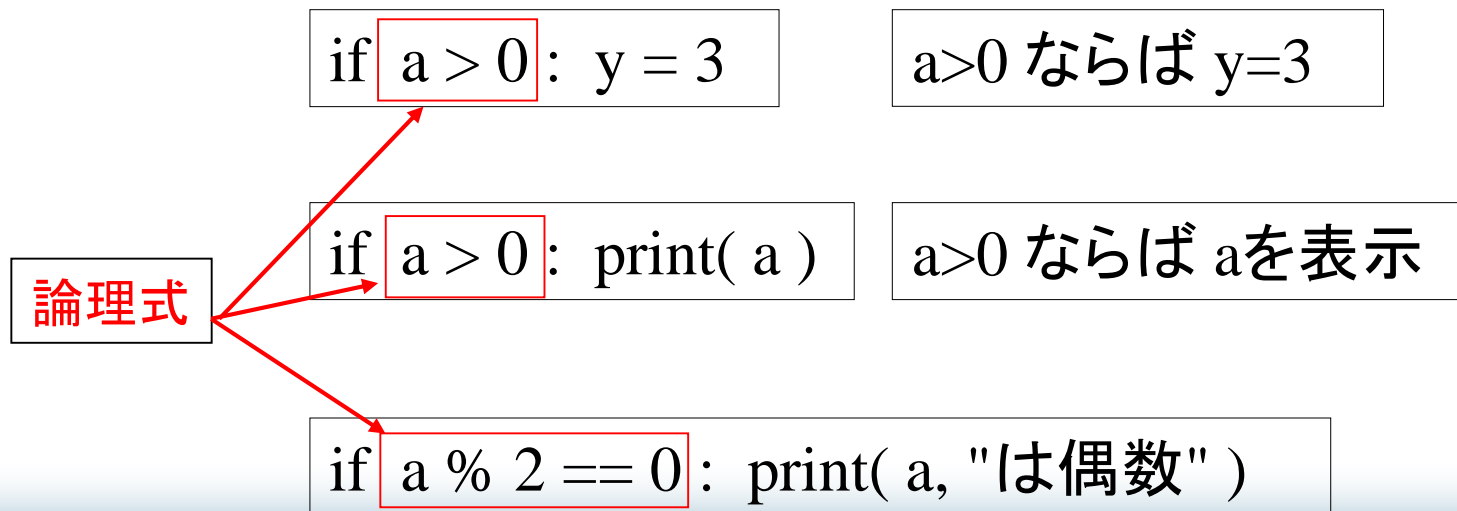
比較演算子



論理演算子

# 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

## 条件文②

```
>>> x = 5
>>> a = 10
>>> if a > 0: x = 3
>>> print(x)
3
```

$a > 0$ はTrue→ $x=3$ を実行

Enterをもう一回入力

```
>>> x = 5
>>> a = -10
>>> if a > 0: x = 3
>>> print( x )
5
```

$a > 0$ はFalse→ $x=3$ は実行  
されない



## 条件文③

```
x = -10
```

```
if x > 0: y = x
```

```
if x <= 0: y = -x
```

```
print( y )
```

x>0 ならば y=x

x<=0 ならば y=-x

xは-10なのでこちらの式が実行される

```
>>> x=-10
>>> if x > 0: y = x
>>> if x <=0: y = -x
>>> print(y)
10
```

