この資料の使い方

最初から順に試してください。

初心者は、書かれたとおり入力して、ほかにも似た例を試してみてください。

初級者は、書かれたとおり入力して、なぜそれで動くのか調べてください。

中級者は、設問から解答を考えてください。【根性】問題も挑戦してください。

上級者は、【根性】問題も含めて、この資料の改善をしてください。

【根性】問題は、やる気と（多少の）知識がある人に挑戦して欲しいものです。

1.起動と実行

1.1 コマンドラインから Python を起動せよ。

コマンドプロンプト(MS-DOS プロンプト)を起動する。

環境変数 PYTHONHOME を設定する。

環境変数 PATH を設定する。

```
c:¥home>set PYTHONHOME=C:¥Python24
c:¥home>set PATH=%PATH%;%PYTHONHOME%
c:¥home>python
Python 2.4 (#60, Nov 30 2004, 11:49:19) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

1.2 Hello World を実行せよ。

```
>>> "Hello World"
'Hello World'
>>>
```

1.3 電卓として使ってみる。

四則演算: +, -, *, /

剰余: %

乗数(exp): **

```
>>> 1+2
3
>>> 1*2*3*4*5
120
>>> 100/3
33
```

```
>>> 100/3.0
33.333333333333336
>>> 42%8
2
>>> 2**10
1024
```

1.4 インタプリタを終了せよ。

```
>>> ^Z
c:¥home>
```

1.5 変数に値を入れて使用せよ。

変数の宣言は不要(できない)

変数は型を持たない。

【根性】abs(),hex(),ord(),pow()を使え。

http://www.python.jp/doc/nightly/lib/built-in-funcs.html を参照せよ。

```
>>> a=10
>>> b=20
>>> a*b
200
>>> c=a+b
>>> c
30
>>> c=a/b
>>> c
0
>>> a="Hello"
>>> b="World"
>>> c=a+b
>>> c
'HelloWorld'
>>>
```

1.6 すべてがオブジェクトであることを確認せよ

数字や文字列もオブジェクトである。

dir()でオブジェクトの属性を調べられる。

help()でドキュメンテーションを表示できる。

```
>>> a=10
>>> dir(a)
['__abs__', '__add__', '__and__', '__class__', …(省略)
>>> s="abcde"
>>> dir(s)
['__add__', '__class__', '__contains__', …(省略)… 'capitalize', 'center',
'count', 'decode', …(省略)
>>> help(s.capitalize)
Help on built-in function capitalize:

capitalize(...)
   S.capitalize() -> string

   Return a copy of the string S with only its first character
   capitalized.
>>> s.capitalize()
'Abcde'
```

2. ソースファイルとプログラム

2.1 Hello World をソースファイルから実行せよ。

ソースファイルの拡張子は .py

表示は print 文を使う。

最初の行に文字コードのおまじないを入れる。

```
----------hello.py
# -*- coding: windows-31j -*-
print 'Hello World'
----------
```

以下のように実行。

```
C:¥temp>python hello.py
Hello World
```

```
C:¥temp>
```

Windows だと、ソースファイル名を直接実行できる。

```
C:¥temp>hello.py
Hello World

C:¥temp>
```

2.2 1 から 100 までの和を計算せよ。

ドキュメントは http://www.python.jp/doc/nightly/参照。

1 から 100 までのリストを作り、ループで和を計算する。

連続した数字のリストは range()で作れる。

range については、上記から「チュートリアル」→「4.3 range()関数」を見よ。

ループは for <要素> in <リスト> を使う。

for については、上記から「チュートリアル」→「4.2 for 文」を見よ。

インタプリタ上で試してからソースコードに落とせ。

Windows ならば、スタートメニューから「すべてのプログラム」→「Python 2.4」→「Python Manuals」を見て、上記ドキュメントと比較せよ。

【根性】1 から 100 までの積を計算せよ。

【根性】3 種類の異なるプログラムを作成せよ。

