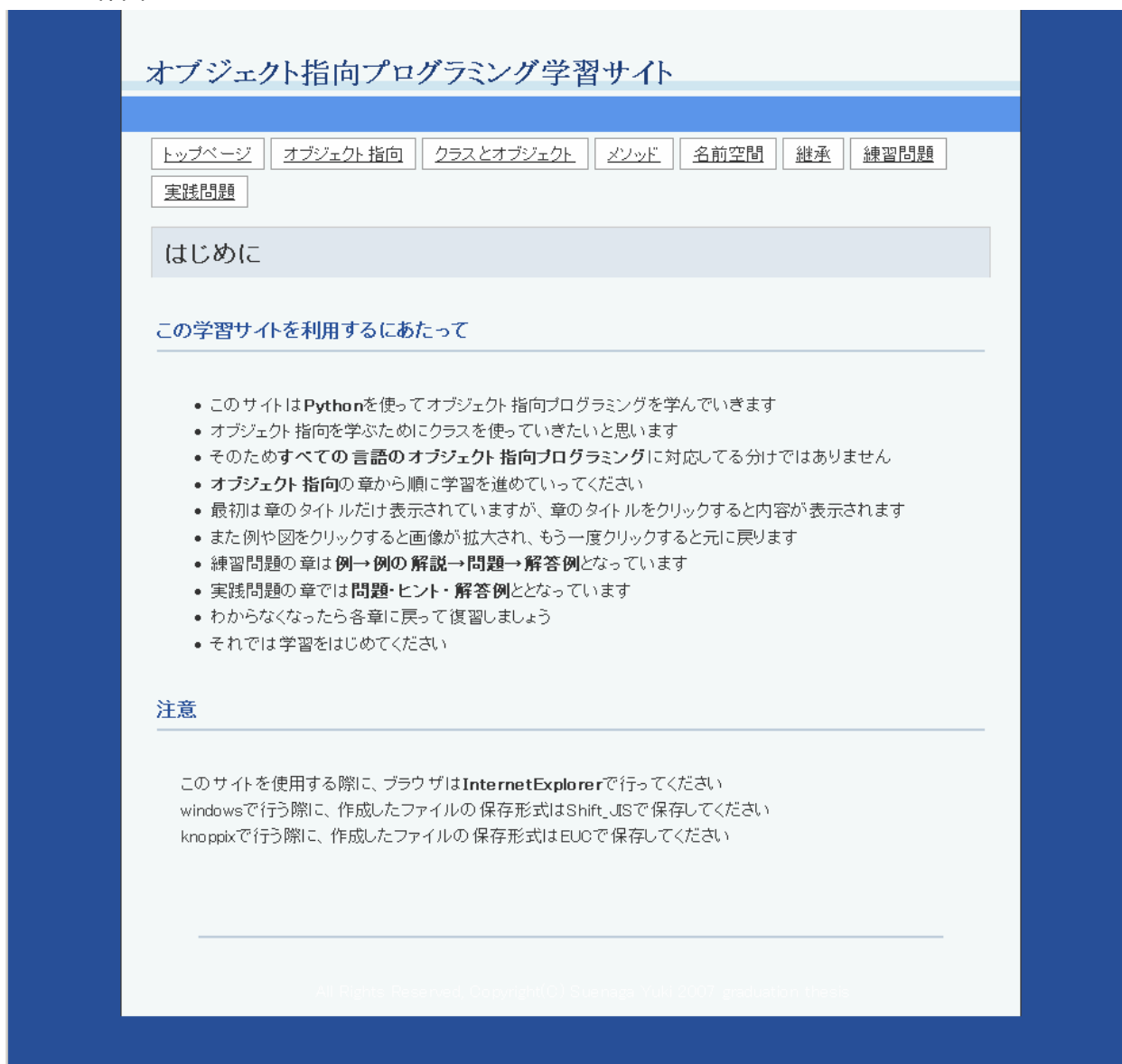


全体図



Chapter1.html(オブジェクトの章)

全体図



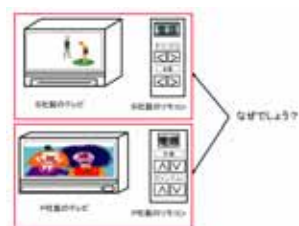
chapter1.html 図 1



chapter1.html 図 2



chapter1.html 図 3



Chapter2.html(クラスとオブジェクトの章)

全体図

オブジェクト指向プログラミング学習サイト

[トップページ](#)
[オブジェクト指向](#)
[クラスとオブジェクト](#)
[メソッド](#)
[名前空間](#)
[継承](#)
[練習問題](#)
[実習問題](#)

クラス

オブジェクト

クラスとオブジェクトに関する問題

[前の章へ](#)
[次の章へ](#)

All Rights Reserved, Copyright(C) Suenaga Yuki 2007 graduation thesis

Chapter2.html 図 1

Chapter2.html 図 2



例 原子のクラス

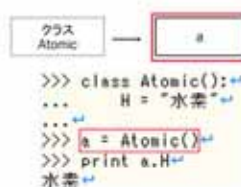
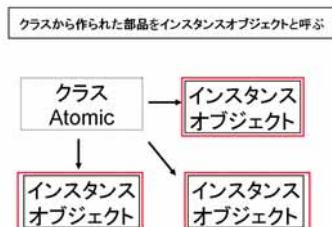
実際の世界で
原子のグループの中には水素が含まれています。
考えやすくするために例のあるものになりました。

```

>>> class Atomic():
...     H = "水素"
...
>>> a = Atomic()
>>> print a.H
水素
  
```

Chapter2.html 図 3

Chapter2.html 図 4



ここではAtomicクラスを変数aの中に入れていました。
これはAtomicという設計図からaという部品を作っていることと同じです。

Chapter2.html 図 5

```

>>> class Atomic():
...     H = "水素"
...
>>> a = Atomic()
>>> print a.H
水素
    
```

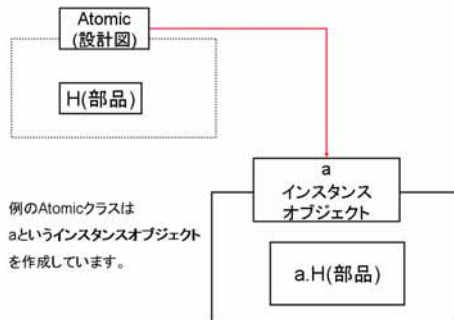
Chapter2.html 図 6

設計図も部品でできている

```

>>> class Atomic():
...     H = "水素"
...
>>> a = Atomic()
>>> print a.H
水素
    
```

Chapter2.html 図 7



Chapter2.html 図 8

```

>>> class Atomic():
...     H = "水素"
...
>>> a = Atomic()
>>> print a.H
水素
    
```

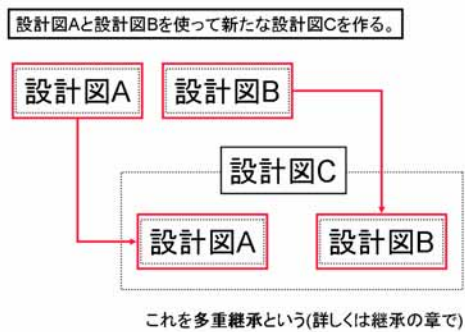
Chapter2.html 図 9

```

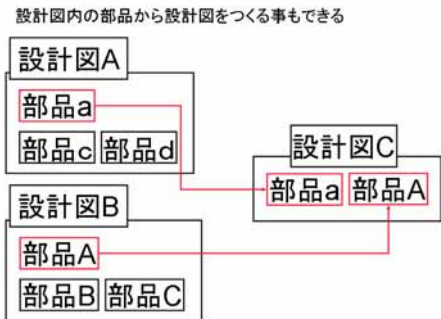
>>> class Atomic():
...     H = "水素"
...
>>> a = Atomic()
>>> print a.H
水素
    
```