## 条件文④

```
x = 5
```

```
if x % 2 == 0: print( x , "は偶数" )
```

```
if x % 2 != 0: print( x , "は奇数" )
```

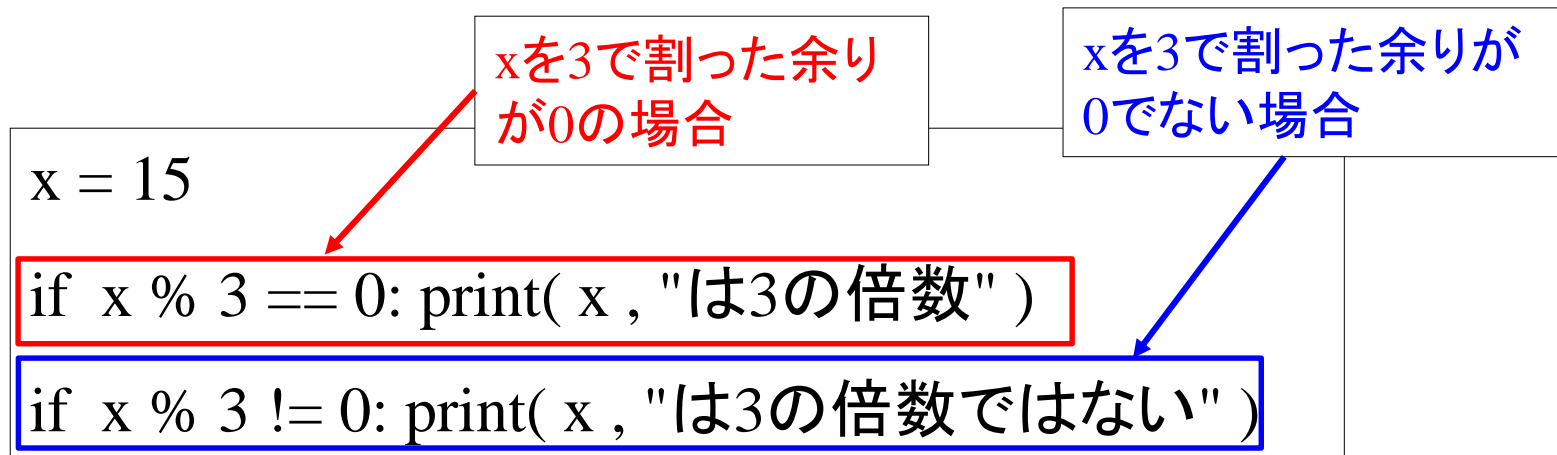
xを2で割った余りが0  
の場合

xを2で割った余りが0  
の場合

```
>>> x = 5
>>> if x % 2 == 0: print( x , "は偶数" )
>>> if x % 2 != 0: print( x , "は奇数" )
5 は奇数
```

xを2で割った余りは0でないので奇数

# 条件文⑤



```
>>> x = 15
>>> if x % 3 == 0: print( x , "は3の倍数" )
15 は3の倍数
>>> if x % 3 != 0: print( x , "は3の倍数ではない" )
```

こちらの文が実行される

## 条件文⑥

year = 2019

yearを4で割った  
余りが0の場合

yearを4で割った余  
りが0でない場合

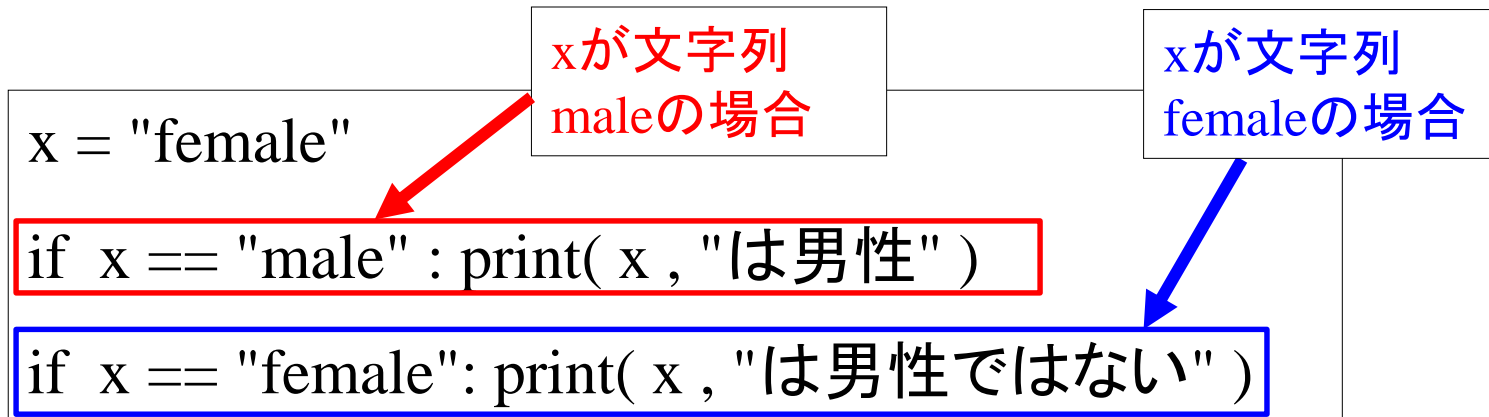
if year % 4 == 0: print( year , "年は閏年" )

if year % 4 != 0 : print( year , "年は閏年ではない" )

```
>>> year = 2019
>>> if year % 4 == 0: print( year , "年は閏年" )
>>> if year % 4 != 0 : print( year , "年は閏年ではない" )
2019 年は閏年ではない
```

