クラスのつくりかた（１）（作成の基礎）

クラス＝オブジェクトの設計図（雛形）

前回の辞書のように，いろんな仕事をする上で欠かせない様々な「モノ＝オブジェクト」をいろいろ作って，組み合わせてプログラムを作るのが，「オブジェクト指向」 なプログラミングのスタイルで，Pythonはこの考え方に則ったプログラミングに適した言語です．

このオブジェクトを新たに自分好みに作るための枠組みが「クラス」です．

クラスを定義する

クラスは，自分の用途にあったオリジナルのオブジェクトの設計図です．

以前習ったdefを使った関数の定義によくにていますが，一般的には以下の書式で，表したいオブジェクトの「状態」を表すインスタンス変数（名前）と，それらを 使ってオブジェクトの振る舞いを記述するメソッド（関数）を（両者を合わせて属性という）定義します．

```
class クラス名(object):

    #このクラスのオブジェクトを一つ新たに作った時，その内容を初期化するために必ず呼ばれるメソッド．
    def __init__(self):
        #各オブジェクトが持っている変数はself.インスタンス変数名と書く
        self.インスタンス変数名1= ...
        self.インスタンス変数名2= ...

    #インスタンス変数を使ってオブジェクトに関する処理をするメソッド
    def メソッド名1(self):
        self.インスタンス変数名1= ...

    def メソッド名2(self, 引数1, 引数2, ...):
        ...
```

うーん，具体的でないのでなんのこっちゃですね．例えば次のような「湯沸しポット」を表すクラスを考えてみましょう．

- クラス名
  - Kettle
- インスタンス変数
  - water：いま「その」ポットに入っている水（お湯）の量をmlで表現．最初は100．
  - temp：いま「その」ポットに入っている水（お湯）の温度を℃で表現．最初は常温として20℃．
- メソッド
  - __init__(self)：新しくポット（オブジェクト）を作った時，「その」ポットの水の量を100ml，温度を20℃とする．
  - putWater(self)：「水を入れます．」と表示したのち，「その」ポットに20℃で100mlの水を入れる．その際，今入っている水の温度と量，入れる水の温度と量から合わせた温度と量を計算する．
  - printStatus(self)：今の「その」ポットの状態を表示する．
    具体的には，「ポットには？℃のお湯が？mlはいっています。」と表示する．

このクラスを定義すると以下のようになります．

```
class Kettle(object):

    def __init__(self):
        self.water= 100
        self.temp= 20


    def putWater(self):
        self.temp= (self.water*self.temp+100*20)/(self.water+100)
        self.water= self.water+100

    def printStatus(self):
        print("ポットには{0:.1f}℃のお湯が{1:.1f}mlはいっています。"\
              .format(self.temp, self.water))

```

こちらの方が具体的な意味づけがわかりそうですね．以下の点に注意して，両者を眺めてみて，どんな風に書かれているか確認しましょう．

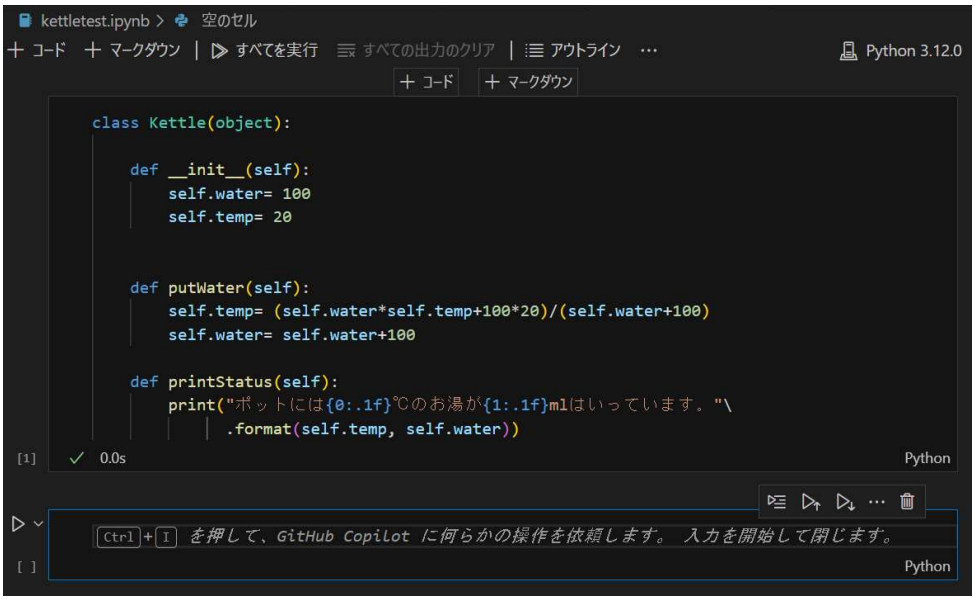
- クラスの定義は「class」で始まる．関数やfor文と同じように，定義の範囲をインデント（字下げ）で表現する．
- メソッドはdefキーワードで普通の関数と同じようにして定義される．
- どのメソッドにも必ず最初の仮引数として「self」をつけるのが慣例．
- selfという変数は，「その」オブジェクト自身を示しており，各メソッドの中で，そのオブジェクト自身が持つ変数やメソッドを「self.変数名」 「self.メソッド名」のように（ドットノーテーションで）明示する．ただし，実際にメソッドを利用する際にはselfは省略される（あとでわかる）．
- __init__(self)は特殊なメソッドで，オブジェクトを作ると必ず呼び出される．多くの場合，この__init__(self)メソッドの 中で，新しい変数に値を割り当てることで，このクラスのオブジェクトが持つ変数を定義する．この場合は，waterとtempがそれにあたり，その中身を それぞれ100，20と割り当てている．
- 複数のメソッドを定義する場合は，インデントを維持したまま列挙すれば良い．

