Ｐｙｔｈｏｎ入門

ＩＴのチカラで研究を支援

amelieff

アメリエフ株式会社

本講義にあたって

テキストが穴埋めになっています

埋めて完成させてください

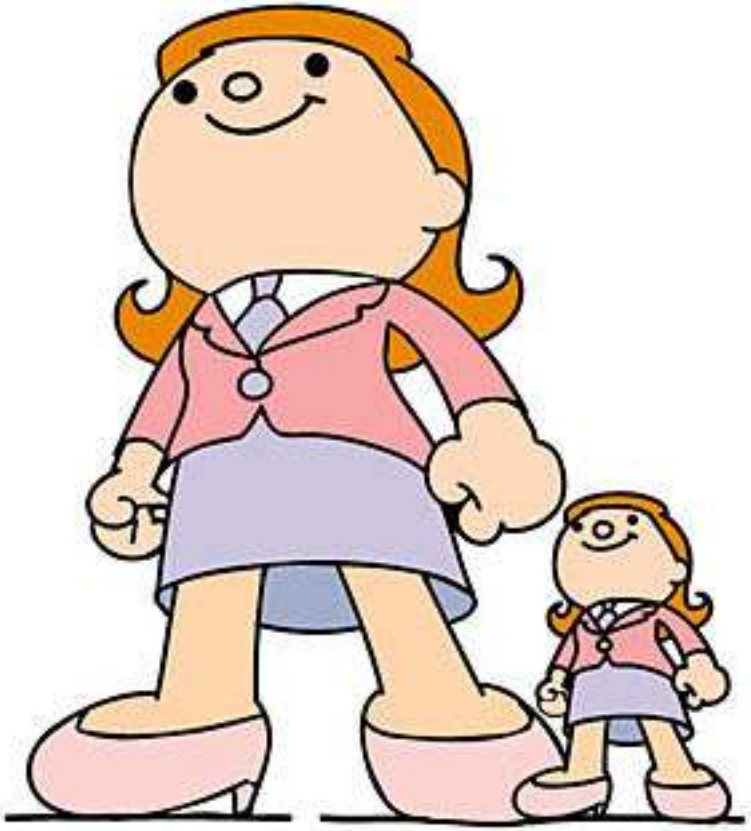
クイズがたくさんあります

めざせ全問正解！

実習がたくさんあります

とにかく書いてみるのが理解の早道です

Pythonが導く
明るい未来



Pythonが導く明るい未来

あなたは解析担当者です
Perlを使ってバリバリ仕事しています
共同研究者から一本の電話がかかってきました

頼んでたPerlのスクリプトだけど
Pythonで書いてくれる？

Ｐｙｔｈｏｎが導く明るい未来

PythonはPerlと同じスクリプト言語で
Perlとよく似ています

（Perlじゃダメ
なんですか…？）

Ｐｙｔｈｏｎが導く明るい未来

その時です

Pythonのほうが
他人が見てもわかりやすい
スクリプトが書けるよ！

Ｐｙｔｈｏｎが導く明るい未来

PythonはPerlより「 文法にうるさい 」
誰が書いても似たスクリプトになります

このため他人のスクリプトでも
「 理解しやすい 」
という利点があります

Ｐｙｔｈｏｎが導く明るい未来

Perlとの使いわけの提案です

他人に渡す
スクリプト
→**Python**

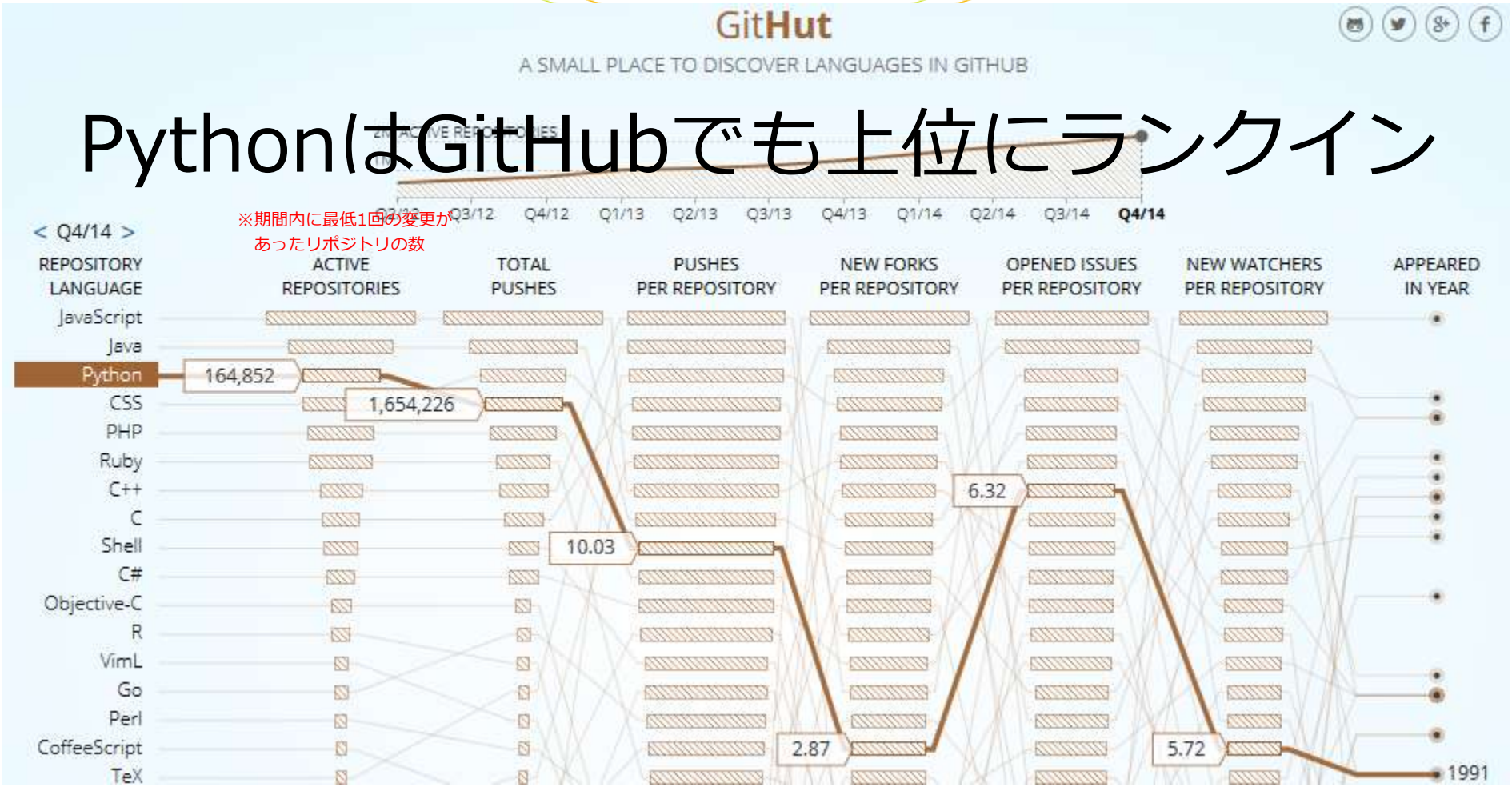
ルールに則った
公式文書

自分しか使わない
or 1回しか使わない
スクリプト　→**Perl**

気軽に書ける
プライベートな手帳

Ｐｙｔｈｏｎが導く明るい未来

PythonはGitHubでも上位にランクイン



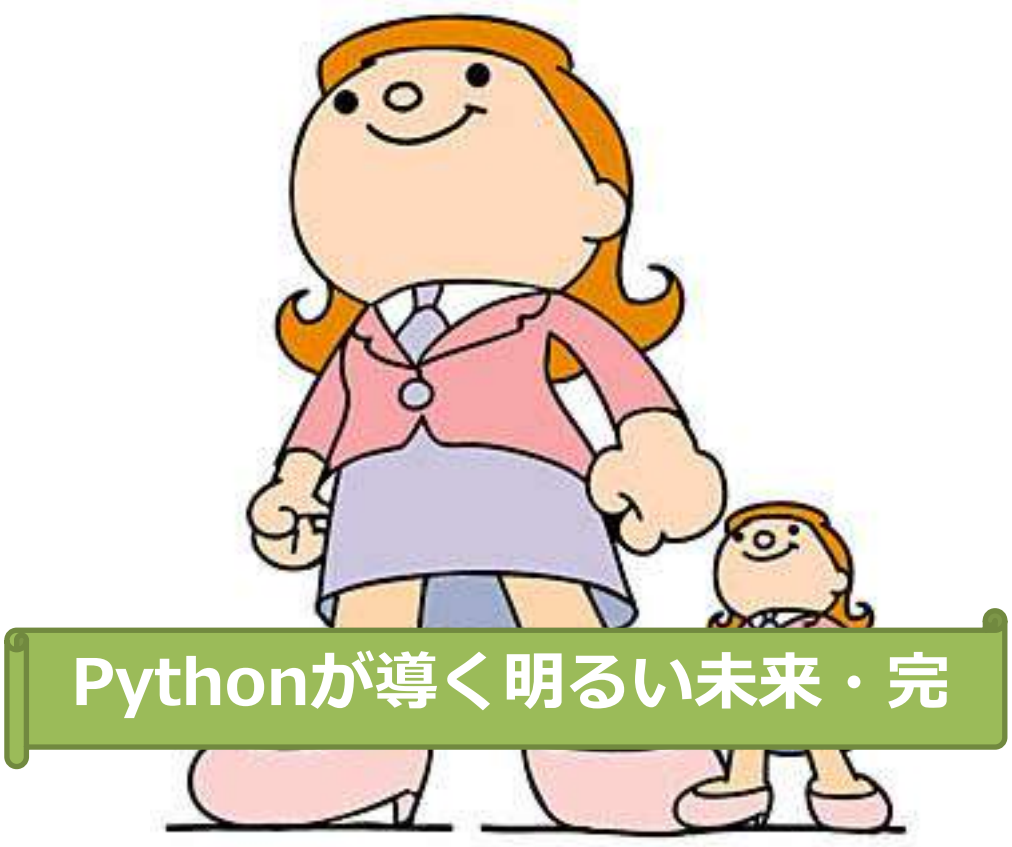
http://githut.info/ (2015/05/14時点)

Pythonが導く明るい未来