こちらの文が実行される

# 条件文⑦



```
>>> x = "female"
>>> if x == "male": print( x , "は男性" )
>>> if x == "female": print( x , "は男性ではない" )
female は男性ではない
```

こちらの文が実行される

## 条件文⑧

```
x="abcde"
```

```
if len(x) > 3 : print( "x > 3" )
```

```
if len(x) <= 3 : print( "x <= 3" )
```

文字列xの長さが  
3より大きい場合

文字列xの長さが  
3以下の場合

```
>>> x="abcde"
```

```
>>> if len(x) > 3 : print( "x > 3" )
```

```
x > 3
```

```
>>> if len(x) <= 3 : print( "x <= 3" )
```

こちらの文が実行される

## 条件文⑨

```
x="abcde"
```

```
if len(x) > 3 : x=x.upper(); x=x*2
```

```
if len(x) <= 3 : x=x+"efg"; x=x*2
```

```
print(x)
```

複数の文を実行したい  
場合、「;」でつなげる

```
>>> x="abcde"
```

```
>>> if len(x) > 3 : x=x.upper(); x=x*2
```

```
>>> if len(x) <= 3 : x=x+"efg"; x=x*2
```

```
>>> print(x)
```

```
ABCDEABCDE
```

こちらの文が実行される

# Pythonプログラム

- ◆ Python のプログラムは, 「文」を並べたもの
- ◆ 「文」一個でもプログラムです
  - print 何とか, if何とかも「文」です
- ◆ 次回からは複数個の「文」=何かの動作をするプログラムを作成していきます



# 練習問題

- ◆ 練習問題①～⑤までは頑張って行なって下さい
- ◆ 余力のある人は練習問題⑥～⑧も行なって下さい

# 練習問題①

① 以下の式の値を求める式を書きなさい.

$$\sin(\pi / 4)^2 + \cos(\pi / 4)^2$$

② x秒 (xは整数)をh時間m分s秒に変換する文を書きなさい.

- ◆ x=10,000(秒)の場合, h=2(時間), m=46(分), s=40(秒)です

## 練習問題②

- ◆ 次の論理式の結果はどうなるでしょうか.
- ◆ 実行する前に考えて下さい.

①  $3 + 2 > 5$

②  $3 + 2 > 4 - 2$

③  $3 + 2 > 4 - 2$  and  $3 + 2 > 5$

④  $3 + 2 > 4 - 2$  or  $3 + 2 > 5$

⑤  $5 > 2$  and  $1 > 2$  or  $3 > 2$

⑥  $5 > 2$  and  $1 > 2$  and  $3 > 2$

## 練習問題③

- ①整数型の変数  $x$  の値が5の倍数の場合は,  $x$  を10倍し, 5の倍数でない場合, 100倍し印字する条件文をそれぞれ書きなさい.

$x=12$

上記の条件文を書きなさい

## 練習問題③

- ②文字列型の変数xの長さが5以上の場合は、変数xを2回繰り返し、5未満の場合は10回繰り返して印字する条件文をそれぞれ書きなさい

文字列型の変数の長さ `len(x)`

文字列型の変数をN回繰り返し `x*N`

`x="Python"`

上記の条件文を書きなさい

## 練習問題④

- ◆ 整数型の変数  $x$ ,  $y$  において,  $x$  と  $y$  の大小を判定する条件文を書きなさい.
  - ◆  $x$  の方が大きい場合は, " $x$ の方が大きい"
  - ◆  $y$  の方が大きい場合は, " $y$ の方が大きい"
- と印字する条件文をそれぞれ書きなさい(等しい場合は無視してもよい)

$x=32$

$y=16$

上記の条件文を書きなさい

# 練習問題⑤

- 変数  $x$  と  $y$  の値を入れ替えたい. なぜ, 次の式では目的を達成できないのか?

$$x = y$$

$$y = x$$

- 代入文を3回用いて,  $x$ と $y$ の値を入れ替えなさい.