早速これを使ってみましょう．

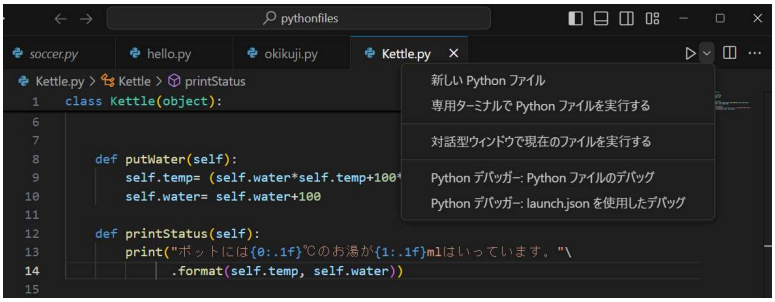
２つの方法を示します．

１）ノートブック上で動かす

先週紹介したノートブック（???.ipynbのファイル）の1つのセルに，上の定義全体を記入し，再生ボタンを押して（もしくは左シフトキー＋エンター）実行します．結果としては何も示されず，おそらく下に新たなセルが表れた状態になると思います．これでOKです．ノートブック形式の場合は，ひとつのセルを実行しても，まだプログラム自体は実行中で，続けて任意のセルを実行をすることを繰り返して（結果を見ながらインタラクティブに）実行できます．



２）上のコードが書かれたKettle.pyをこれまで通りに作成します．つぎに，ウィンドウ右上の再生ボタンで実行しますが，この時そのすぐ隣の「v」のところをクリックして，メニューを出し，「対話型ウィンドウで現在のファイルを実行する」を選んで実行します．こちらではコードのファイルの実行後，Pythonの実行を維持して右画面にノートブックと同様な対話形式のセルを利用できます．



（これ以降の資料では，昔ながらのPythonシェルでインタラクティブに入力している形式（>>>）になっていますが，適宜上のセルを使って入力・実行する方法に置き換えて試してください.）

作成したクラスからのオブジェクトの生成

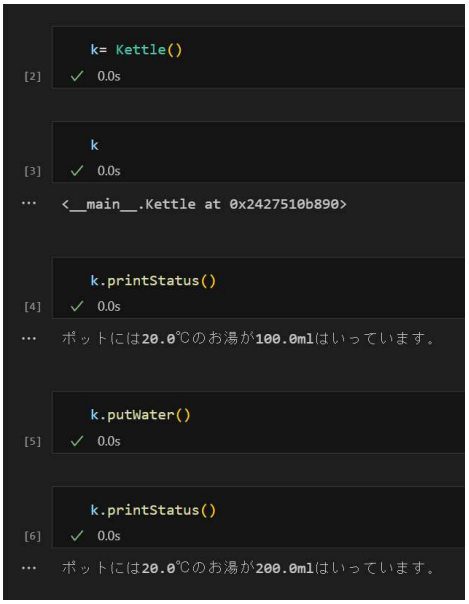
具体的には，以下のようにしてクラスのオブジェクトを作り，変数を割り当て，それを使ってメソッドを実行します.

```
#あるクラスのオブジェクトを生成し，変数名に割り当てる
変数名= クラス名(__init__メソッドで定義された引数（ただしselfなし）)

#割り当てた変数でそのオブジェクトのメソッドを実行
変数名.メソッド名(定義された引数（ただしselfなし）)

```

具体的にはこんな感じ．順番に実行して，様子を見てみましょう.




```
>>> k=Kettle()
>>> k
<__main__.Kettle object at 0x02B65D30>
>>> k.printStatus()
ポットには20度のお湯が100mlはいっています。
>>> k.putWater()
>>> k.printStatus()
ポットには20度のお湯が200mlはいっています。
>>>

```

- まず最初にクラス名の名前がついた関数でKettleクラスのオブジェクトを作り，それを変数kに割り当てます.