それに、Perlが書ける人なら
Pythonの勉強も捗ると思うよ！

じゃあPython
やってみようかな

Perlよりちょっと厳しい。
だからみんなに優しい。



本講義の内容

PerlとPythonの比較

文法の話
- リストとタプル
- 辞書
- ファイル入出力
- 関数
- ライブラリ

Ｐｙｔｈｏｎのバージョン

Pythonの2と3は大きく仕様が変わっています

本テキストはなるべくPython2でも3でも動くような記述にしました

※Biolinux8にはPython2もPython3も入っています

Python2と3で仕様が変わっているところには説明を入れるようにしました

Pythonのバージョン

BioLinux8では

Python2を実行するには「`python`」コマンドを
Python3を実行するには「`python3`」コマンドを
使います

```
$ python foo.py

# Python2で実行される
```

```
$ python3 foo.py

# Python3で実行される
```

実習では、どちらも試してみてください

実習環境

1. 仮想環境を起動します
2. デスクトップに「python」ディレクトリを作成します

```
$ cd ~/Desktop
$ mkdir python
$ cd python
```

本日の実習はすべてこの中で行います

実習環境

テストデータ

デスクトップの「Sample Data」から以下の1ファイルを「python」にコピーしてください

「../S」だけ入力してTabキーを押すと「Sample¥ Data」まで入ります

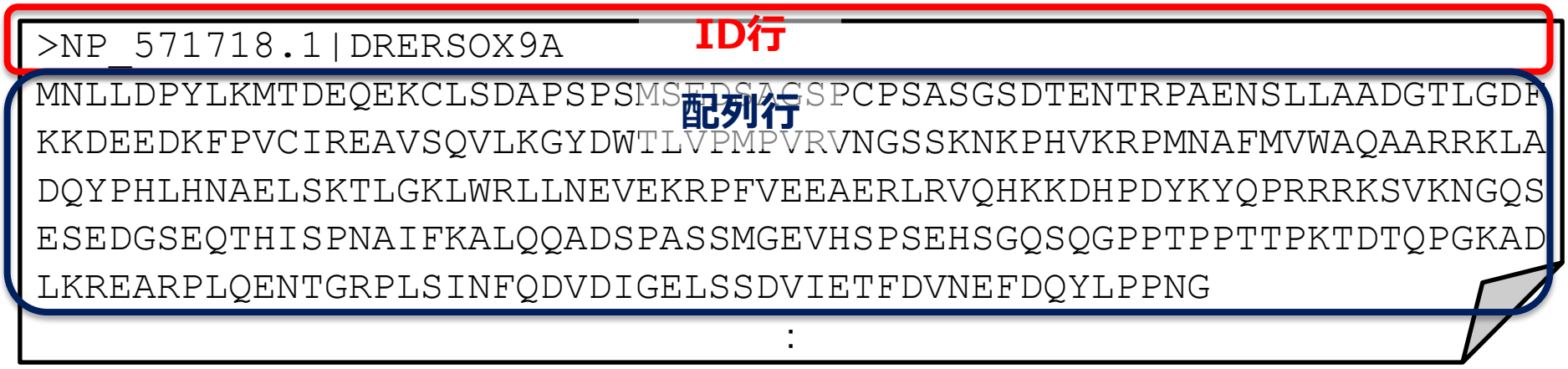
```
$ cp ../Sample¥ Data/peptide_seqs/peptides_longer_headers.fasta .
```