原子のクラスで説明すると
 Atomicがクラス名となり
 クラス内の変数Hは
 Atomicクラス内の部品になり。
 クラス内の変数aは
 Atomicクラスから作った部品(インスタンスオブジェクト)になります。

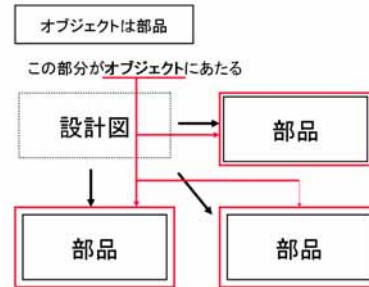
Chapter2.html 図 10



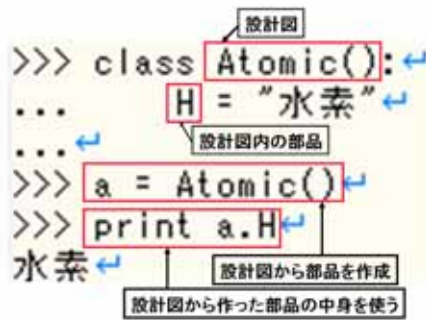
Chapter2.html 図 1 1



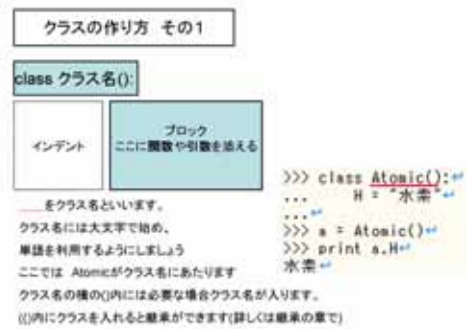
Chapter2.html 図 1 2



Chapter2.html 図 1 3



Chapter2.html 図 1 4



Chapter2.html 問題

クラスとオブジェクトに関する問題

すべて正解できたらのメダルを獲得しましょう

問題1 次の中で設計図(クラス名)にあたるものはどれでしょう？

```
class Atomic:
...     H = "水素"
```

解答: 解答

問題2 次のクラスが動くように空欄に適切な答えをいれてください。

```
class Atomic:
...     H = "水素"
```

解答: 解答

問題3 次の中でオブジェクトはどれでしょう？ 選択性の中から選びなさい

```
class Atomic:
...     H = "水素"
```

a = Atomic()

class

"水素"

a

答えは—

問題3

Chapter3.html

全体図

オブジェクト指向プログラミング学習サイト

トップページ オブジェクト指向 クラスとオブジェクト メソッド 名前空間 継承 練習問題 実践問題

メソッド

メソッドに関する問題

前の章へ 次の章へ

All Rights Reserved, Copyright(C) Suenaga Yuki 2007 graduation thesis

資料編 capter3 図 1



資料編 capter3 図 2

```
クラスの作り方 その2に續いて...
```

```
>>> class Atomic():  
...     def H(self):  
...         print "水素"  
...  
>>> a = Atomic()  
>>> a.H()  
水素
```

クラスの作り方 その2に 續いて
異なるクラスを継承しなした。
AtomicクラスのメソッドHは"水素"という機能をしました。

資料編 capter3 図 3

```
>>> class Atomic():  
...     def H(self):  
...         print "水素"  
...  
>>> a = Atomic()  
>>> a.H()  
水素
```

..... 実行している部分がメソッド名です。
またメソッドは、インスタンスオブジェクトとして使うことができます。

資料編 capter3 図 4



資料編 capter3 図 5

```
メソッドの使い方 2
```

```
>>> class Atomic():  
...     def H(self):  
...         print "水素"  
...  
>>> a = Atomic()  
>>> a.H()  
水素
```

AtomicクラスのメソッドHを呼ぶ場合はメソッドの使い方1の図のよ
うにオブジェクト、メソッド名()で呼ぶ事ができます。
例で言うとa.H()がそれにあたります。
(()を忘れると表示されないので注意しましょう)

資料編 capter3 図 6

メソッドの定義

メソッドを定義するには、関数と同様にdefで使います

```
>>> class Atomic():
...     def H(self):
...         print "水素"
...
...
>>> a = Atomic()
>>> a.H()
水素
```

クラスのブロック内に定義する。

メソッドは、class文から続くブロックの中に定義

必ず最初にselfという引数を定義する

一度目の引数としてselfという引数を添えますselfという引数は、メソッドの中でインスタンスオブジェクト自体を利用したいときに使います。メソッドの名前は、関数名をつける時の同様でしょう。

資料編 capter3 図 7

引数selfとは？

```
>>> class Atomic():
...     def H(self):
...         print "水素"
...
...
>>> a = Atomic()
>>> a.H()
水素
```

先ほどselfは第1引数だと説明しました。ここではHメソッドの第1引数がselfになります。しかしインスタンスオブジェクトで引数を指定していないのに表示できているのはどうしてでしょう？

それはselfがHメソッドの中のインスタンスオブジェクトには"水素"自体をそのまま参照しているからです。

selfという引数には、aがHメソッドを呼び出す時にPythonが自動的に渡してくれているからです。

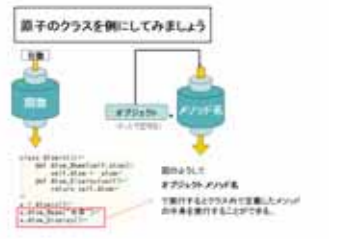
資料編 capter3 図 8

メソッドを複数使った原子のクラス

```
class Atomic():
...     def Atom_Name(self,atom):
...         self.Atom = atom
...     def Atom_Display(self):
...         return self.Atom
...
a = Atomic()
a.Atom_Name("水素")
a.Atom_Display()
```

原子のクラス内のメソッド名は原子の名前とそれを表示するメソッドの二つに分かれました。実際に表示するには以下の例のように書きましょう。

資料編 capter3 図 9



資料編 capter3 図 10

```
class Atomic():
...     def Atom_Name(self,atom):
...         self.Atom = atom
...     def Atom_Display(self):
...         return self.Atom
...
a = Atomic()
a.Atom_Name("水素")
a.Atom_Display()
```

クラスインスタンスを呼んでメソッドを呼び出すために、selfで区切ってメソッド名を記述、呼び出すことができます。

この例ではオブジェクト a.Atom_Display() が実行される

資料編 capter3 図 11

特殊メソッドとは？

Python内で決められたメソッドの名(名前も決まっています)

例 `__init__` `__del__` など

使用例

```
class Atomic():
...     def __init__(self,atom):
...         self.Atom = atom
...     def Atom_Display(self):
...         return self.Atom
...
a = Atomic()
a.Atom_Display()
```