```
----------sum100.py
# -*- coding: windows-31j -*-

nums = range(101)
sum = 0
for n in nums:
        sum += n
print sum
----------
```

2.3 2.2 のロジックを関数にして、実行せよ。

100 まで固定ではなく、合計する範囲を引数で指定せよ。

関数定義は def を使う。

インデントを揃えよ。タブと空白に注意。

「チュートリアル」→「4.6 関数を定義する」

```
----------sum100_func.py
```

```
# -*- coding: windows-31j -*-

def sum(max):
    nums = range(max + 1)
    sum = 0
    for n in nums:
        sum += n
    return sum

print sum(100)
----------
```

2.4 Person クラスを作れ。

姓と名をそれぞれ変数で、氏名の表示をメソッドで実装せよ。
クラスは class で定義する。
コンストラクタは__init__()で定義する。
self は省略できない。
main()の書き方を理解せよ。

```
----------person.py
# -*- coding: windows-31j -*-
import sys
import os

class Person:
    def __init__(self, firstname, lastname):
        self.firstname = firstname
        self.lastname = lastname

    def print_name(self):
        print "My name is " + self.firstname + "  " + self.lastname

def main():
    p = Person("やっ", "とむ")
    p.print_name()

if __name__ == '__main__':
```

```
    main()
---------
```

2.5 Person クラスに生年月日を持たせ、現在の年齢を計算せよ。

datetime 標準モジュールを使え。
モジュールは import 文でロードする。
使用するときはモジュール名で修飾する。
インタプリタ上で十分実験せよ。
help()を使え。
dir()を使え。
http://www.python.jp/doc/nightly/lib/module-datetime.html

```
----------person_age.py
# -*- coding: windows-31j -*-
import datetime

class Person:
    def __init__(self, firstname, lastname, birthday):
        self.firstname = firstname
        self.lastname = lastname
        self.birthday = birthday

    def print_name(self):
        print "My name is " + self.firstname + "  " + self.lastname

    def get_age(self, at=datetime.date.today()):
        bd = self.birthday
        age = at.year - bd.year
        if (at.month, at.day) <= (bd.month, bd.day):
            age -= 1
        return age

def main():
    p = Person("やっ", "とむ", datetime.date(1973,4,20))
    p.print_name()
    print "age: ", p.get_age()
```

```
if __name__ == '__main__':
   main()
----------
```

2.6 クラスに文字列表現能力を持たせよ

str()と repr()関数で文字列変換する。
内部では__str__()と__repr()__メソッドで変換される。
__repr__()では、再構築できる文字列に変換すると便利。
【根性】obj == eval(repr(obj))を確認せよ。
【根性】その他の特別なメソッドを調べよ。dir()を使え。

```
>>> class Name:
...     def __init__(self, name):
...             self.name = name
...
>>> n = Name("Yattomu")
>>> repr(n)
'<__main__.Name instance at 0x00B4B5D0>'
>>> str(n)
'<__main__.Name instance at 0x00B4B5D0>'
>>> class Name:
...     def __init__(self, name):
...             self.name = name
...     def __repr__(self):
...             return "Name('"+self.name+"')"
...
>>> n = Name("Yattomu")
>>> repr(n)
'Name('Yattomu')'
>>>
```

3. 開発環境

3.1 PyDev を使ってみよ。

Eclipse プラグインである。

3.2 IDLE を使ってみよ。

標準で添付されている。

3.3 PyCrust を使ってみよ。

wxPython に添付されている。

4. シーケンス

4.1 シーケンスの中身を 1 つずつ取り出して表示せよ。

ﾘｽﾄ、文字列、タプルで試せ。

range()と xrange()で試せ。

```
>>> l=[1,2,3,4,5]
>>> for e in l:
...     print e
...
1
2
3
4
5
>>> s='12345'
>>> for c in s:
...     print c
...
1
2
3
4
5
>>> t=(1,2,3,4,5)
>>> for e in t:
...     print e
...
1
2
3
4
```

```
5
>>>
```

4.2 キーボードから入力したものをシーケンスにせよ

リスト、文字列で試せ。

キーボードからの入力には raw_input()を使え。

