基本的な用語の復習

2019/10/16

木南　貴志

最初に

今回は基本的な用語を復習しますが、用語の説明では明確な定義を用いずに**イメージ**で説明する場合があります。

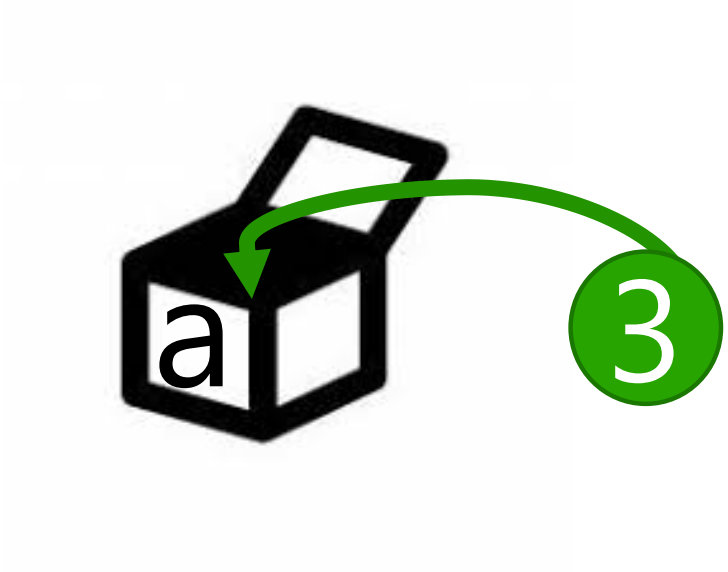
（明確な定義が知りたい人はネットで調べてください）

また、プログラムはPythonの文法で示します

変数(*variable*)

数値、文字列などを収納する箱のようなもの

例) a = 3



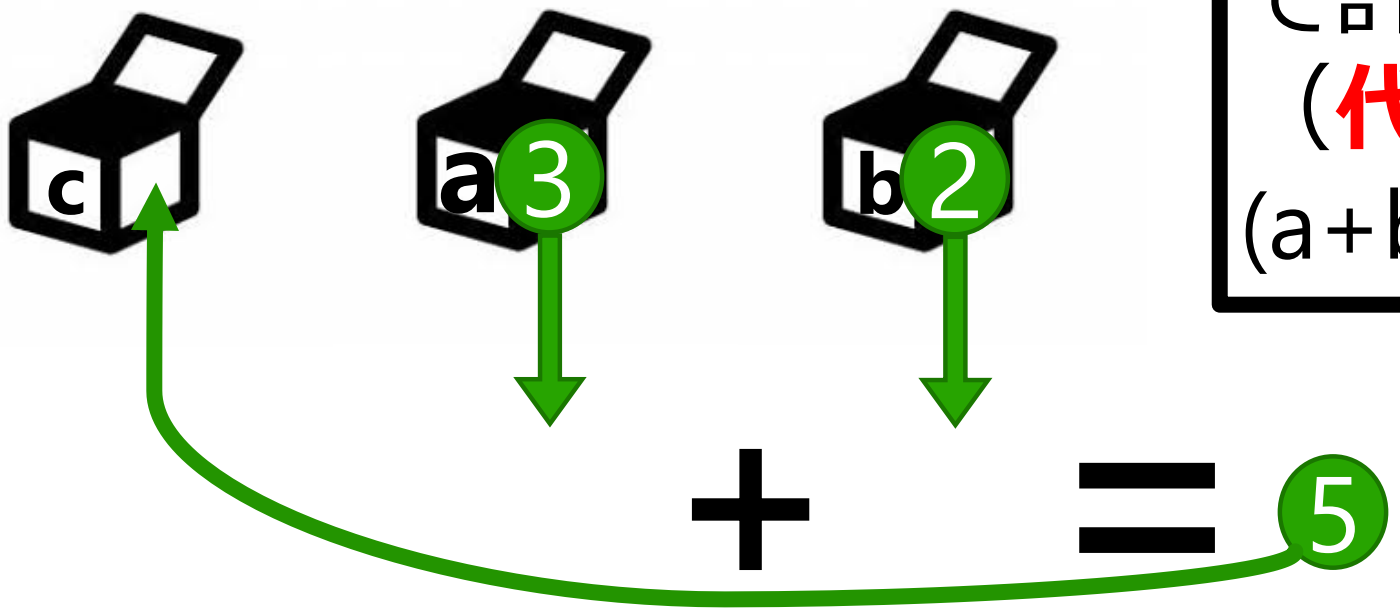
イメージ
aという箱を用意して、
その箱に3を収納（**代入**）
する

変数(*variable*)