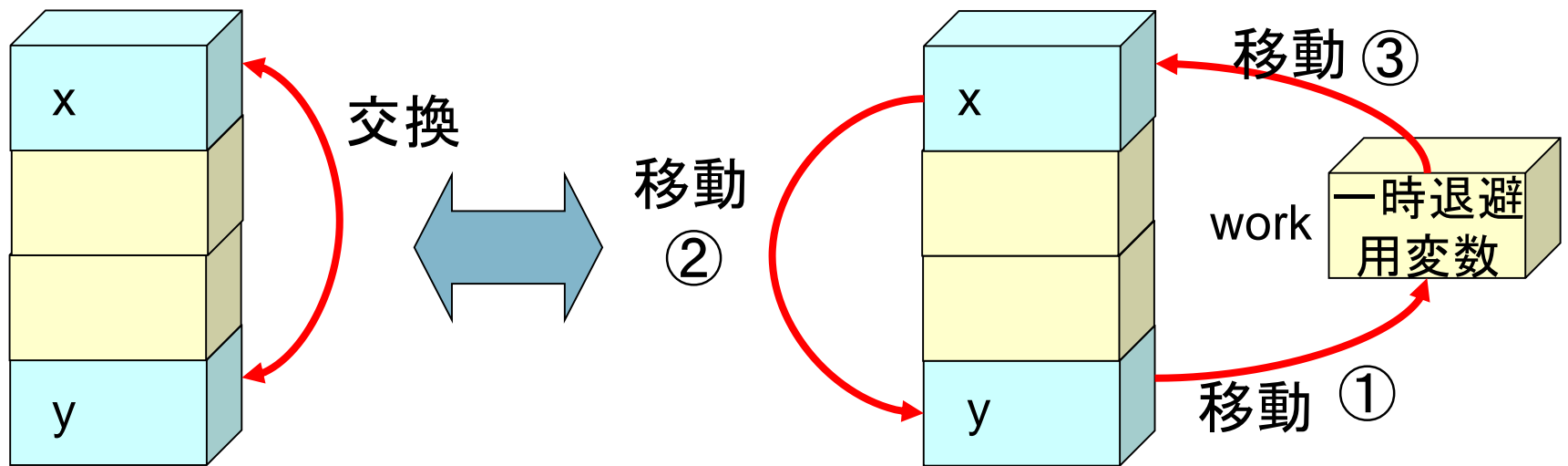
$x=24$

$y=16$

$x$ と $y$ の値を入れ替えるため,  
3個の代入文を書きなさい

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

# 変数の値の交換





## 練習問題⑥

- ◆  $x$ は二桁の整数とします. 10の位の数字が1の位の数字よりも大きい場合は,  $x$ を二倍し, そうでない場合は,  $x$ を10倍して印字する条件文をそれぞれ書きなさい.
- ◆ (等しい場合は無視してもよい)
- ◆  $x=36$

上記の条件文を書きなさい

# 練習問題⑦

- ◆ x年が閏年か判定する条件文を書きなさい(ここでは, 閏年とは4で割り切れる年とします)
- ◆ 閏年の場合は, xを印字し, 閏年でない場合は, 次の閏年を印字する条件文をそれぞれ書きなさい
- ◆ (スライド中に判定する条件文はあります)

x=2019

上記のif式を書きなさい

## 練習問題⑧

- 変数  $x$  と  $y$  に整数が代入されている.  $x$  と  $y$  のうち, 小さいものを  $x$  に, 大きいものを  $y$  に入れたい. どうすればいいか?