資料編 capter3 図 12

メソッド使用上での注意点

できる限り入力には引数、出力にはreturnステートメントをつかう
ブローカー(変数)名空間の中でどうしても必要な場合は使わない
メソッドの役割は1つに絞る
メソッドはオブジェクトとして渡せる
print,def,returnなどpythonにはじめから入っているコマンドの名前は使えません。
もし使う場合ははじめの文字を大文字にすれば使う事はできます
でもできるだけ使わないようにしましょう

資料編 capter3 問題

メソッドに関する問題

すべて正解したら300ポイントの賞へ進みましょう

問題1 次の中でメソッドはどれでしょう？空欄に書いてください

```
class AtomicO
  -- def HnameID
  -- print "水素"
```

問題2 次のAtomicクラス内にはいくつメソッドがあるでしょう？空欄に書いてください

```
class AtomicO
  -- def HnameID
  -- set H_NAME = H_name
  -- def displayID
  -- return set H_NAME
```

問題3 次のクラス内にはいくつメソッドがあるでしょう？

```
class PGD
  -- def PG_displayID
  -- print "モニター"
```

②
③
④

Chapter4.html(名前空間の章)

全体図

オブジェクト指向プログラミング学習サイト

[トップページ](#)
[オブジェクト指向](#)
[クラスとオブジェクト](#)
[メソッド](#)
[名前空間](#)
[継承](#)
[練習問題](#)
[実践問題](#)

名前空間

名前空間に関する問題

[前の章へ](#)
[次の章へ](#)

All Rights Reserved, Copyright(C) Suenaga Yuki 2007 graduation thesis

Chapter4.html 図 1

グローバル変数 = クラス外で定義する変数

モジュールのトップレベルのブロックで定義されている関数や変数などの名前が定義される変数。プログラムで自由に名前を定義できる。一番上目的変数

```
>>> O = u酸素
>>> H = u水素
>>> class AtomicO:
...     H = u水素
...     O = u酸素
```

ここはクラスの外にある
a,bがグローバル変数になる

Chapter4.html 図 2

ローカル変数 = クラス内で定義する変数

関数の中で定義した変数は、関数のブロックの中だけで有効です。pythonの関数は型宣言する必要がありません。そのため代入が行われると変数が作られるので関数のブロックの中で代入される変数はローカル変数として扱われます。

```
>>> class AtomicO:
...     def displayID():
...         O = u酸素
...         H = u水素
```

ここはクラス内にある
a,bがローカル変数になる

Chapter4.html 図 3

同じ変数名が使える理由

```

>>> H = 氷茶
>>> O = 緑茶
>>> class Atomic():
...     H = すいそ
...     O = さんそ
>>> print H
氷茶
>>> print O
緑茶
>>> print Atomic.H
すいそ
>>> print Atomic.O
さんそ
    
```

机(A) = クラス外の変数を置く場所
机(B) = クラス内の変数を置く場所

Chapter4. 図 4

解説

まずクラス外に変数をおいて置く机(A)があるとします。変数は定義された机の上に並べられます。クラス外では変数を置く机は1つしかありません。クラス内に入ると外とは違う机(B)が用意されます。クラス内で定義された変数は、新しい机に並べられます。クラス内で定義された変数は新しくできた机に並べられます。机が別なので同じ名の変数を定義する事ができる。

Chapter4.html 図 5

アンダースコアの使い方

クラス名には、特別な理由がない限り小文字だけを使うようにします。英単語を区切りたいときにはアンダースコア(_)を使うとよいでしょう。するとビルトイン変数も使うことが可能です。また例のような使い方もできます。

例

```

>>> class class x
... class class o
... class class _o
... class class _o
... class class _o
    
```

```

>>> class class
...     File "stdin", line 1
...     class class
SyntaxError: invalid syntax
    
```

このようなエラーがです

Chapter4. 図 6

アンダースコアを二つつける

アンダースコアを二つつけると外部からのアクセスをできなくします。注意...

使う際には特殊メソッドと同じ名前にならないように気をつけてください。

```

>>> class _Class:
...     _Class = 10
...     _Class = 20
...     _Class = 30
...     _Class = 40
>>> a = _Class()
>>> a._Class
30
>>> a.__Class__
Traceback (most recent call last):
  File "stdin", line 1, in <module>
AttributeError: Class instance has no attribute '__Class__'
    
```

これは、カプセル化を導く

__Class__はクラスの外部からアクセスする事ができない。

Chapter4.html 図 7

名前空間

グローバル変数 = クラス外で定義する変数

ローカル変数 = クラス内で定義する変数

名前空間は三つの変数の区分から成り立っている三つ目にグローバルスコープが存在します。

Chapter4.html 問題

名前空間に関する問題

すべて正解できたら次の継承の章へ進みましょう

問題1 次の中でグローバル変数はどれでしょう？

```
H_NAME = "水素"  
class Atomic():  
...     def H(self, H_Name):  
...         self.H_NAME = H_Name  
...     def display(self, H_Name):  
...         print self.H_NAME  
...
```

解答:

問題2 次のクラスの中でローカル変数はどれでしょう？

```
H_NAME = "水素"  
class Atomic():  
...     def H(self, H_Name):  
...         self.H_NAME = H_Name  
...     def display(self, H_Name):  
...         print self.H_NAME  
...
```

- self.H_Name
- H_Name
- H_NAME

答えは…

閉じる

Chapter5.html(継承の章)

全体図

オブジェクト指向プログラミング学習サイト

[トップページ](#)
[オブジェクト指向](#)
[クラスとオブジェクト](#)
[メソッド](#)
[名前空間](#)
[継承](#)
[練習問題](#)
[実践問題](#)

継承

継承に関する問題

[前の章へ](#)
[次の章へ](#)

All Rights Reserved, Copyright(C) Suenaga Yuki 2007 graduation thesis

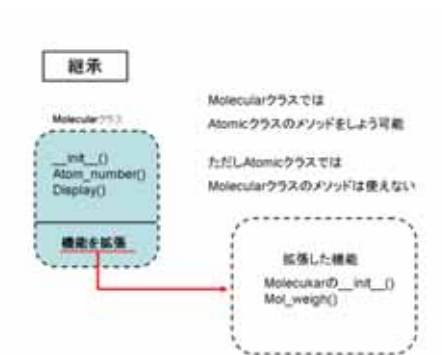
Chapter5 図1

継承で使う2つのクラス
AtomicクラスとMolecularクラス

```

class Atomic():
    def __init__(self, atom_name):
        self.all = "原子の名前: %s No: %d" % (atom_name,
        self.atom_number(self, atom_name):
        self.all += "原子の番号: %s No: %d" % (atom_name,
        def __str__(self):
            print self.all
class Molecular(Atomic):
    def __init__(self, atom_name, mol_name):
        Atomic.__init__(self, atom_name)
        self.all += "物質名: %s" % mol_name
        def Mol_weight(self, mol_name):
            self.all += "分子量: %s" % Mol_weight()
m = Molecular("水", "H2O")
m.Mol_weight()
m.__str__()
    
```

Chapter5 図3



Chapter5 図2

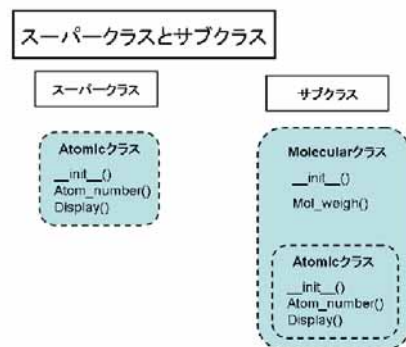
__init__を使う場合

```

class Molecular(Atomic):
    def __init__(self, atom_name, mol_name):
        Atomic.__init__(self, atom_name)
        self.all = self.all + "物質名: %s" % mol_name
    
```

