pf-5. 式の抽象化と関数

（Python 入門，全６回）

URL:https://www.kkaneko.jp/cc/pf/index.html

金子邦彦

５－１．式の抽象化

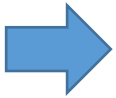
式の抽象化

100 * 1.1

150 * 1.1

400 * 1.1

類似した複数の**式**

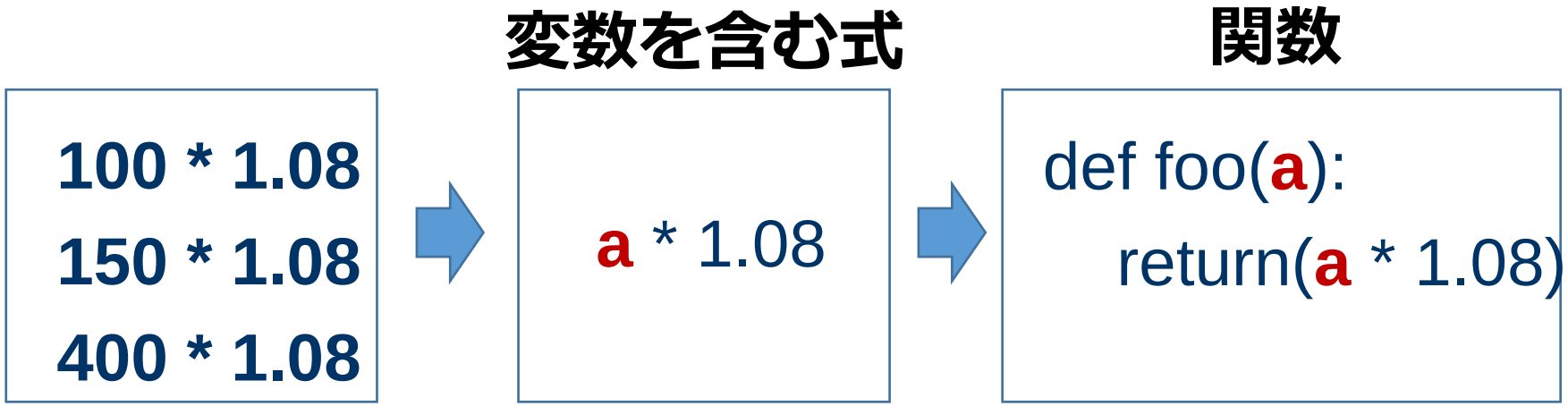


a * 1.1

変数 a を使って，複数の**式**を１つにまとめる（**抽象化**）

5－2．関数

・式


変数を含む式
関数
100 * 1.08
150 * 1.08
400 * 1.08
a * 1.08
def foo(a):
return(a * 1.08)
抽象化
「変数を含む式に
名前（関数名）を
付けたものが
『関数』である」
と見立てることも

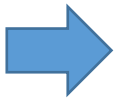
関数

100 * 1.1

150 * 1.1

400 * 1.1

類似した複数の**式**



a * 1.1

変数 a を使って，複数の**式**を１つにまとめる（**抽象化**）

```
def foo(a):
  return a * 1.1
print(foo(100))
print(foo(150))
print(foo(400))
```

式「**a * 1.1**」を含む**関数 foo** を**定義**

関数 foo を使用.
100, 150, 400 は**引数**

```
110.00000000000001
165.0
440.00000000000006
```

関数

```
def foo(a):
  return a * 1.1
```

- この**関数**の**本体**は
  「**return a * 1.1**」

- この**関数**は，式「**a * 1.1**」に，**名前 foo** を付けたものと考えることもできる

式の抽象化と関数

抽象化前

```
print(100 * 1.1)
print(150 * 1.1)
print(400 * 1.1)
```

類似した複数の**式**

```
110.00000000000001
165.0
440.00000000000006
```

抽象化後

```
def foo(a):
    return a * 1.1
print(foo(100))
print(foo(150))
print(foo(400))
```