$x = 15$

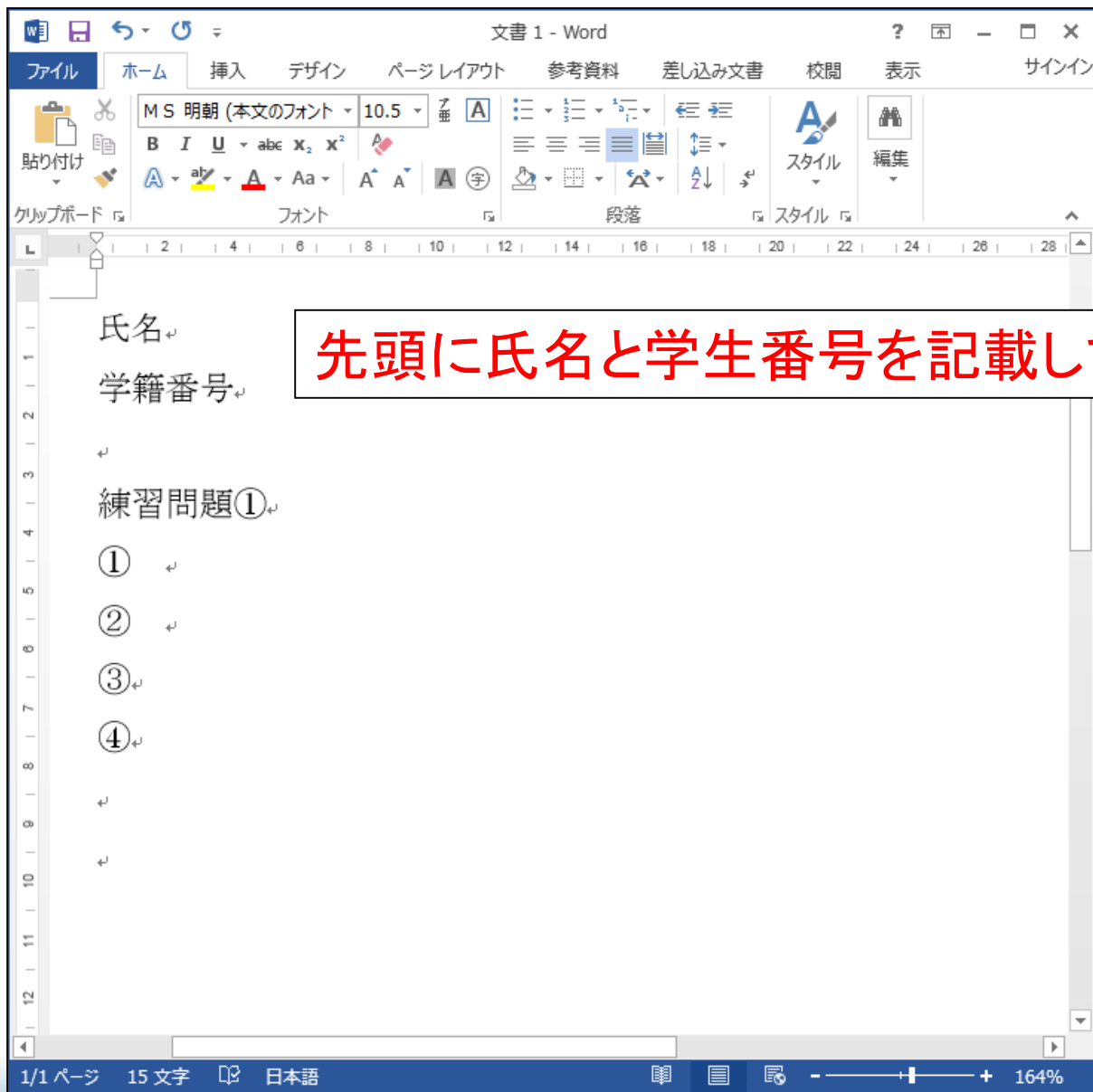
$y = 10$

上記の条件文(1個)を書きなさい

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

# 提出方法

- ◆ 回答をMS-Wordで作成して下さい.
- ◆ 先頭に氏名と学籍番号を書いて下さい.
- ◆ 提出方法
  - keio.jp 上から作成したファイルを「第三回講義練習問題」に提出して下さい.
  - MS-Wordファイルのままでけっこうです.
  - (PDFファイルには変換しないで下さい)
  - 練習問題の締め切りは、当日の18時とします.



先頭に氏名と学生番号を記載して下さい

ファイル名は自由につけて下さい