__init__ は一番はじめに読み込む特殊メソッドです。
そのためスーパークラスで__init__を使っている場合は
サブクラスでも__init__を使いましょう。
スーパークラスで使った引数をそのまま
サブクラスの__init__で使えます
サブクラス内で使うには
スーパークラスのクラス名.__init__(引数)
でスーパークラスで使った引数を使うことができます。

Chapter5 図4



Chapter5 図 5

```

class Atom():
    def __init__(self, atom_name):
        self.all = "原子の名前: %s" % (atom_name)
    def atom_number(self, atom_num):
        self.all += "原子の番号: %s" % (atom_num)
    def display(self):
        print self.all

class Molecule(Atom):
    def __init__(self, atom_name, mol_name):
        Atom.__init__(self, atom_name)
        self.all += self.all + "物質名: %s" % (mol_name, atom_name)
    def Mol_weight(self, mol_w):
        self.all += "分子量: %s" % (mol_w)

a = Molecule("水", "H2O")
a.Mol_weight()
a.display()

```

Chapter5 図 6

継承のしかた

継承する場合
 Class クラス名(継承したいクラス):

多重継承する場合
 Class クラス名(継承したいクラスA, 継承したいクラスB):

継承の例
 Class a(b):

多重継承の例
 Class a(b,c):

Chapter5 図 7

```

class Atom():
    def __init__(self, atom_name):
        self.all = "原子の名前: %s" % (atom_name)
    def atom_number(self, atom_num):
        self.all += "原子の番号: %s" % (atom_num)
    def display(self):
        print self.all

class Molecule(Atom):
    def __init__(self, atom_name, mol_name):
        Atom.__init__(self, atom_name)
        self.all += self.all + "物質名: %s" % (mol_name, atom_name)
    def Mol_weight(self, mol_w):
        self.all += "分子量: %s" % (mol_w)

a = Molecule("水", "H2O")
a.Mol_weight()
a.display()

```

Chapter5 問題

継承に関する問題

すべて正解できなかった場合は練習問題を一度お試しください

問題1 次の中でスーパークラスはどれでしょう？

```

class Atom():
    def __init__(self, atom_name):
        self.all = "原子の名前: %s" % (atom_name)
    def atom_number(self, atom_num):
        self.all += "原子の番号: %s" % (atom_num)
    def display(self):
        print self.all

class Molecule(Atom):
    def __init__(self, atom_name, mol_name):
        Atom.__init__(self, atom_name)
        self.all += self.all + "物質名: %s" % (mol_name, atom_name)
    def Mol_weight(self, mol_w):
        self.all += "分子量: %s" % (mol_w)

a = Molecule("水", "H2O")
a.Mol_weight()
a.display()

```

問題1のクラスが継承されているのはどれですか？

正解はこうになります。

class Atom()
 class Molecule
 class Molecule a
 class Molecule b

解説

Chapter6.html(練習問題の章)

全体図

オブジェクト指向プログラミング学習サイト

[トップページ](#) [オブジェクト指向](#) [クラスとオブジェクト](#) [メソッド](#) [名前空間](#) [継承](#) [練習問題](#) [実践問題](#)

練習問題

練習問題2

練習問題3

練習問題5

練習問題4

練習問題6

練習問題5

練習問題7

練習問題6

前の章へ [この章へ](#)
All Rights Reserved. Copyright(C) Sanyo 1941-2007. gsdaron@nifty

練習問題 例 1 (図 1)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
class My_Data():
    def My_Name(self):
        print "自分の名前"
a = My_Data()
a.My_Name()
```

練習問題 解答例 (図 2)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
class My_Data():
    def My_Name(self):
        print "自分の名前をここに記入"
    def My_Nature(self):
        print "自分の性別をここに記入"
a = My_Data()
a.My_Name()
a.My_Nature()
```

練習問題 例 2 - 1 (図 3)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
from class_My_Data import My_Data
a = My_Data()
a.My_Name()
```

練習問題 例 2 - 2 (図 4)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
class My_Data():
    def My_Name(self):
        print "ここに自分の名前を記入"
```

練習問題 2 解答例その 1 (図 5)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
from class_My_Data import My_Data
a = My_Data()
a.My_Name()
a.My_Nature()
```

練習問題 2 解答例その 2 (図 6)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
class My_Data():
    def My_Name(self):
        print "ここに名前を記入してください"
    def My_Nature(self):
        print "ここに性別を記入してください"
```

練習問題 3 例 3 - 1 (図 7)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
from class_My_Data import My_Data
↓
a = My_Data()
a.Write_Nature("男")
a.My_Name()
a.Display_Nature()
```

練習問題 例 3 - 2 (図 8)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
class My_Data():
↓
↓
^   def My_Name(self):
^   ^   print "ここに名前を記述"
^   def Write_Nature(self,My_Nature):
^   ^   self.My_NATURE = My_Nature
^   def Display_Nature(self):
^   ^   print self.My_NATURE
```

練習問題 3 解答例その 1 (図 9)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
from class_My_Data import My_Data
↓
a = My_Data()
a.Write_Nature("男")
a.Write_Blood_Type("AB")
a.My_Name()
a.Display_My_Nature()
a.Display_My_Blood_Type()
```

練習問題 解答例その 2 (図 10)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
class My_Data():
^   def My_Name(self):
^   ^   print "ここに名前を記述"
^   ↓
^   def Write_Nature(self,My_Nature):
^   ^   self.MY_NATURE = My_Nature
^   ↓
^   def Write_Blood_Type(self,My_Blood_type):
^   ^   self.My_BLOOD_TYPE = My_Blood_type
^   ↓
^   def Display_My_Nature(self):
^   ^   print self.MY_NATURE
^   ↓
^   def Display_My_Blood_Type(self):
^   ^   print self.My BLOOD TYPE
```

練習問題 4 例 4 - 1 (図 11)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
↓
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
from class_My_Data import My_Data
↓
a = My_Data("ここに名前を記入")
print a
```

練習問題 4 4 - 1 (図 12)

```
#!/usr/bin/env python
# -*- coding:Shift_JIS -*-
↓
class My_Data():
^   def __init__(self,My_Name):
^   ^   self.My_NAME = My_Name
^   def __str__(self):
^   ^   return self.My_NAME
```

練習問題 4 解答例その 1 (図 1 3)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
←
^   def My_Name(self):←
^   ~   print "ここに名前を記述"←
^   def Write_Nature(self,My_Nature):←
^   ~   self.My_NATURE = My_Nature←
^   def Display_Nature(self):←
^   ~   print self.My_NATURE←
```

練習問題 4 解答例その 2 (図 1 4)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import My_Data←
←
a = My_Data("名前",",","性別",",","血液型")←
print a←
```

練習問題 5 例 5 - 1 (図 1 5)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import My_Data←
←
a = My_Data("名前",",","性別",",","血液型")←
print a←
```

練習問題 5 - 2 (図 1 6)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
←
^   def __init__(self,My_Name,My_Nature,My_Blood_Type):←
^   ~   self.My_NAME = My_Name←
^   ~   self.My_NATURE = My_Nature←
^   ~   self.My_BLOOD_TYPE = My_Blood_Type←
←
^   def __str__(self):←
^   ~   return self.My_NAME + self.My_NATURE + self.My_BLOOD_TYPE←
```

練習問題 5 解答例その 1 (図 1 7)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import My_Data←
←
a = My_Data("名前を記述","性別を記述","血液型を記述","年齢","生年月日")←
a.My_Data.Display()←
```

