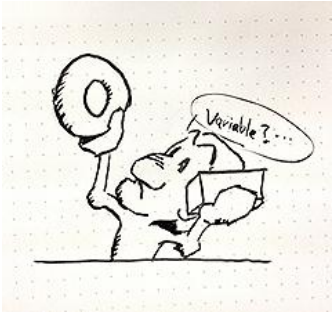
難易度：　★☆☆☆☆



押忍！麓です。・・・さて、前回で「変数」の使い方が何となくわかってきたところかな？

そこで今回はまず最初は、少しだけ Python 独特の変数の使用方法を工夫する事例から始めます。

1. メソッドを変数に入れる

Python は「関数はオブジェクト」であるという特性を持っています。

※詳細は「Python の関数はオブジェクト」と Google で検索するとたくさん出てきますので省きます。

また、正確に言うとすこし意味が違いますが、前述の項でオブジェクトを変数に入れるということをしました。

さて、何が言いたいのかというと。いままで、

Application.CreatePrim

Application.Logmessage

のように頭に”Application”をつけてコマンドを読んでいました。Softimage の持っているコマンドは Application オブジェクトから呼ばれるので必ず頭に付ける必用があるのですが、文字数が多いので面倒ですし、コードが読みにくくなります。

そこでまずは Application を変数に入れて短くしちゃいましょう。

```
si = Application
```

こうやって Application を si という変数に入れておくことで、上の CreatePrim や Logmessage は

```
si.CreatePrim("Torus", "MeshSurface", "", "")
si.Logmessage("Torus")
```

と記述することができます。

コマンドを視認しやすくなって良いですね！

もう少し踏み込んで、自分が PythonScript を書くときに必ずやっていることがありますので、ついでに紹介しておきます。

```
Logm = si.Logmessage
```

です。

Application の Logmessage というメソッドを Logm で呼び出せるようにしています。

※Softimage の SDK ドキュメントでは Application を app という変数名に格納しています。決められた関数名などにぶつからなければ何でも良いと思います。

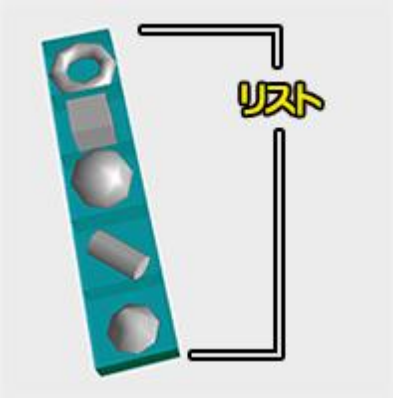
それでは前回に引き続き、多言語では「配列」と言われているものの扱いについて Softimage ではどう使うのかを解説していきます。

2. リスト

Python の配列には**リストとタプル**と、二種類あります。

リストは動的配列、タプルは静的配列と目的別に使い分けますが、多くの場合、どんなデータが来ても処理できるように・・・とかフレキシブルな対応を求められますので、リストを選択する場面が増えます。

そんなところで、ここでリスト操作をメインに紹介していきます。



リストのイメージはこんな感じです。

仮にこの５個のオブジェクトが入っている連なった箱があり、これらをセットにして「変数名」を付けて(定義する)処理をします。

変数名をつけた時にこの箱の数が**固定されているのがタプル、自由に増減できるのがリスト**、と呼ばれています。

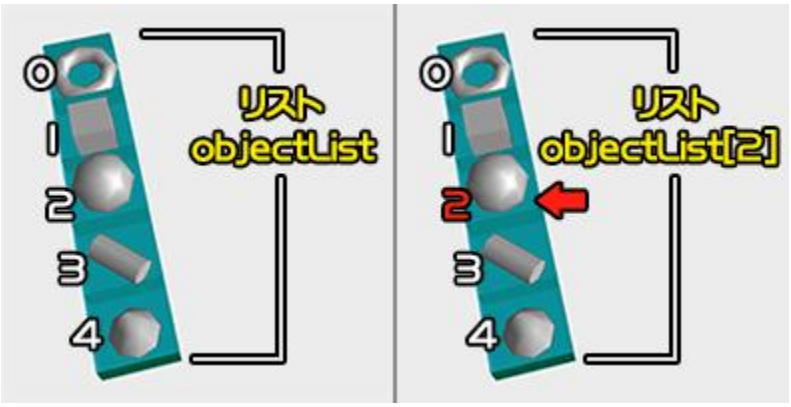
実際に PythonScript で書くと・・・

```
objectList = ['torus','cube','sphere','cylinder','cone']
```