改行を入れずに続けて入力

Fastaフォーマットのファイルです

Fastaフォーマット

＞で始まるID行と配列行（塩基またはアミノ酸）から成るフォーマットです
ゲノムや遺伝子の配列を表すのによく使われます



本講義の達成目標

以下の作業をPythonスクリプトで実行できるようになります

「BioPythonを用いてFastaファイルを操作できる」

PythonとPerlの比較

例：変数iが5未満なら"S"、5以上なら"L"と出力せよ
　　Perlではいろいろな書き方ができ、どれも正解

○整頓したい派

```
if ( $i < 5 ) {
  print "S¥n";
}
else {
  print "L¥n";
}
```

○1行に収めたい派

```
if($i<5){print "S¥n"}else{print "L¥n"}
```

```
if       ($i
<           5   )  {
 print   "S¥n"          ;
}else{     print"L¥n";}
```

○無頼派

PythonとPerlの比較

例：変数iが5未満なら"S"、5以上なら"L"と出力せよ
　　PythonではPerlほどの自由度が無い

○Python正解例

```
if i < 5:
    print("S")
else:
    print("L")
```

ブロックの中は
先頭を半角4文字
下げる

×条件文の中身は
インデントしないと
エラーになる！

```
if i < 5:
print("S")
else:
print("L")
```

※Python2ではPerlに近い「print "S"」という書き方もできますが
本資料ではPython3と互換性のある「print("S")」を用います

PythonとPerlの比較

- 行末に「;」がつくのがPerl、つかないのがPython
- 変数名に$、@、%がつくのがPerl、つかないのがPython
- ブロック構造（繰り返し処理や条件分岐）を {}で囲むのがPerl、「:」とインデントで示すのがPython

```
if ( $i < 5 ) {
  print "S¥n";
}
else {
  print "L¥n";
}
```

```
if i < 5:
    print("S")
else:
    print("L")
```

Pythonの「print」関数はデフォルトで
末尾に改行をつけます

データ型

Pythonにはいろいろなデータ型があります

文字列、数値、論理値、…

「type」関数でデータ型を確認できます

「 str 」関数で数値を文字列に変換できます

```
a = 'a'
b = 1
print(type(a))
print(type(b))
c = str(b)
print(type(c))
```

「**str(文字列)**」であることが示される

「**int(整数型の数値)**」であることが示される

「**str(文字列)**」であることが示される

実習１

次のPythonスクリプト・py1.pyを書いて実行してみましょう

Hello!と出力するPythonスクリプトです

```
$ gedit py1.py
```

py1.pyにこの2行を書いて保存し
右のコマンドを実行します

```
print('Hello!')
print(type('Hello!'))
```

```
$ python py1.py
Hello!
<type 'str'>
```

```
$ python3 py1.py
Hello!
<class 'str'>
```

演算子

Pythonの主な演算子

算術演算子

A + B	足し算
A - B	引き算
A * B	掛け算
A / B	割り算
A % B	剰余
A ** B	べき乗

比較演算子

A == B	A＝Bなら
A != B	A≠Bなら
A < B	A＜Bなら
A <= B	A≦Bなら
A >= B	A≧Bなら
A > B	A＞Bなら

+は文字列にも数値にも使える

文字列を結合する

```
a = 'ameli'
b = 'eff'
c = a + b
```

c = 'amelieff'

数値を足し算する

```
a = 1
b = 2
c = a + b
```

c = 3

クイズ

実行結果は
どうなりますか？

```
gene  = 'p53 '
value = 20

result = gene + value
print(result)
```

A

```
p53
20
```

B

```
p53 20
```

C

```
p5320
```

D

エラーになる

クイズ

正解は、**D**！！　エラーになる

```
Traceback (most recent call last):  File "Q1.py", line 4, in <module>    result = gene + valueTypeError: cannot concatenate 'str' and 'int' objects
```

文字列と数値はそのままでは結合できません
以下のように、数値を文字列に変換すれば結合できます

```
result = gene + str(value)
```

条件付き処理

条件の後ろの「:」を忘れないこと！

```
if 条件1:
    条件1を満たした時の処理
elif 条件2:
    条件1は満たさなかったが、
    条件2を満たした時の処理
else:
    どの条件も満たさなかった
    時の処理
```

処理の前のインデントを忘れないこと！

参考：Perlの場合

```
if(条件1){
   条件1を満たした時の処理
}
elsif(条件2){
   条件1は満たさなかったが、
   条件2を満たした時の処理
}
else{
   どの条件も満たさなかった
   時の処理
}
```

繰り返し処理

```
for 要素 in リストやタプル:
    各要素に対する処理
```

例

```
gene_arr = ["Oct4", ¥
"Sox2", "Kif4", "c-Myc"]

for gene in gene_arr:
    print(gene)
```

参考：Perlの場合

```
my @gene_arr = ("Oct4",
"Sox2", "Kif4", "c-Myc");

for my $gene(@gene_arr){
  print $gene, "¥n";
}
```