練習問題 5 解答例その 2 (図 1 8)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
^   def __init__(self,My_Name,My_Nature,My_Blood_Type,My_Age,My_Brith):←
^   ~   self.MY_NAME = My_Name←
^   ~   self.MY_NATURE = My_Nature←
^   ~   self.MY_BLOOD_TYPE = My_Blood_Type←
^   ~   self.MY_AGE = My_Age←
^   ~   self.MY_BRITH = My_Brith←
^   ~   ←
^   def My_Data.Display(self):←
^   ~   print "名前: %s 性別: %s" % (self.MY_NAME,self.MY_NATURE)←
^   ~   print "血液型: %s 年齢: %s" % (self.MY_BLOOD_TYPE,self.MY_AGE)←
^   ~   print "生年月日: %s" % (self.MY_BRITH)←
```

練習問題 6 例 6 - 1 (図 1 9)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import Otani_Student←
←
a = Otani_Student("学部名を記述","学科名を記述","名前を記述","性別を記述")←
a.My_Data.Display()←
a.Otani_Data.Display()←
```

練習問題 6 例 6 - 2 (図 2 0)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
^   def __init__(self,My_Name,My_Nature):←
^   ~   self.MY_NAME = My_Name←
^   ~   self.MY_NATURE = My_Nature←
^   def My_Data.Display(self):←
^   ~   print "名前: %s 性別: %s" % (self.MY_NAME,self.MY_NATURE)←
←
class Otani_Student(My_Data):←
^   def __init__(self,Otani_gakubu,Otani_gakka,My_Name,My_Nature):←
^   ~   self.OTANI_GAKUBU = Otani_gakubu←
^   ~   self.OTANI_GAKKA = Otani_gakka←
^   ~   My_Data.__init__(self,My_Name,My_Nature)←
^   def Otani_Data.Display(self):←
^   ~   print "学部: %s 学科: %s" % (self.OTANI_GAKUBU,self.OTANI_GAKKA)←
```

練習問題 6 解答例その 1 (図 2 1)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import Otani_Student←
←
a = Otani_Student("学部名を記述", "学科名を記述", "学年を記述", ←
                  "学籍番号を記述", "所属ゼミを記述", "名前を記述を記述", ←
                  "性別を記述", "血液型を記述", "年齢を記述", "生年月日を記述")
a.My_Data_Display()←
a.Otani_Data_Display()←
```

練習問題 6 解答例その 2 (図 2 2)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
    def __init__(self, My_Name, My_Nature, My_Blood_Type, My_Age, My_Brith):←
        self.MY_NAME = My_Name←
        self.MY_NATURE = My_Nature←
        self.MY_BLOOD_TYPE = My_Blood_Type←
        self.MY_AGE = My_Age←
        self.MY_BRITH = My_Brith←
    def My_Data_Display(self):←
        print "名前: %s 性別: %s" % (self.MY_NAME, self.MY_NATURE)←
        print "血液型: %s 年齢: %s" % (self.MY_BLOOD_TYPE, self.MY_AGE)←
        print "生年月日: %s" % (self.MY_BRITH)←
class Otani_Student(My_Data):←
    def __init__(self, Otani_gakubu, Otani_gakka, Otani_gakunen, Otani_Number, Otani_Seminer, ←
                 My_Name, My_Nature, My_Blood_Type, My_Age, My_Brith):←
        self.OTANI_GAKUBU = Otani_gakubu←
        self.OTANI_GAKKA = Otani_gakka←
        self.OTANI_GAKUNEN = Otani_gakunen←
        self.OTANI_NUMBER = Otani_Number←
        self.OTANI_SEMINER = Otani_Seminer←
        My_Data.__init__(self, My_Name, My_Nature, My_Blood_Type, My_Age, My_Brith)←
    def Otani_Data_Display(self):←
        print "学部名: %s 学科名: %s" % (self.OTANI_GAKUBU, self.OTANI_GAKKA)←
        print "学籍番号: %s 学年: %s" % (self.OTANI_NUMBER, self.OTANI_GAKUNEN)←
        print "所属ゼミ: %s" % (self.OTANI_SEMINER)←
```

練習問題 7 例 7 - 1 (図 2 3)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
from class_My_Data import Otani_Student , My_Data←
a = My_Data("学部名を記述", "学科名を記述")←
b = Otani_Student("学部名を記述", "学科名を記述", "名前を記述", "性別を記述")
a.Display()←
b.Display()←
```

練習問題 7 例 7 - 2 (図 2 4)

```
#!/usr/bin/env python←
# -*- coding:Shift_JIS -*-←
←
class My_Data():←
    def __init__(self, My_Name, My_Nature):←
        self.all = "名前: %s #n 性別: %s#n" % (My_Name, My_Nature)←
    def Display(self):←
        print self.all←
class Otani_Student(My_Data):←
    def __init__(self, Otani_gakubu, Otani_gakka, My_Name, My_Nature):←
        My_Data.__init__(self, My_Name, My_Nature)←
        self.all = self.all + "学部: %s #n 学科: %s #n" % (Otani_gakubu, Otani_gakka)←
```


練習問題 7 解答例 その1 (図 2 5)

```
#!/usr/bin/env python←  
# -*- coding:Shift_JIS -*-←  
←  
from class_My_Data import Otani_Student , My_Data←  
a = My_Data("学部名を記述","学科名を記述")←  
←  
a.My_Blood("AB型")←  
a.My_Age("22")←  
a.My_Brith("19850101")←  
←  
b = Otani_Student("学部名を記述","学科名を記述")←  
←  
b.Otani_faculty("文学部"):←  
b.Otani_gakubu("人文情報")←  
b.Otani_gakunen("四回生"):←  
b.Otani_number("0448000"):←  
b.Otani_Seminer("福田ゼミ"):←  
←  
←  
a.Display()←  
b.Display()←
```

練習問題 7 解答例 その2 (図 2 6)

```
#!/usr/bin/env python←  
# -*- coding:Shift_JIS -*-←  
←  
class My_Data():←  
    def __init__(self,My_Name,My_Nature):←  
        self.all = "名前: %s 性別: %s %n" % (My_Name,My_Nature)←  
    ←  
    def My_Blood(self,Blood):←  
        self.all += "血液型: %s %n" % (Blood)←  
    ←  
    def My_Age(self,Age):←  
        self.all += "年齢: %s %n" % (Age)←  
    ←  
    def My_Brith(self,Brith):←  
        self.all += "生年月日: %s %n" % (Brith)←  
    ←  
    def Display(self):←  
        print self.all←  
←  
class Otani_Student(My_Data):←  
    def __init__(self,Otani_gakubu,Otani_gakka,My_Name,My_Nature):←  
        ←  
        My_Data.__init__(self,My_Name,My_Nature)←  
        self.all = self.all + "学部: %s %n学科: %s %n" % (Otani_gakubu,Otani_gakka)←  
    def Otani_gakunen(self,gakunen):←  
        self.all += "学年: %s %n" % (gakunen)←  
    def Otani_number(self,number):←  
        self.all += "学籍番号: %s %n" % (number)←  
    def Otani_Seminer(self,seminer):←  
        self.all += "所属ゼミ: %s %n" % (seminer)←
```