このようになります。

それぞれのオブジェクトの名前(文字列)をカンマで区切って、角括弧=[] で囲みます。

また、リストの中身にアクセスするには[]に何個目かを指定するインデックス(開始は 0)を使用するのが一番オーソドックスな方法です。



```
si = Application
Logm = si.Logmessage
objectList = ['torus','cube','sphere','cylinder','cone']
Logm( objectList[2] )
```

上記コードを ScriptEditor で実行すると、

INFO : sphere

とログが表示されます。

最終行で objectList のインデックス2(3個目)をログに表示させているためです。

さて、この状態だとただ文字列のをリストにしているだけですのでオブジェクトそのものにアクセスすることができません。

“選択オブジェクトに対して”処理をさせる。というスクリプトをわりと頻繁に書くことがありますが、そういう時は Selection という Application のプロパティを使用します。例えば上記を置き換えると、

```
si = Application
Logm = si.Logmessage
objectList = si.Selection
Logm( objectList[2] )
```

Softimage の場合、SDK ドキュメントにはオブジェクトのリストオブジェクトを「**コレクション**」と表記しています。Python は全てオブジェクトとして処理しますので、中身が Softimage オブジェクトに限り同一の意味を持ち、コレクションに使えるメソッドやプロパティが使用出来ます。

※選択オブジェクトの数が 3 個以上無いと Logm の行でエラーが出ます。リストのインデックス指定は実際のサイズより大きい値を指定するとエラーになります。

```
si = Application
Logm = si.Logmessage
objectList = si.Selection
Logm( objectList[2] )


Logm( objectList.GetAsText() )
objectList.Add(si.Dictionary.GetObject('Camera_Root'))
```

ここで少しメソッドとプロパティについて説明します。

ググったりすれば見つかりますが Softimage に特化して書くと・・・

- メソッド(method)

  オブジェクトに対して処理する命令です。最後に括弧()が付くのは殆どの場合に引数を設定する事が多く、その引数を使って処理をするからです。例えば上記の GetAsText()や Add()は **objectList のメソッド**です。Add に関しては引数の'Camera_Root'をオブジェクトとして追加します。

- プロパティ(property)
  オブジェクトが持っている情報です。読み取り専用と書き込みができる場合があります。例えば、コレクションの中身がいくつあるか？ **objectList.Count** は読み取り専用で取得のみですが、**.Name** はオブジェクトに対して名前を取得したり設定したり出来ます(コレクションには使えません)。

スライス

ここで、Python を使う利点としてリストのスライス参照について紹介します。
折角 Python を使うのであれば、便利なこの辺りの機能は使い倒していきたいですね。
まず、上記の文字列のリストのみを作っておきます。

```
objectList = ['torus','cube','sphere','cylinder','cone']
```

わりと便利でよく使うのが、、
リスト[開始位置:終了位置:ステップ](それぞれ省略可:省略した場合 0 となる)
です。
例えば objectList の 2 つめ' cube' から4つめ' cylinder' までが欲しい場合、

```
print objectList[1:4]
```

とします。
ログは
['cube', 'sphere', 'cylinder']
です。
他にも最初の要素から 2 つ置きに取得したい場合、

```
print objectList[::2]
```

とすると、
['torus', 'sphere', 'cone']
となります。例えば上記が奇数要素取得とした場合、偶数要素が欲しい時は、

```
print objectList[1::2]
```

と記述します。ログは、
['cube', 'cylinder']
こうなります。
他にも逆順にするにはマイナスを付けると出来ますので、

```
print objectList[::-1]
```

とすると、リストを反転して内容を取りに行きます。
['cone', 'cylinder', 'sphere', 'cube', 'torus']
この辺りが Softimage で良く使うスライス機能です。
他にもスライス記述例はたくさんありますので、その都度検索して調べてみて下さい。