- kとだけ入力すると，割り当てたオブジェクトがKettleだというのがわかります．
- 次に，kを使って，printStatus()のメソッドを実行．このときselfは無視するので引数ゼロで実行します．その結果，__init__ メソッドで定義したように，このオブジェクトにはwaterには100，tempには20が入っていて，それを元にprint関数が呼ばれた結果であるこ とがわかります．
- もう一つのメソッドを実行して，水を100ml加えています．でもこれ自体は出力はありません．
- 最後にもう一度printStatus()を実行しています．最初と比べてちゃんと水が100ml増えていることを確認しましょう．

もっとポットらしくしてみよう

水を注ぐだけがポットの機能ではありません．色々付け足したり，改変して機能を充実させながら，どこに何を書くとどうなるかみていきましょう．
それぞれ，改変や追加ができた段階で，期待通りの動作ができているか，オブジェクトを作り，メソッドを実行して確かめてみましょう．ノートブックの場合は，Kettleクラスの定義が書かれたセルを実行しなおしましょう．ソースコードの場合は，クラスのファイルを書き換えたら保存して，対話ウィンドウを終了し，再度実行することでクラスの定義も更新するのを忘れずに．

1）お湯を温めるメソッドheatを作る（新しいメソッドの追加）
このメソッドを一度実行すると，1分間加熱して1000mlのお湯を25度上げるだけのエネルギーをお湯に加えられること にしましょう．とすると，今，お湯はwater（ml）入っているのだから，これを温めた結果は何度になるかわかりますね．addWaterメソッドの中のself.waterと同じようにして，新たに用意したheatメソッドの中で，この値をself.tempに割り当てれば良さそうです．

つまり，以下のメソッドを追加すれば良さそうです．

```
def heat(self):
    self.temp= self.temp+1000.0/self.water*25.0
```

heatメソッドを何度か実行してみたら，お湯の温度が適切に上がりますか？

2）沸騰したら「ピー」（条件に応じた振る舞いの制御）
ところが，今のポットでは，お湯の温度は上がりっぱなしで100度を超えてしまいますね．
これではおかしいので，お湯の温度は100度以上には上がらないようにしてみましょう．上のheatメソッドに，「もし今の温度が100を超えていたら100にす る」という内容の文を追加してみましょう．また，100にしたと同時に，「お湯が沸きました！」と画面に表示するようにしてみましょう．

（heatメソッドの最後に以下を追加）

```
if(self.temp>100):
    self.temp=100
    print("お湯が沸きました")
```

3）お湯を全部注ぐ（メソッド内での別のメソッドの利用）
お湯が沸いたら注いでみましょう．pourメソッドを新たに追加して，これを実行すると，いまその中にあるお湯を全部出すことにしましょう．「？℃のお湯を？ml 注ぎました.」と表示したあと，お湯を全部出し，ポットの温度を常温の20℃にしておきましょう．

```
def pour(self):
    print("{0:.1f}℃のお湯を{1:.1f}ml注ぎました. ".format(self.temp, self.water))
    self.water= 0
    self.temp= 20
```

4）もういっぱいです（新たなインスタンス変数の追加とメソッドの引数の追加）
普通ポットは入れられる水の量が決まっていますが，いまのポットにはそれがありません．そこで，新たにこのオブジェクトを生成するときに，このポットの容量を引数 として指定できるように改変してみます．生成する際に呼び出される__init__(self)メソッドを次のように書き換えてみましょう．

```
def __init__(self, maxw=1000):
    self.water= 100
    self.temp= 20
    self.maxwater= maxw
```

最初に入っていた引数selfに加えて，新たに引数maxwを加えています．また，その値をデフォルト値，つまり，もしその引数が与えられなかった時に入れておく 値を1000としています．これを，最後に新しく追加した，最大容量を示すインスタンス変数maxwaterに割り当てています．

さらに，printStatusの中のprint文を書き換えて，「？ml入るポットには・・・」のように，容量も表示するようにしてみましょう．

（例省略）

以上ができたら，ポットを作ると，1000ml入るようになっていることを確認しましょう．
そのあと，ポットのオブジェクトを作成する際に，引数を指定して任意の容量のポットが作れることを確認しましょう．具体的には，例えば500mlのポットを作るに は，以下のようにします．

```
k= Kettle(500)
k.printStatus()
```