Chapter7.html

全体図

オブジェクト指向プログラミング学習サイト

トップページ | オブジェクト指向 | クラスとオブジェクト | ゴッド | 名前空間 | 継承 | 練習問題 | 実践問題

実践問題1 HTMLクラスの作成

実践問題2 作成したHTMLファイルをknoppixで表示

実践問題3 CGIクラスの作成

実践問題4 CGIクラスを使っでの四則計算サイトの作成

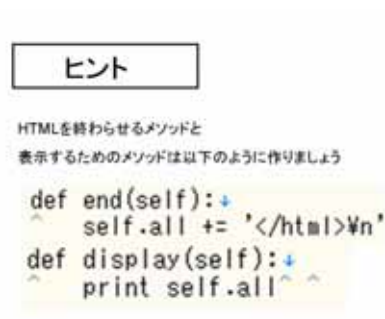
実践問題5 掲示板の作成

実践問題 1

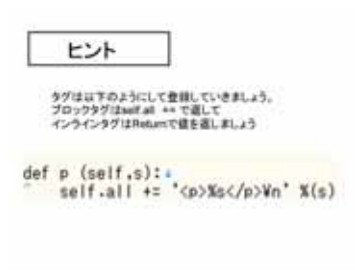
ヒントの画像 1 (ヒント図 1)



ヒントの画像 2 (ヒント図 2)



ヒントの画像 3 (ヒント図 3)



実践問題 1 解答例その 1 (解答例図 1)

```
#!/usr/bin/env python
# -*- coding:shift-jis -*-
+
from class_html import HTML
h = HTML("HTMLクラス")
h.p("このアドレスは大谷大学のアドレスです" + h.a("大谷大学のHP", "http://www.otani.ac.jp"))
h.p("大谷大学生")
h.table_begin("1")
h.tr(h.td("学部:") + h.td("学科:"))
h.tr(h.td("学年:") + h.td("学籍番号:"))
h.tr(h.td("所属ゼミ:"))
h.table_end()
h.end()
h.display()
```

実践問題 1 解答例その 1 (解答例図 2)

```
<meta http-equiv="Content-Type" content="text/html; charset=euc-jp">
<title>X</title>
</head>
*** X (title):
+
def p (self,s):
    self.all += '<p>X</p>\n' X(s)
def h1 (self,s):
    self.all += '<h1>X</h1>\n' X(s)
def a (self,url,a):
    return '<a href="X">X</a>' X (url,a)
def end(self):
    self.all += '</body>\n</html>\n'
+
def ol_begin(self):
    self.all += '<ol>\n'
def ol_end(self):
    self.all += '</ol>\n'
def ul_begin(self):
    self.all += '<ul>\n'
def ul_end(self):
    self.all += '</ul>\n'
def li (self,s):
    return '\n<li>X</li>' X (s)
+
def table_begin(self,border=""):
    self.all += '<table border="X">\n' X (border)
def table_end(self):
    self.all += '</table>\n'
+
def tr(self,s):
    self.all += '<tr>X</tr>\n' X (s)
def td(self,s):
    return '<td>X</td>' X (s)
+
def dl(self,s):
    self.all += '<dl>\nX\n</dl>\n' X (s)
def dd(self,s):
    return '<dd>X</dd>\n' X (s)
def dt (self,s):
    return '<dt>X</dt>' X (s)
def display(self):
    end()
    print self.all
```

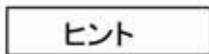
実践問題 2 解答例 その2 (解答例図2)

```
#!/usr/bin/env python+
# -*- coding: euc-jp -*-+
+
+
+
class HTML:+
+ def __init__(self, title='New title'):+
+     self.all = '''<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">+
+ <html>+
+ <head>+
+ <meta http-equiv="Content-Type" content="text/html; charset=euc-jp">+
+ <title>%s</title>+
+ </head>+
+ ''' % (title)+
+
+ def p(self, s):+
+     self.all += '<p>%s</p>\n' % (s)+
+ def h1(self, s):+
+     self.all += '<h1>%s</h1>\n' % (s)+
+ def a(self, url, a):+
+     return '<a href="%s">%s</a>' % (url, a)+
+ def end(self):+
+     self.all += '</body>\n</html>\n'+
+
+ def ol_begin(self):+
+     self.all += '<ol>\n'+
+ def ol_end(self):+
+     self.all += '</ol>\n'+
+ def ul_begin(self):+
+     self.all += '<ul>\n'+
+ def ul_end(self):+
+     self.all += '</ul>\n'+
+ def li(self, s):+
+     return '\n<li>%s</li>' % (s)+
+
+ def table_begin(self, border=""):+
+     self.all += '<table border="%s">\n' % (border)+
+ def table_end(self):+
+     self.all += '</table>\n'+
+
+ def tr(self, s):+
+     self.all += '<tr>%s</tr>\n' % (s)+
+ def td(self, s):+
+     return '<td>%s</td>' % (s)+
+
+ def dl(self, s):+
+     self.all += '<dl>\n%s\n</dl>\n' % (s)+
+ def dd(self, s):+
+     return '<dd>%s</dd>\n' % (s)+
+ def dt(self, s):+
+     return '<dt>%s</dt>' % (s)+
+ def display(self):+
+     end()+
+     print self.all
```

実践問題 2 解答例 その2 (解答例図4)

```
#!/usr/bin/env python+
# -*- coding: shift-jis -*-+
+
+
+
from class_html import HTML+
h = HTML("HTMLクラス")+
h.p("このアドレスは大谷大学のアドレスです" + h.a("大谷大学のHP", "http://www.otani.ac.jp"))+
h.p("大谷大学生")+
h.table_begin("1")+
h.tr(h.td("学部:") + h.td("学科:"))+
h.tr(h.td("学年:") + h.td("学籍番号:"))+
h.tr(h.td("所属ゼミ:"))+
h.table_end()+
h.end()+
h.display()
```

実践問題 3 ヒント画像 (ヒント図4)



CGIのクラスではまず以下のように書きましょう

```
__init__(self, title='New title'):+
import cgi, cgitb+
self.form = cgi.FieldStorage()+
HTML.__head__(self, title='New title')+
self.all = "Content-type: text/html\n\n" + self.all
```

実践問題3 解答例その1 (解答例図5)

```
#!/usr/bin/env python+
# -*- coding:euc-jp -*-+
+
from class_html import CGI+
+
h = CGI_test("検査画面")+
h.p("大学のホームページ" + h.a("abc", "http://www.otani.ac.jp"))+
h.p("大谷大学生")+
h.ol_begin("")+
h.p(h.li("学部:") + h.li("学科:") + h.li("学籍番号:"))+
h.ol_end()+
h.ul_begin()+
h.p(h.li("学年:") + h.li("ゼミ"))+
h.ul_end()+
h.end()+
h.display()+
```