リスト内を調べる

他にもよく使う機能では in 演算子があります。

これはリストの中に指定した要素が含まれているかどうかを調べる方法で、

要素 in リスト

と記述することで True か False の結果を取得できます。

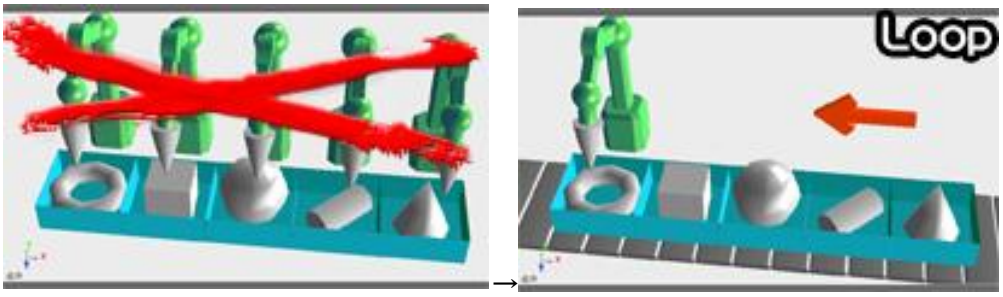
```
print 'cube' in objectList
print 'camera' in objectList
```

結果のログは上が True で下が False となります。

この結果をどう活かすのか？はまた後ほど条件分岐の項で紹介します。

3. 反復処理

反復処理とは、文字通り繰り返し同じ処理を行うということで、例えばリストになっている場合、オブジェクト全部へ一度に処理を行うことも Softimage は場合によって可能(いつか説明できるかも)ですが、全てにそのワザが使えるわけではないので、基本的には中身一つ一つに対して処理をすることになります。



反復処理の書き方は while 文と for 文の二種類がありますが、Softimage では複数のオブジェクトやパラメータに対して処理をすることが多く、そういった場合は for 文が簡単で理解し易いと思いますので、ここでは先に for 文を使って解説します。

for 文

for 文の基本的な構成は、

for 変数 in リストオブジェクト:

　　実行する処理

といった感じになります。リストオブジェクトから一つづつ中身を取り出し、変数へ入れて、処理内容へ流し込むイメージです。

また、Python ではインデント(実行する処理の前にある Tab/スペース>　　)が処理の入れ子を表す重要な意味を持ちます。

今後も頻繁に出てきますが、if 文や関数の def 内に入れ子処理を描いた場合のインデントは忘れないで下さい。

```
for num in ['0','1','2','3']:
        print num
```

これを ScriptEditor で実行してみて下さい。

```
#  0
# 1
# 2
# 3
```

なぜか頭にスペースがついてきますが、0 から順番に3まで1行ずつログに表示されます。

リスト['0','1','2','3']から一つづつ要素を取り出し print でログ表示をするという流れがイメージ出来ますでしょうか？

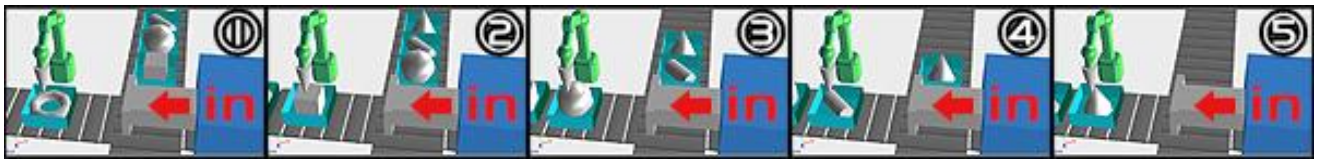
それではこれを参考に、実用へ進みます。

```
si = Application
Logm = si.Logmessage
objectList = si.Selection
for object in objectList:
        Logm(object)
```

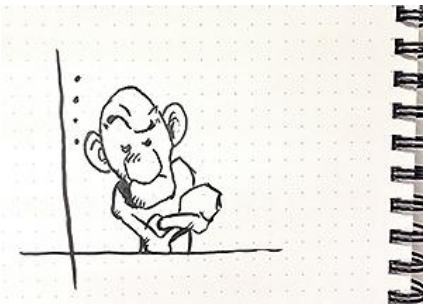
適当に作ったシーンで適用すると、選択したオブジェクト一つ一つをヒストリペインにログとして表示するという結果になると思います。

```
# INFO : torus
# INFO : cube
# INFO : sphere
# INFO : cylinder
# INFO : cone
```

in という押出機を使って処理ラインにデータを流しているイメージがつかめるようになると理解できるようになると思います。



4. 問題



最後に、今回の内容を踏まえて問題です。

まず適当にオブジェクトを複数作成し、X 方向に 10 間隔で並べて、回転値 X に 0～100 の間で 90 度回転するアニメーションを設定するスクリプトを作ってみましょう。