容量が決まったら，それ以上入れられないようにしなくてはなりませんね．2）の，温度が100度を超えられないようにするのを真似て，今度は，水が最大容量を超え ないようにする処理を考えてみましょう．このとき，最大容量に達している時には，「ポットはいっぱいです.」と表示するようにしてみます．

ただし，いっぱいになるまで入れられるだけ入れて，残りは捨てることにします．なので，まず，入れられる最大量をmaxamountと名付けた変数に割り当てま す．その後，maxamountが一階に入れる量である100mlより小さければ，maxamount分だけ入れるようにし，そうでなければそのまま100mlを 入れるようにします．下の例では，if文を使って最終的に入れる量を変数netamountに割り当てています．

```
def putWater(self):
    maxamount= self.maxwater-self.water
    if(maxamount<100):
        netamount= maxamount
        print("お湯がいっぱいです。")
    else:
        netamount= 100
    self.temp= (self.water*self.temp+netamount*20)/(self.water+netamount)
    self.water= self.water+netamount
```

5）入れる量を決める（メソッドのリファクタリングと引数のデフォルト指定）

今のputWaterメソッドは，毎回一定の量の水しか入れられないので，面倒ですよね．そこで，4）で行った，メソッドに引数を追加する方法を参考にし て，putWaterにも引数amountとtemperatureを追加しましょう．その名前が示す通り，amountには水の量，temperatureにはその温度 が入るものとします．

（例省略）

6）完成？
ここまでできたら，以下の動作に対応する命令を，Pythonシェル，もしくは，Kettleクラスの定義の後に列挙して実行してみましょう．

- 最大容量500mlのポットのオブジェクトをつくり，変数kに割り当てる．
- kの状態を表示する．
- kに25℃のお湯200mlを入れる．
- kの状態を表示する．
- kに10℃のお湯300mlを入れる．
- kの状態を表示する．
- kを2度あたためる．
- kからお湯を注ぐ．

（例省略）

7）あのポットとこのポット
それなりにポットらしいクラスが出来上がり，ポットを操作することができました．でも，世の中にポットはこれただ一つではありません．給湯室にポットが並べられて いる時，同じポットでも大きさや入っているものは異なることがあるでしょう．
クラスを使ったプログラムの設計の大きなメリットの一つは，あるクラスのオブジェクトを複数用意できることです．オブジェクトは生成されるたびに，それぞれ独立し たものとして扱われます．

例えば，以下のようにKettleクラスのオブジェクトを2度生成し，それらを別の変数に割り当てて，挙動を確認してみましょう．(>>>の後を各セルに入れる内容と思ってください)

```
>>> k1=Kettle(1000)
>>> k2=Kettle(300)
>>> k1.putWater(500, 30)
水を入れます。
>>> k1.printStatus()
容量1000.0mlのポットには28.3℃のお湯が600.0mlはいっています。
>>> k2.putWater(100,15)
水を入れます。
>>> k2.printStatus()
容量300.0mlのポットには17.5℃のお湯が200.0mlはいっています。
>>> k1.pour()
28.3℃のお湯を600.0ml注ぎました.
>>> k1.printStatus()
容量1000.0mlのポットには20.0℃のお湯が0.0mlはいっています。
>>> k2.printStatus()
容量300.0mlのポットには17.5℃のお湯が200.0mlはいっています。
```

冒頭で作られた二つのポットのオブジェクトはそれぞれ独立して存在していて，割り当てた変数k1とk2でメソッドを使い分けて操作することができま す．

どうでしょう．最初は単なる変数と関数のちいさな集まりだったのが，次第にある種の機能を持ったオブジェクトに見えてきませんか？

インスタンス変数へのアクセス

上記では取り上げていませんが，メソッドと同様にして，ドットノーテーションを用いて，インスタンス変数を直接参照することができます．

例えば，以下のように，tempやwaterの値を直接調べたり，値の割り当てを変えたりすることができます．

```
>>> k= Kettle(500)
>>> k.temp
20.0
>>> k.water
100.0
>>> k.water=3000
>>> k.temp=200
>>> k.temp
200
>>> k.water
3000
>>>
```

このような使い方は簡単に参照できるというメリットがまずあり，よく使われます．

その反面，予期せぬ値が割り当てられることなどもあり得ます．例えば，上の例では，tempに200が割り当てられていますが，これはナンセンスです．
クラスのインスタンス変数には意味づけがあるので，それに則さないものが入ってしまうとこまるというわけです．その代わりに，今回は文脈に応じた形で温度をあげる heatがあるわけです．

実際には，そのクラスの役割に応じて臨機応変に，直接触ることをよしとするか，そうしないかを決めます．(慣習として，触れない場合は変数名の冒頭に_をつけることもあります.)

(独自クラスの作成と利用に関する課題)