関数の定義と使用

```
110.00000000000001
165.0
440.00000000000006
```

同じ
実行結果になる

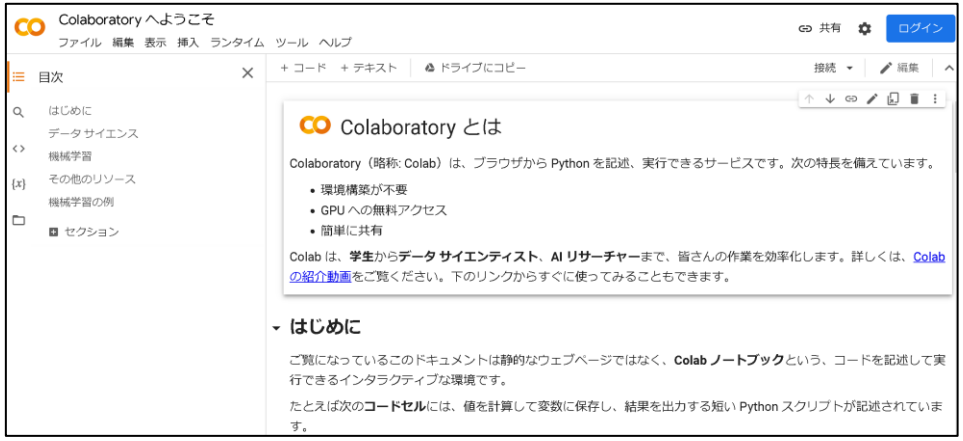
私の意見

- プログラミングでの根本問題は何でしょうか？
  - 誤り（バグ）の無いプログラムの作成
- プログラミングの一番の基礎は何でしょうか？
  - 抽象化を行うこと.
  - 抽象化により，繰り返し同じことを書くことが減り，バグを防げる.
  - プログラムの変更（消費税率10%変更）も簡単に.

Google Colaboratory の使用

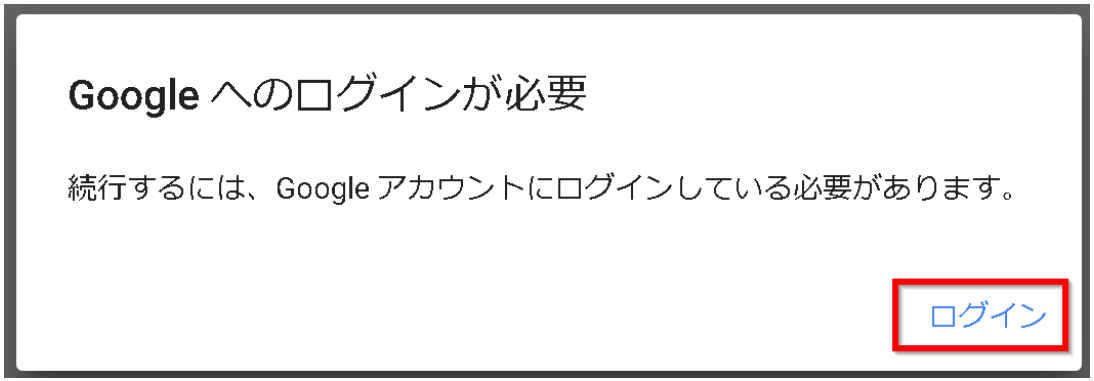
① Google Colaboratory のWebページを開く

https://colab.research.google.com



②「**ファイル**」で、「**ノートブックを新規作成**」を選ぶ

③ Google アカウントでのログインが求められたときはログインする



演習

① コードセルに，次のように入れる

```
def foo(a):
    return a * 1.1
print(foo(100))
print(foo(150))
print(foo(400))
```

すべて**半角文字**

「*」は掛け算の記号

「def foo(a)」の直後に「:」

字下げも正確に！「return a * 1.1」だけを字下げ

② 実行結果を確認

```
def foo(a):
  return a * 1.1
print(foo(100))
print(foo(150))
print(foo(400))
```

```
110.00000000000001
165.0
440.00000000000006
```

```
def foo(a):
  return a * 1.1
  print(foo(100))
  print(foo(150))
  print(foo(400))
```

これでは動かない

```
def foo(a):
  return a * 1.1
print(foo(100))
print(foo(150))
print(foo(400))
```

これは**動く**

「delキー」などを使いながら書き換えてください

関連資料

- **Python まとめページ**

https://www.kkaneko.jp/pro/python/googlecolab.html

- **Python の基本**

Google Colaboratory，Paiza.IO を使用.

https://www.kkaneko.jp/cc/colab/index.html

- **Python 入門（全６回）**

Google Colaboratoryを使用.

https://www.kkaneko.jp/cc/pf/index.html

- **Python プログラミング演習（全９回）**

Python Tutor, VisuAlgo を使用

https://www.kkaneko.jp/cc/po/index.html

- **さまざまな Windows アプリケーションのインストールと設定**

https://www.kkaneko.jp/cc/tools/index.html