選択したものを for 文で処理していく形で良いと思います。

参考までにオブジェクトを複数作るのが面倒な場合、以下のスクリプトをお使い下さい。

```
si = Application
si.CreatePrim("Sphere", "MeshSurface", "", "")
si.CreatePrim("Torus", "MeshSurface", "", "")
si.CreatePrim("Cube", "MeshSurface", "", "")
si.CreatePrim("Cylinder", "MeshSurface", "", "")
si.CreatePrim("Cone", "MeshSurface", "", "")
```

5. 答え

↓

↓

↓

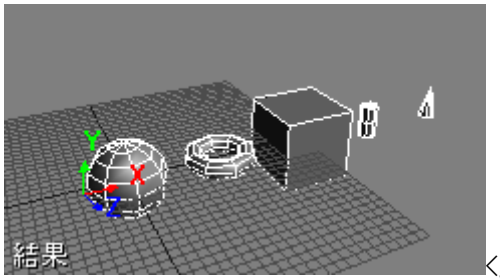
↓

↓

↓

```
# Application を短縮
si = Application

# 増分する座標は 0 からスタート
xvalue = 0

# RotationX のパラメータを格納する空のリスト"rotxlist"を作成
rotxlist = []

# for 文を使って選択オブジェクトを回す
for obj in si.selection:
        # オブジェクト一つづつの Local.posx に上記変数"xvalue"の数値をセット
        obj.Kinematics.Local.posx = xvalue
        # xvalue に 10 を足すをオブジェクト分だけ繰り返す
        xvalue = xvalue + 10
        # リスト"rotxlist"にオブジェクトの rotx パラメータを追加
        rotxlist.append(obj.Kinematics.Local.rotx)

# リスト rotxlist に対してフレーム 1 にキーを設定する
si.SaveKey(rotxlist,1,'','','',False,'')

# リスト rotxlist に 90 をセットする
si.SetValue(rotxlist,90)

# リスト rotxlist に対してフレーム 100 にキーを設定する
si.SaveKey(rotxlist,100,'','','',False,'')
```

動作すればソレで良かろうなのですが、例えば解答の一つはこんな感じの内容になっています。VBScript でよく頭文字を大文字で書きだす人は注意して下さい。Python では For ～ In ～と記述するとエラーになります。



<gif アニメ>

結果、こんな感じのシーンになっていれば正しく動いています。

おまけ

Python スクリプトは Maya でもインストール直後から使えるようになっています。
言語が共通と言うことはソースコードの共有とは行かないまでも、ひとつの言語に対しての知識だけでたりると言う利点がありますね。
例えば今回のスクリプトを Maya 用に置き換えると・・・

```
import maya.cmds as cmds

# 増分する座標は 0 からスタート
xvalue = 0

# RotationX のパラメータを格納する空のリスト"rotxlist"を作成
rotxlist = []

for obj in cmds.ls( sl=True, fl=True):
    print obj
    # オブジェクト一つづつの tx に上記変数"xvalue"の数値をセット
    cmds.setAttr( obj+".tx", xvalue )
    # xvalue に 10 を足すをオブジェクト分だけ繰り返す
    xvalue = xvalue + 10
    # リスト"rotxlist"にオブジェクトの rotx パラメータを追加
    rotxlist.append(obj+".rotateX")

# リスト rotxlist に対してフレーム 1 にキーを設定する
cmds.setKeyframe(rotxlist, time=1, value=0)

# リスト rotxlist に対してフレーム 24 に 90 度のキーを設定する
cmds.setKeyframe(rotxlist, time=24, value=90)
```

Softmage では si = Application としてアプリケーションをオブジェクト化しているのに対して、Maya はコマンドのモジュールをインポート”import maya.cmds as cmds”している点にご注目下さい。

それではみなさん、ごきげんよう。