難易度：★★

クイズ

- エラーが出るのはどれでしょう

（複数回答可）

A

```
a=1
if a == 1:
    print('a is 1')
else:
    print('a is not 1')
```

B

```
a=1
if a == 1:
    print('a is 1')
else
    print('a is not 1')
```

C

```
for i in (1, 2, 3):
    print(i)
```

D

```
for i in (1, 2, 3):
    print(i):
```

クイズ

正解は、**B、D**！！

B

```
a=1
if a == 1:
    print('a is 1')
else
    print('a is not 1')
```

「else」のあとにも「:」が必要です

D

```
for i in (1, 2, 3):
    print(i):
```

「for」のあとには「:」が必要ですが、その中の実行文には必要ありません

リストとタプル

Perlの配列にあたるものが
Pythonのリストとタプルです

「リスト」は値の変更や追加が可能ですが
「タプル」は値の変更や追加ができません

→ 一回定義したら値が変わらないものや、
辞書のキーにするものはタプルにすると良い

初心者は、迷ったらとりあえずリストを使っておけばいいと思います

リスト

リストxの作成

```
x = [2, 4, 7, 3]
```

要素をコンマ区切りで[]内に羅列します

リストxの1番目の要素をyに代入

```
y = x[0]
```

添字は0スタート

リストxの2番目の値を「5」に変更

```
x[1] = 5
```

リストの操作

リストxの末尾に新しい要素を追加

```
x.append(8)
```

リストxの最初の値を削除

```
del x[0]
```

タプル

リストは[]で
タプルは()です

タプルxの作成

```
x = (2, 4, 7, 3)
```

タプルxの1番目の要素をyに代入

```
y = x[0]
```

※タプルは値の変更や追加ができません

リストとタプル

リストとタプルの違いの覚え方（©服部）

初めに入れた値を変えたいんだけど…

かく（角）も柔軟、リストくん

[^ o ^] ｲｲﾖｲｲﾖｰ

丸いが頑固なタプルちゃん

(＊`Д´;) ﾀﾞﾒ! ｾﾞｯﾀｲ

リスト・タプルの共通操作

リスト/タプルxの要素数

```
len(x)
```

※ちなみにlen(文字列)で文字列長が得られます

リスト/タプルxの最小値と最大値

```
max(x)
min(x)
```

リスト・タプルの共通操作

リスト/タプルxの各値を出力

```
for y in x:
    print(y)
```

リスト/タプルxに値'4'があるか調べる

```
if 4 in x:
    print('yes')
```

リスト・タプルの相互変換

リストxをタプルに変換

```
x = [2, 4, 7, 3]
x = tuple(x)
type(x)
```

タプルxをリストに変換

```
x = (2, 4, 7, 3)
x = list(x)
type(x)
```

実習2

次のPythonスクリプト・py2.pyを書いて実行してみましょう

リストxを作って最初の値を出力するPythonスクリプトです

```
$ gedit py2.py
```

py2.pyにこの2行を書いて保存し実行します

```
x = [1, 2, 3]
print(x[0])
```

```
$ python py2.py
```

1と出力されればOK

実習３

次のPythonスクリプト・py3.pyを書いて実行してみましょう

タプルxを作って最後の値を出力するPythonスクリプトです

```
$ gedit py3.py
```

py3.pyにこの3行を書いて保存し実行します

```
x = (1, 2, 3)
index = len(x) - 1
print(x[index])
```

```
$ python py3.py
```

3と出力されればOK

クイズ

以下の実行結果を得るためには、どのスクリプトを実行すればよいでしょう？

実行結果

```
Hello
Python
```

A

```
for w in ['Hello', 'Python']:
    print(w)
```

B

```
for w in ['Hello', 'Python']:
print(w)
```

C

```
for w in ['Hello', 'Python']
    print(w)
```

クイズ

正解はA！

```
Hello
Python
```

エラー！
for文の末尾に「:」がない

A

```
for w in ['Hello', 'Python']:
    print(w)
```

```
Hello
Python
```

C

```
for w in ['Hello', 'Python']
    print(w)
```

B

```
for w in ['Hello', 'Python']:
print(w)
```

エラー！
forの中身がインデントされていない

【参考】改行をつけなくする

Python2の場合：,をつける

```
print w,
```

Python3の場合：end=""をつける

```
print(w,end="")
```

クイズ

実行結果は
どうなりますか？

```
a = ['A', 'B', 'C']
b = ('D', 'E', 'F')
a.append('G')
print(len(a))
print(b[1])
```

A

```
3
D
```

C

```
4
E
```

B

```
3
F
```

D

```
4
D
```

クイズ

正解は、**C**！！

```
4
E
```

```
a = ['A', 'B', 'C']
b = ('D', 'E', 'F')
a.append('G')
print(len(a))
print(b[1])
```

a = ['A', 'B', 'C', 'G']

b = ('D', 'E', 'F')

コメント

行の頭に「#」をつけるとその行はコメントとなり処理に無関係になります

```
# This is comment.
```

複数行に渡るコメントを書くには、「"」または「'」を3つ連続で書いたもので囲います

```
"""
These are
comment lines.
"""
```

難易度：★★

クイズ

- 下のスクリプトを実行した後、
  リスト a の要素のうち 'spam' はいくつでしょうか。

```
a = ['egg','spam','pork','tomato']
a.append('spam')
#a[1]='bean'
del a[2]
a = a + ['bacon','spam']
if len(a) > 4:
    '''
    a[1] = 'tomato'
    '''
```

A
2

B
3

C
4

D
5

クイズ

正解は、 **3個！！**

```
a = ['egg','spam','pork','tomato']
a.append('spam')
#a[1]='bean' 何も起きない
del a[2]
a = a + ['bacon','spam']
if len(a) > 4:
    '''
    a[1] = 'tomato' 何も起きない
    '''
```

```
['egg','spam','pork','tomato']
['egg','spam','pork','tomato','spam']
['egg','spam','tomato','spam']
['egg','spam','tomato','spam','bacon',
 'spam']
```

答： `['egg','spam','tomato','spam','bacon','spam']`

辞書

Perlのハッシュにあたるものが
Pythonの辞書（ディクショナリ）です

キーと値をセットで格納します

辞書

キー:値をコンマ区切りで{}内に羅列します

辞書xの作成

```
x = {'a':1, 'c':4}
```

辞書xのキーが'a'の値をyに代入

```
y = x['a']
```

辞書xのキーが'c'の値を「5」に変更

```
x['c'] = 5
```

辞書の操作

辞書xに新しいキーと値を追加

```
x['b'] = 10
```

辞書xにキー「'a'」が含まれるか調べる

```
if 'a' in x:
```

辞書xの要素数

```
len(x)
```

実習４

次のPythonスクリプト・py4.pyを書いて実行してみましょう

```
x = {'even':0}
for i in [1,2,3,4,5]:
    if i % 2 == 0:
        x['even'] += 1

print('even=' + str(x['even']))
```