実践問題3 解答例 その2 (解答例図6)

```
class CGI(HTML):+
    def __init__(self, title='New title'):+
        ~ import cgi, cgiib+
        ~ self.form = cgi.FieldStorage()+
        ~ HTML.__head__(self, title='New title')+
        ~ self.all = 'Content-type: text/html\n\n' + self.all+
+
    def get_str(self, Name):+
        return self.form.getfirst(Name, '')+
+
    def get_int(self, Name):+
        return int(self.form.getfirst(Name, 0))+
+
    def get_float(self, Name):+
        ~ return float(self.form.getfirst(Name, 0))+
        ~ +
+
    def form_begin(self, action, met="POST"):+
        self.all += '<form action="%s" method="%s">\n' % (action, met)+
    def form_end(self):+
        self.all += '</form>\n'+
    def input_type(self, write, title):+
        self.all += '<input type="%s" name="%s">\n' % (write, title)+
+
    def textarea(self, title):+
        return '<textarea name="%s"></textarea>' % (title)+
    def textarea_size(self, title, high, width):+
        return '<textarea name="%s" cols="%s" rows="%s" wrap="%s"></textarea>' % (title, high, width)
+
    def input_line(self, Name, Value=""):+
        return '<input type="text" name="%s" value="%s">\n' % (Name, Value)+
    def input_text(self, Name, Value=""):+
        return '<textarea name="%s">%s</textarea>' % (Name, Value)+
    def b_send(self, Name):+
        return '<input type="submit" value="%s">' % (Name)+
    def b_clear(self, Name):+
        return '<input type="reset" value="%s">' % (Name)+
```

実践問題4 ヒントの画像1 (ヒント図5) 実践問題4 ヒントの画像2 (ヒント図6)

```
class CGI(HTML):+
    def __init__(self, title='New title'):+
        ~ import cgi, cgiib+
        ~ self.form = cgi.FieldStorage()+
        ~ HTML.__head__(self, title='New title')+
        ~ self.all = 'Content-type: text/html\n\n' + self.all+
+
    def get_str(self, Name):+
        return self.form.getfirst(Name, '')+
+
    def get_int(self, Name):+
        return int(self.form.getfirst(Name, 0))+
+
    def get_float(self, Name):+
        ~ return float(self.form.getfirst(Name, 0))+
        ~ +
+
    def form_begin(self, action, met="POST"):+
        self.all += '<form action="%s" method="%s">\n' % (action, met)+
    def form_end(self):+
        self.all += '</form>\n'+
    def input_type(self, write, title):+
        self.all += '<input type="%s" name="%s">\n' % (write, title)+
+
    def textarea(self, title):+
        return '<textarea name="%s"></textarea>' % (title)+
    def textarea_size(self, title, high, width):+
        return '<textarea name="%s" cols="%s" rows="%s" wrap="%s"></textarea>' % (title, high, width)
+
    def input_line(self, Name, Value=""):+
        return '<input type="text" name="%s" value="%s">\n' % (Name, Value)+
    def input_text(self, Name, Value=""):+
        return '<textarea name="%s">%s</textarea>' % (Name, Value)+
    def b_send(self, Name):+
        return '<input type="submit" value="%s">' % (Name)+
    def b_clear(self, Name):+
        return '<input type="reset" value="%s">' % (Name)+
```

同じ色の箱には同じ変数が入ります
変数名は考えて入れましょう

```
h = CGI("面積計算")+
+
h.get_float( ) = h.get_float( )+
h.get_float( ) = h.get_float( )+
+
h.get_float( ) = h.get_float( )+
h.get_float( ) = h.get_float( )+
+
h.get_float( ) = h.get_float( )+
h.get_float( ) = h.get_float( )+
+
h.get_float( ) = h.get_float( )+
h.get_float( ) = h.get_float( )
```

実践問題4 ヒントの画像3(ヒント図7)

```
if 高さ != 0:+
    menseki = 底辺 * 高さ / 2.0+
    h.p("三角形の面積は")+
    h.p(str(menseki))+
+
if 底辺 != 0:+
    tyo_menseki = 高さ * 底辺 / 2.0+
    h.p("長方形の面積は")+
    h.p(str(tyo_menseki))+
+
if 半径 != 0:+
    en = 半径 * 半径 * 3.14+
    h.p("円の面積は")+
    h.p(str(en))+
+
if 上底 != 0:+
    dai_men = (上底 + 下底) * 高さ / 2.0+
    h.p("台形の面積は")+
    h.p(str(dai_men))+
```

実践問題4 ヒントの画像4 (ヒント図8)

```
h.p("いろいろな計算をしてみよう")+
h.form_begin("wensaki_html.cgi")+
h.h1("三角形")+
h.p("高さ:" + h.input_line( ))+
h.p("底辺:" + h.input_line( ))+
+
h.h1("長方形")+
h.p("底:" + h.input_line( ))+
h.p("横:" + h.input_line( ))+
+
h.h1("円")+
h.p("半径:" + h.input_line( ))+
+
h.h1("台形")+
h.p("上底:" + h.input_line( ))+
h.p("下底:" + h.input_line( ))+
h.p("高さ:" + h.input_line( ))+
+
h.p(h.b_send("送信") + h.b_clear("削除"))+
h.form_end()+
h.end()+
```

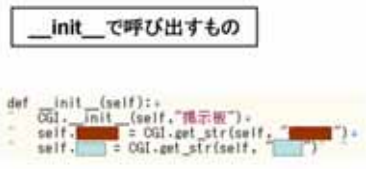
実践問題 4 解答例 その1(解答例図7)

```
#!/usr/bin/env python+
# -*- coding:euc-jp -*-+
+
+
class HTML:+
    def __init__(self, title='New title!):+
        self.all = '''<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>+
<head>+
<meta http-equiv="Content-Type" content="text/html; charset=euc-jp">+
<title>%s</title>+
</head>+
''' % (title)+
+
    def p(self,s):+
        self.all += '<p>%s</p>%n' % (s)+
+
    def h1(self,s):+
        self.all += '<h1>%s</h1>%n' % (s)+
+
    def a(self,url,a):+
        return '<a href="%s">%s</a>' % (url,a)+
+
    def end(self):+
        self.all += '</html>%n'+
+
    def ol_begin(self):+
        self.all += '<ol>%n'+
+
    def ol_end(self):+
        self.all += '</ol>%n'+
+
    def ul_begin(self):+
        self.all += '<ul>%n'+
+
    def ul_end(self):+
        self.all += '</ul>%n'+
+
    def li(self,s):+
        return '%n<li>%s</li>' % (s)+
+
    def table_begin(self,border=""):+
        self.all += '<table border="%s">%n' % (border) +
+
    def table_end(self):+
        self.all += '</table>%n'+
+
    def tr(self,s):+
        self.all += '<tr>%s</tr>%n' % (s)+
+
    def td(self,s):+
        return '<td>%s</td>' % (s)+
+
    def dl(self,s):+
        self.all += '<dl>%n%s%nc</dl>%n' % (s)+
+
    def dd(self,s):+
        return '<dd>%s</dd>%n' % (s)+
+
    def dt(self,s):+
        return '<dt>%s</dt>%n' % (s)+
+
+
    def display(self):+
        print self.all^ +
```