```
>>> l=[]
>>> for i in range(5):
...     l.append(raw_input('INPUT>'))
...
INPUT>1
INPUT>2
INPUT>3
INPUT>4
INPUT>5
>>> l
['1', '2', '3', '4', '5']
>>> for i in range(5):
...    s += raw_input('INPUT>')
...
INPUT>1
INPUT>2
INPUT>345
INPUT>6
INPUT>
>>>
>>> s
'123456'
```

4.3 リストを内包表記で生成せよ。

[x for x in …]でリストを生成できる。

【根性】(x for x in …)でジェネレータを生成できる。

生成したものを確認せよ。

```
>>> l=[x*x for x in range(5)]
>>> l
[0, 1, 4, 9, 16]
```

4.4 テキストファイルからデータを読み込め。

fileinput モジュールを使え。

読み込んだものをリストにせよ。

【根性】コマンドライン引数指定のファイルを読み込め。help(fileinput)を参照せよ。

```
----------foo.txt
foo
bar
bazz
----------

>>> import fileinput
>>> i = fileinput.input('foo.txt')
>>> for l in i:
...     print l
...
foo

bar

bazz

>>>
```

4.5 ディクショナリを使ってみよ。

ディクショナリは{key:value, key:value, …}で作成する。

d[key]で値にアクセスできる。

【根性】すべてのキー、すべての値をアクセスせよ。

```
>>> d={'name':'Yattomu', 'age':32, 105:69, 'Girlfriend':None}
>>> d
{'Girlfriend': None, 105: 69, 'age': 32, 'name': 'Yattomu'}
>>> print d[105]
69
>>> print d['Girlfriend']
None
```

```
>>> print d['bonus']
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
KeyError: 'bonus'
>>> d['bonus']=1000000
>>> d
{'Girlfriend': None, 105: 69, 'age': 32, 'bonus': 1000000, 'name': 'Yattomu'}
>>> d.get('bonus')
1000000
```

5. クラスとオブジェクト

5.1 [根性] オブジェクトの等価性と同一性を調べよ。

==では等価性を検査する。

is で同一性を検査する。

等価性の検査のためには、__eq__()を実装する。

```
>>> class Foo:
...     def __init__(self, num):
...             self.num = num
...     def __eq__(self, other):
...             if not isinstance(other, Foo):
...                     return False
...             if self.num==other.num:
...                     return True
...             return False
...
>>> f1=Foo(10)
>>> f2=Foo(10)
>>> f3=Foo(20)
>>> f1==f2
True
>>> f1==f3
False
>>> f1===f2
  File "<stdin>", line 1
    f1===f2
```

```
      ^
SyntaxError: invalid syntax
>>> f1 is f2
False
>>> f4 = f3
>>> f3 is f4
True
```

5.2 オブジェクトに属性を追加せよ。

新しい属性として変数を追加せよ。

【根性】新しい属性としてメソッドを追加せよ。メソッドの第１引数は self。クラスに加えよ。

【根性】オブジェクトにメソッドを追加するとどうなるか確認せよ。

```
>>> class Foo:
...     pass
...
>>> f = Foo()
>>> print f.x
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
AttributeError: Foo instance has no attribute 'x'
>>> f.x = 10
>>> print f.x
10
>>> f2 = Foo()
>>> f2.x
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
AttributeError: Foo instance has no attribute 'x'
```

5.3 【根性】演算子をオーバーロードせよ。

四則演算を定義せよ。__add__(),__sub__(),__mul__(),__floordiv__()をオーバーロードして定義せよ。

コンテナして機能できるようにせよ。__len__(),__getitem__(),__setitem__()をオーバーロードせよ。

http://www.python.jp/doc/nightly/ref/specialnames.html

5.4 【根性】標準モジュールを使ってみよ。

sys,os,re,math は基本。

StringIO,wxPython,pickle を使ってみよ。

pydoc を使ってみよ。

http://www.python.jp/doc/nightly/lib/lib.html