$ python py4.py

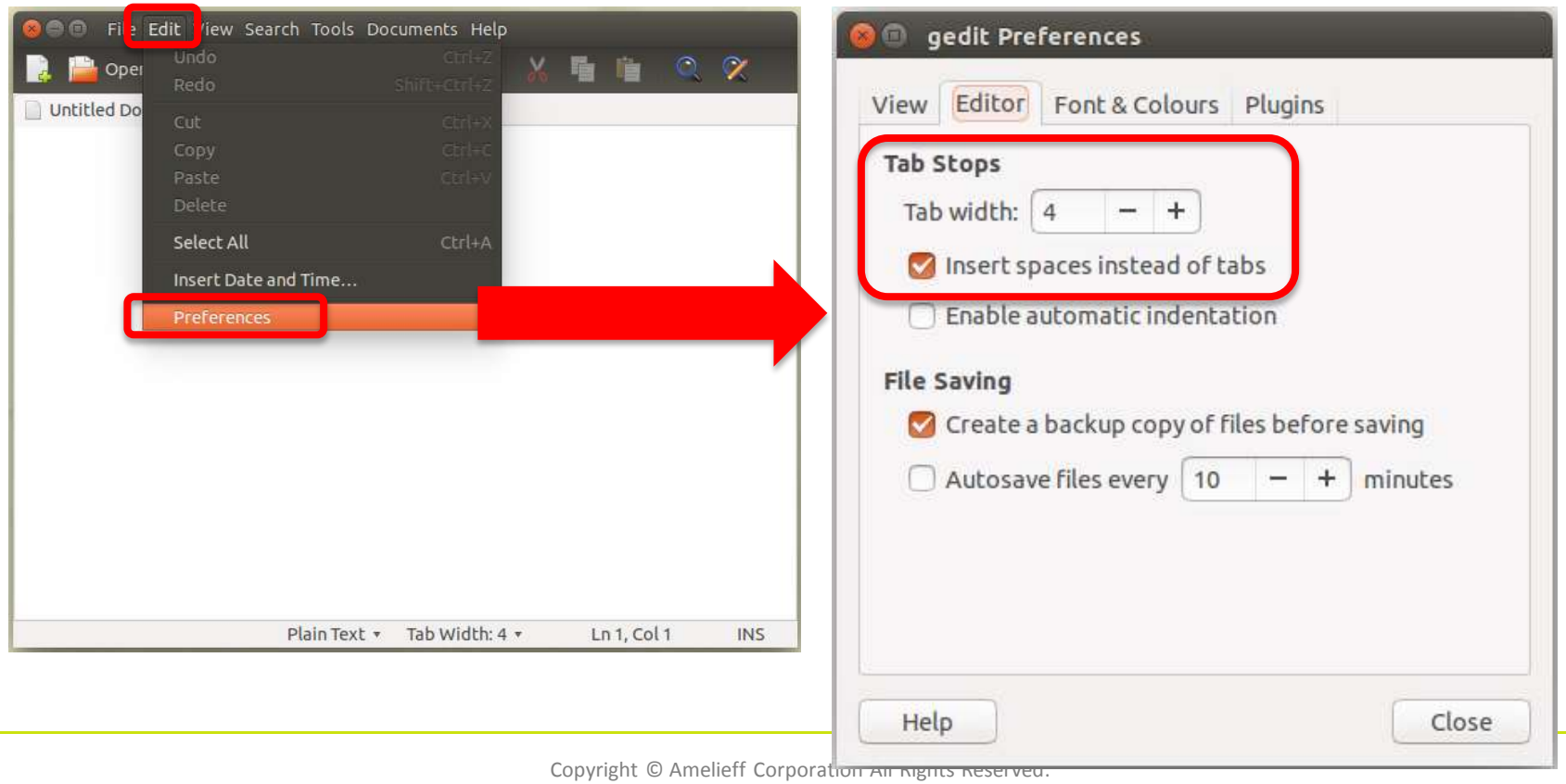
even=2と出力されればOK

a+= 1はa = a +1と
同じ意味になります

・1から5までの間の偶数の数を数えるスクリプトです
・余裕のある方はelseを使って奇数（odd）の数も数えてみましょう

geditで楽にインデントする方法

Tabキーを一回押した時に半角スペースを4つ入れるように設定できます



クイズ

実行結果は
どうなりますか？

```
orange = {'price': 1, 'stock': 30}
orange['stock'] = 'sold out'
print(orange['stock'])
```

A

30

C

orange

B

エラーになる

D

sold out

クイズ

正解は、**D**！！

```
sold out
```

辞書の値には、

文字列も使用可能です

```
orange = {'price': 1, 'stock': 30}
orange['stock'] = 'sold out'
print(orange['stock'])
```

クイズ

実行結果は
どうなりますか？

```
orange = {'price': 1, 'stock': 30}
# add 20 oranges
orange['stock'] += 20
for key in orange:
    print(key+' : '+str(orange[key]))
```

A

```
key : value
```

B

```
1
50
```

C

```
price : 1
stock : 50
```

D

```
1
20
```

クイズ

正解は、C！！

```
price : 1
stock : 50
```

```
orange = {'price': 1, 'stock': 30}
# add 20 oranges
orange['stock'] += 20
for key in orange:
    print(key+' : '+str(orange[key]))
```

itemsメソッドを使うと

辞書のキーと値を

セットで取得できます

```
orange = {'price': 1, 'stock': 30}
# add 20 oranges
orange['stock'] += 20
for key, value in orange.items():
    print(key+' : '+str(value))
```

値の整形

formatを使うと、printの中に値を埋め込んだり、数値の桁数を揃えたりできます

```
print('My name is {0}'.format("Alice"))
```

```
My name is Alice.
```

```
print('My name is {0} {1}.'.format("Alice", "Liddell"))
```

```
My name is Alice Liddell.
```

```
print('I have {0} yen.'.format(100))
```

```
I have 100 yen.
```

```
print('π is {0:.2f} '.format(3.1415926535))
```

```
π is 3.14
```

クイズ

辞書からキーと値を取得したいです。

どう書けばいいでしょうか？

```
orange = {'price': 1, 'stock': 30}
orange['stock'] += 20
_____________________:
    _____________________________
```

?

```
price : 1
stock : 50
```

A

```
for key in orange:
    print('{0} : {1}'.format(key, orange[key]))
```

B

```
for key, value in orange.items():
    print('{0} : {1}'.format(key, value))
```

クイズ

どちらも**正解**！！

```
orange = {'price': 1, 'stock': 30}
orange['stock'] += 20
for key, value in orange.items():
    print('{0} : {1}'.format(key, value))
```