実践問題 4 解答例 その2(解答例図8)

```
#!/usr/bin/env python+
# -*- coding:euc-jp -*-+
+
from html import CGI+
+
h = CGI("面積計算")+
+
san_tate = h.get_float("san_tate")+
san_yoko = h.get_float("san_yoko")+
+
tate = h.get_float("tate")+
yoko = h.get_float("yoko")+
+
hankei = h.get_float("hankei")+
+
joutei = h.get_float("joutei")+
katei = h.get_float("katei")+
dai_takasa = h.get_float("dai_takasa")+
+
+
+
h.p("いろいろな計算をしてみよう")+
h.form_begin("menseki_html.cgi")+
h.h1("三角形")+
h.p("高さ: " + h.input_line("san_tate"))+
h.p("底辺: " + h.input_line("san_yoko"))+
+
h.h1("長方形")+
h.p("縦: " + h.input_line("tate"))+
h.p("横: " + h.input_line("yoko"))+
+
h.h1("円")+
h.p("半径: " + h.input_line("hankei"))+
+
h.h1("台形")+
h.p("上底: " + h.input_line("joutei"))+
h.p("下底: " + h.input_line("katei"))+
h.p("高さ: " + h.input_line("dai_takasa"))+
+
h.p(h.b_send("送信") + h.b_clear("削除"))+
h.form_end()+
h.end()+
+
if san_tate != 0:+
    menseki = san_tate * san_yoko / 2.0+
    h.p("三角形の面積は")+
    h.p(str(menseki))+
+
if tate != 0:+
    tyo_menseki = tate * yoko / 2.0+
    h.p("長方形の面積は")+
    h.p(str(tyo_menseki))+
+
if hankei != 0:+
    en = hankei * hankei / 3.14+
    h.p("円の面積は")+
    h.p(str(en))+
+
if joutei != 0:+
    dai_men = (joutei + katei) * dai_takasa / 2.0+
    h.p("台形の面積は")+
    h.p(str(dai_men))+
+
h.display()+
```

実践問題 5 ヒントの画像1(ヒント図9) 実践問題 5 ヒントの画像2(ヒント図10)



実践問題 5 ヒントの画像3(ヒント図11) 実践問題 5 ヒントの画像4(ヒント図12)



実践問題 5 ヒントの画像 5 (ヒント図 1 3) 実践問題 5 ヒントの画像 6 (ヒント図 1 4)

投稿部分をつくるための部品

```
def display_form(self):
    self.form_begin("BBS.cgi")
    self.a("投稿フォーム")
    self.a("名前:" + self.input_line("名前"))
    self.a("Eメール:" + self.input_text("Eメール"))
    self.a(self.b_send("送信") + self.b_clear("クリア"))
    self.form_end()
```

時間を表示させるための部品

```
def getDateTime(self):
    date_time = strftime("%Y/%m/%d, %H:%M:%S", localtime())
    return date_time
```

実践問題 5 ヒントの画像 7 (ヒント図 1 5) 実践問題 5 ヒントの画像 8 (ヒント図 1 6)

掲示板を表示させるための部品

```
def display(self):
    if self.kiji != "":
        self.write_kiji()
        self.display_data()
        self.display_form()
    print self.all
```

これを最後につけて完成

注意: 掲示板のクラス内にはいれないように注意してください

```
if __name__ == "__main__":
    bbs = BBS()
    bbs.display()
```

実践問題 5 解答例 その 1 (解答例図 9) 実践問題 5 解答例 その 2 (解答例図 1 0)

```
#!/usr/bin/perl
# coding: utf-8

class HTML
{
    def __init__(self, title="New title"):
        self.all = ""
        #HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        #>>
        meta HTTP-equiv="Content-Type" content="text/html; charset=utf-8"
        <title>BBS</title>
        <head>
        <title>
        def a(self, a):
            self.all += "<a href='%s'>%s</a>" % (a, a)
        def b(self, b):
            self.all += "<b>%s</b>" % (b, b)
        def form_begin(self, action, met="POST"):
            self.all += "<form action='%s' method='%s'>" % (action, met)
        def form_end(self):
            self.all += "</form>"
        def input_line(self, Name, Value=""):
            self.all += "<input type='text' name='%s' value='%s'>" % (Name, Value)
        def input_text(self, Name, Value=""):
            self.all += "<input type='text' name='%s' value='%s'>" % (Name, Value)
        def b_send(self, Name):
            self.all += "<input type='submit' value='%s'>" % (Name)
        def b_clear(self, Name):
            self.all += "<input type='reset' value='%s'>" % (Name)
    def display(self):
        print self.all
```

```
class CGI(HTML):
    def __init__(self, title="New title"):
        import cgi, cgitb
        self.form = cgi.FieldStorage()
        HTML.__init__(self, title)
        self.all = "Content-type: text/html\n\n" + self.all

    def get_str(self, Name):
        return self.form.getfirst(Name, '')

    def get_int(self, Name):
        return int(self.form.getfirst(Name, 0))

    def get_float(self, Name):
        return float(self.form.getfirst(Name, 0))

    def form_begin(self, action, met="POST"):
        self.all += "<form action='%s' method='%s'>" % (action, met)
    def form_end(self):
        self.all += "</form>"

    def input_line(self, Name, Value=""):
        return "<input type='text' name='%s' value='%s'>" % (Name, Value)
    def input_text(self, Name, Value=""):
        return "<input type='text' name='%s' value='%s'>" % (Name, Value)
    def b_send(self, Name):
        return "<input type='submit' value='%s'>" % (Name)
    def b_clear(self, Name):
        return "<input type='reset' value='%s'>" % (Name)
```

実践問題 解答例 その3 (解答例図11)

```
#!/usr/bin/env python+
# -*- coding:euc_jp -*-+
+
from html import HTML, CGI+
import os.path+
from time import localtime, strftime+
+
class BBS(CGI):+
    ~ def __init__(self):+
    ~     CGI.__init__(self, "掲示板")+
    ~     self.shimei = CGI.get_str(self, "shimei")+
    ~     self.kiji = CGI.get_str(self, "kiji")+
    ~ +
    ~ def getDateime(self):+
    ~     date_time = strftime("%Y/%m/%d, %H/%M:%S", localtime()+
    ~     return date_time+
    ~ +
    ~ def write_kiji(self):+
    ~     now = self.getDateime()+
    ~     self.kiji = self.kiji.replace("\n", "<br>")+
    ~     data = self.dt(self.shimei + "(" + now + ")")+
    ~     data += self.dd(self.kiji)+
    ~     bbs_data = open("bbs.data", "a")+
    ~     bbs_data.write(data)+
    ~     bbs_data.close()+
    ~ +
    ~ def display_data(self):+
    ~     if os.path.exists("bbs.data"):+
    ~         kiji = "</dl>\n"+
    ~         kiji_bango = 0+
    ~         bbs_data = open("bbs.data")+
    ~         line = bbs_data.readline()+
    ~         while line != "":+
    ~             if "<dt>" in line:+
    ~                 kiji_bango += 1+
    ~                 line = "<dt>" + str(kiji_bango) + ". " + line[4:]
    ~                 kiji = line + kiji+
    ~                 line = bbs_data.readline()+
    ~             bbs_data.close()+
    ~             self.all += "\n<dl>\n" + kiji+
    ~         else:+
    ~             self.all += "File Not Found"+
    ~ +
    ~ def display_form(self):+
    ~     self.form_begin("BBS.cgi")+
    ~     self.p("投稿フォーム")+
    ~     self.p("名前: " + self.input_line("shimei")+
    ~     self.p("記事: " + self.input_text("kiji")+
    ~     self.p(self.b_send("送信") + self.b_clear("クリア")+
    ~     self.form_end()+
    ~ +
    ~ def display(self):+
    ~     if self.kiji != "":+
    ~         self.write_kiji()+
    ~         self.display_data()+
    ~         self.display_form()+
    ~         print self.all+
    ~     +
    ~ +
if __name__ == "__main__":+
    bbs = BBS()+
    bbs.display()+
```