文字列の挿入は、**format**メソッドで指定します
formatの引数は、順に**0**, **1**に挿入されます

Ｐｙｔｈｏｎのお作法：ＰＥＰ８

http://pep8-ja.readthedocs.org/ja/latest/

１行の文字数を長くしすぎない

PEP8では、1行あたり79文字（コメントは72文字）までにすることが推奨されています
行の途中で改行するには、改行位置に¥を書きます

```
longTale = 'When I was a child, I often went' ¥
           + 'swimming with my father.'
                                :
```

PEP8に準拠しているかチェック・修正する

pep8：スクリプトがPEP8に準拠するかチェックする

autopep8：スクリプトをPEP8に準拠した内容に修正する

```
// pipのインストール
$ wget https://raw.github.com/pypa/pip/master/contrib/get-pip.py
$ sudo python get-pip.py

// pep8とautopep8のインストール
$ sudo pip install pep8
$ sudo pip install autopep8

// pep8およびautopep8の実行
$ pep8 py4.py
$ autopep8 -I py4.py
```

関 数

関数は、オブジェクト（引数）を受け取って、処理を行い、結果（戻り値）を返します

関数は以下のように定義します

```
def 関数名(引数):
    処理内容
    return 戻り値
```

関数内はインデントする

関 数

関数の例

```
def tashizan(num1, num2):
    result = num1 + num2
    return result

test = tashizan(1, 2)
print(test)
# 3
```

二つの数値を受け取り
合計を返す関数

実習5

次のPythonスクリプト・py5.pyを書いて実行してみましょう

```
def U_to_T(nuc):
    if nuc == 'U':
        return 'T'
    else:
        return nuc

print(U_to_T('U'))
```

```
$ python py5.py
```

Tと出力されればOK

実習6

次のPythonスクリプト・py6.pyを書いて実行してみましょう

```
def U_to_T(nucs):
    result = ''
    for nuc in nucs:
        if nuc == 'U':
            result = result + 'T'
        else:
            result = result + nuc
    return result

print(U_to_T('AUGC'))
```

```
$ python py6.py
```

ATGCと出力されればOK

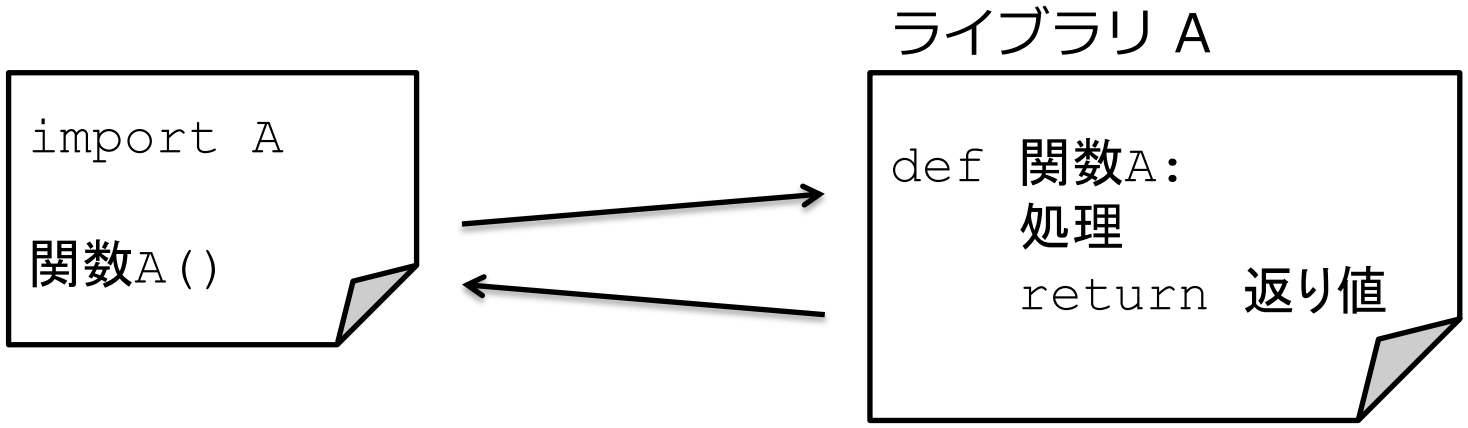
ライブラリ

ライブラリとは、Pythonスクリプトを他のPythonスクリプトから呼び出せるようにしたものです

Pythonではたくさんのライブラリが提供されています

ライブラリをインストールすると

import関数で呼び出して使うことができます



ライブラリ

ライブラリのインストールはライブラリ付属のsetup.pyや、ライブラリ管理コマンド（pip、easy_installなど）で行います

プラットフォーム	標準のインストール場所	デフォルト値
Unix (pure)	*prefix*/lib/pythonX.Y/site-packages	/usr/local/lib/pythonX.Y/site-packages
Unix (non-pure)	*exec-prefix*/lib/pythonX.Y/site-packages	/usr/local/lib/pythonX.Y/site-packages
Windows	*prefix*\Lib\site-packages	C:\PythonXY\Lib\site-packages

http://docs.python.jp/2.7/install/index.html

コマンドライン引数

コマンドラインから引数を受け取るには
sysライブラリをimportします

```
import sys
```

引数は、リスト sys.argv に入ります
sys.argv[0]：スクリプト名
sys.argv[1]：1つ目の引数
sys.argv[2]：2つ目の引数
：

スクリプトを終了する

スクリプトを終了するには sys.exit() を実行します

正常終了時は sys.exit(0)
エラー終了時は sys.exit("エラーメッセージ")

実習7

次のPythonスクリプト・py7.pyを書いて実行してみましょう

```
import sys

if len(sys.argv) < 2:
  sys.exit("Less argument.")

arg1 = sys.argv[1]
print("1st: " + str(arg1))
```

```
$ python py7.py
```

```
$ python py7.py A
```

シバンと日本語化

コード内で日本語（マルチバイト文字）を使用するには、テキストエンコーディングを指定します

一般的には、ソースコードの2行目に以下のように記述します（エンコーディングはUTF-8にしておけば問題ありません）

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

BioPython

BioPythonはバイオインフォマティクスのライブラリです

http://biopython.org/wiki/Main_Page
http://biopython.org/DIST/docs/tutorial/Tutorial.html

BioLinux8には最初からインストールされています

BioPythonでできること

- ファイルの操作

  Blast、Clustalw、FASTA、GenBank、PubMed、UniGene、 ...

- オンラインサービスへのアクセス

  Blast、Entrez、PubMed、Swiss-Prot、Prosite

- プログラムの実行

  Blast、Clustalw、EMBOSS command line tools

  など多数

BioPythonのimport

Bioパッケージから SeqIOインタフェースを読み込む

```
from Bio import SeqIO
```

Fastaファイルを読み込んで各配列のIDと塩基配列を出力する

```
for seq_record in SeqIO.parse(Fastaファイル, "fasta"):
    print(seq_record.id)
    print(seq_record.seq)
```

ファイル入出力

ファイル'a.txt'から読み込んでリストlinesに入れる（linesには1行ずつが要素として入る）

```
f = open('a.txt', 'r')
lines = f.readlines()
f.close()
```

openの一つ目にファイル名、
二つ目にmodeを指定します

r:　読み込み用
w:　書き込み用
a:　追記用

ファイル'a.txt'に'A'と書き込む

```
f = open('a.txt', 'w')
f.write('A')
f.close()
```

最終課題

次のPythonスクリプト・
py8.pyを書いて実行してみましょう

1. コマンドライン引数で与えたFastaファイルを変数「in_fasta」に読み込む
2. 各配列の「ID」と「配列」と「配列長」を出力する

```
$ python py8.py peptides_longer_headers.fasta
```

最終課題

```
import sys
from Bio import SeqIO

in_fasta = sys.argv[1]

for seq_record in SeqIO.parse(in_fasta, "fasta"):
    print("ID:      " + seq_record.id)
    print("Seq:     " + seq_record.seq)
    print("Length: " + str(len(seq_record.seq)))
```

```
$ python py8.py peptides_longer_headers.fasta
```

参考資料

（Perl入門の最終課題をPythonで書いたら）

1. コマンドライン引数で指定したファイルを読込用で開いて、1行ずつ読み込んで改行コードを削除する
2. 読み込んだ行がID行以外なら、一文字ずつ区切って各アミノ酸の出現頻度をハッシュでカウントする
3. カウント結果を出力
4. コマンドライン引数にpeptides_longer_headers.fastaを与えて実行

参考資料

（Perl入門の最終課題をPythonで書いたら）

```perl
#!/usr/bin/perl

use strict;
use warnings;
use autodie;

my $file = $ARGV[0];
my %aaCount;

open my $fh, "<", $file;
while(<$fh>){
  chomp;
  if($_ !~ /^>/){
    my @aaArr = split(//, $_);
      for my $aa(@aaArr){
        $aaCount{$aa} ++;
      }
  }
}
close  $fh;

while(my ($aa, $count) = each %aaCount){
  print $aa, ":", $count, "¥n";
}
```

```python
#!/usr/bin/python

import sys

file = sys.argv[1]
aaCount = {}


f = open(file, 'r')
lines = f.readlines()
f.close()

for line in lines:
    if not line.startswith(">"):
        aaArr = list(line.rstrip("¥n"))
        for aa in aaArr:
            if aa not in aaCount:
                aaCount[aa] = 0
            aaCount[aa] += 1

for aa, count in aaCount.items():
    print(aa + ":" + str(count))
```

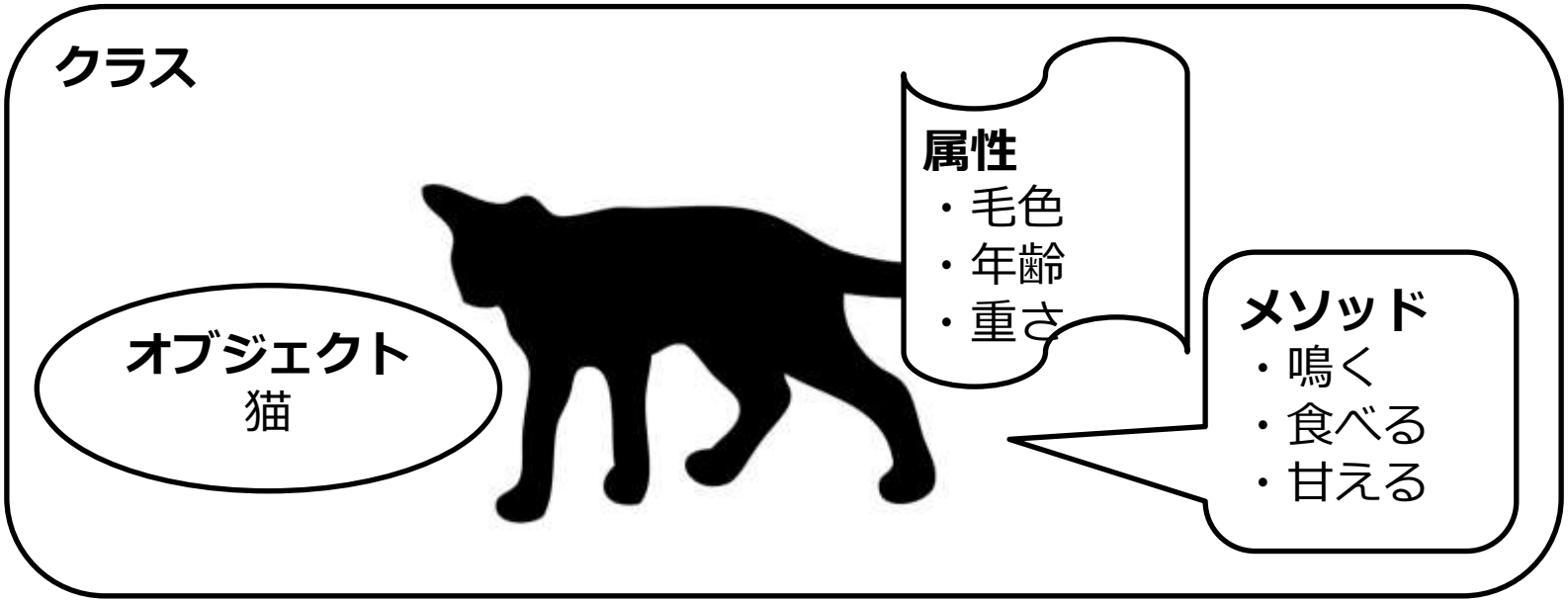
オブジェクト指向とは

プログラミングの書き方の一つで、
データを「 属性 」や「 メソッド 」を持つ
「オブジェクト」として扱います

Pythonはオブジェクト指向に対応した言語として開発されました

オブジェクト指向とは

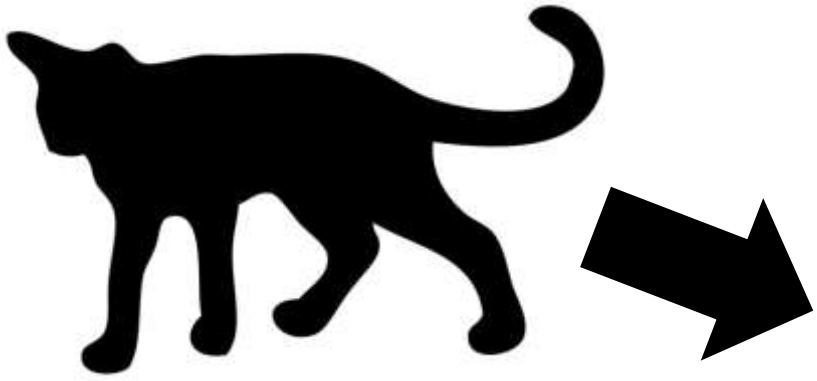
オブジェクト＝属性とメソッド
クラス＝オブジェクトの定義



オブジェクト指向とは

クラスを実現化したもの＝インスタンス

クラス



インスタンス

実習9

次のPythonスクリプト・py9.pyを書いて実行してみましょう

```
class Cat():
    def setName(self, name):
        self.name = name
    def getName(self):
        return self.name

mycat = Cat()
mycat.setName("Tom")
print mycat.getName()
```

```
$ python py9.py
```

指定した名前が出力されればOK

現在いる場所を確認する【pwd】

現在Linuxのどのディレクトリにいるか確認するには次のコマンドを実行します

```
$ pwd
```

コマンドを入力した後、Enterキーを押すとコマンドが実行されます

ディレクトリ内を確認する【ls】

現在いる場所にどのようなファイル・ディレクトリがあるか確認するには次のコマンドを実行します

```
$ ls -l
```

-lをつけて実行するとlsだけを実行するより詳しい結果が表示されます（アクセス権限など）
-lを「オプション」と呼びます

他のディレクトリに移動する【cd】

他のディレクトリに移動するには次のコマンドを実行します

```
$ cd 移動先ディレクトリ
```

コマンドとオプションの間、コマンドと値の間には半角空白を1つ以上入れます

ディレクトリを作成する【mkdir】

```
$ mkdir 移動先ディレクトリ
```

ファイルを作成する【touch】

```
$ touch 作成するファイル名
```

ファイルを閲覧するにはlessやmore、
ファイルを編集するにはgeditやviを使います

ファイルを編集する【gedit】

```
$ gedit 編集するファイル名
```

ファイルが存在しない場合は新規作成されます
GUI環境がない場合はviを使います

ファイルまたはディレクトリをコピーする【cp】

```
$ cp ファイル名|ディレクトリ名 コピー先名
```

ファイルまたはディレクトリを移動する【mv】

```
$ mv ファイル名|ディレクトリ名 コピー先名
```

アクセス権限を変更する【chmod】

```
$ chmod 付与する権限 ファイル名|ディレクトリ名
```

権限の例）755：全員に読み書き実行を許可、700：所有者のみに読み書き実行を許可

主な解凍コマンド

拡張子	圧縮形式	コマンド
.tar.gz	gzip	$ tar zxvf *ファイル名*
.tar.bz2	bzip2	$ tar jxvf *ファイル名*
.gz	gzip	$ gunzip *ファイル名*
		$ gzip -d *ファイル名*
.bz2	bzip2	$ bunzip2 *ファイル名*
		$ bzip2 -d *ファイル名*
.zip	zip	$ unzip*ファイル名*
.tar	tar	$ tar xvf *ファイル名*

Linuxのテキストエディタ

GUIのエディタとCUIのエディタがあります

GUI：Windows/Macソフトのように、マウスで操作する

長所：Linux初心者にも操作が容易

短所：GUIがない環境では使えない

CUI：キーボードからコマンドで操作する

長所：GUIがない環境でも使える

短所：操作コマンドを覚える必要がある

gedit

CentOSにはデフォルトでgeditというGUIエディタが入っています

geditを起動するには `$ gedit` コマンドを実行します

vi

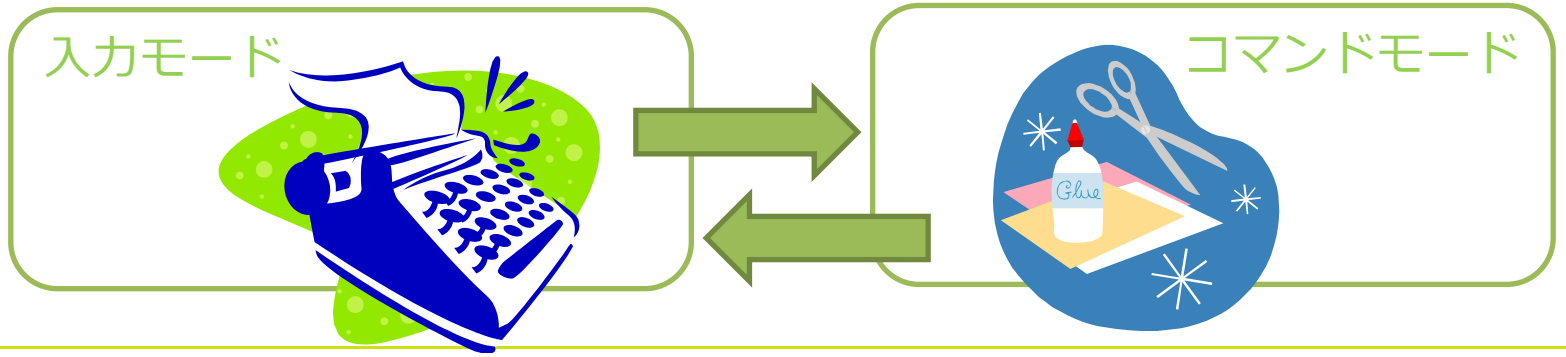
CentOSにはデフォルトでviというCUIエディタが入っています

viを起動するには `$ vi` コマンドを実行します

viには2つのモードがあり、モードを切り替えながら操作します

入力モード：文字を入力する

コマンドモード：編集する（切り貼り、ファイルの保存など）



vi

入力モードのコマンド

Escキー	コマンドモードに移行

コマンドモードのコマンド

a	入力モードに移行（カーソルの右から入力）
o	入力モードに移行（次の行の行頭から入力）
x	1文字カット
dd	今いる行をカット
yy	1 行コピー
p	カットした行をペースト
[数字]g	[数字]行に移動
G	最終行に移動
:%s/*foo*/*bar*/	文字列置換（*foo*を*bar*に置換）