数値、文字列などを収納する箱のようなもの

例）

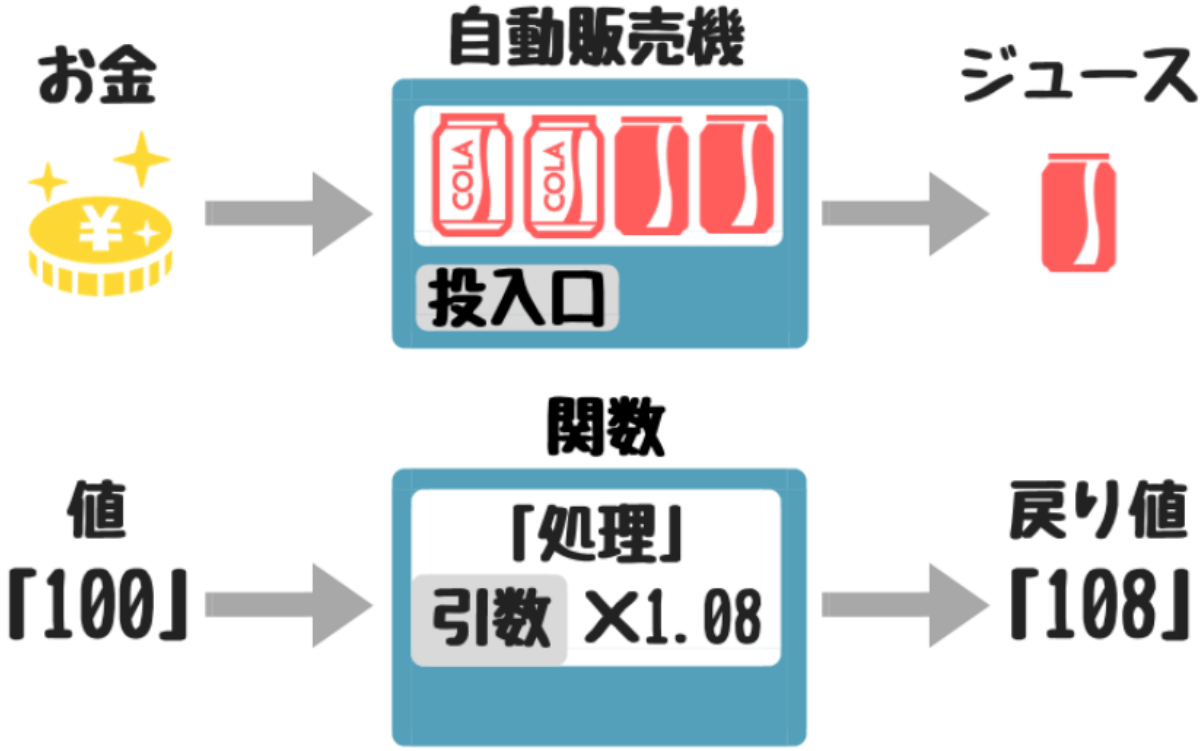
```
a = 3
b = 2
c = a + b
```



イメージ
aとbから3と2を取り出して計算したものをcに収納（**代入**）する
(a+b=cではなくc=a+b)

関数(*function*)

関数のイメージ

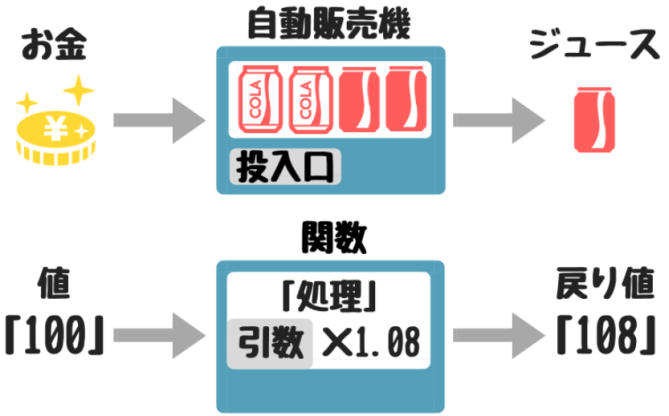


https://kitsune.blog/function

関数(*function*)

プログラミングで記述すると下記のようになる

```
def 関数名（引数１、引数２…）：
    処理
    return 戻り値
```

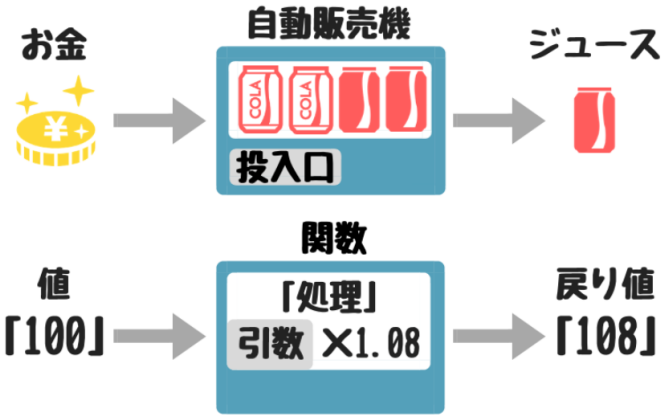


関数(*function*)

例）与えた**引数**に1.08を書けたものを**戻り値**として返す

（消費税込み(8%）の金額を計算する関数）

```
def tax(a):
    result = a*1.08
    return result
```



関数(*function*)

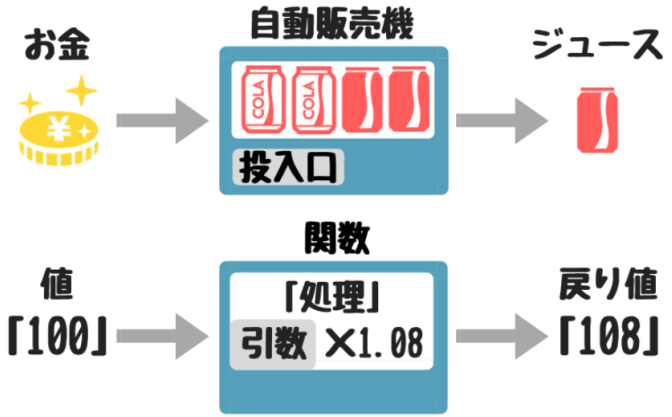
例）これをmain文に組み込むと

```
def tax(a):
    result = a*1.08
    return result
def main() :
    b = tax(100)
    print(b)
main()
```

出力結果

```
108.0
```

① 変数aに100を代入
② 処理
　1) 100*1.08を計算
　2) 結果をresultに格納
③ resultを返り値として返して、bに格納する



関数(*function*)

例）これをmain文に組み込むと

```
def tax(a):
    result = a*1.08
    return result
def main():
    b = tax(100)
    print(b)
main()
```

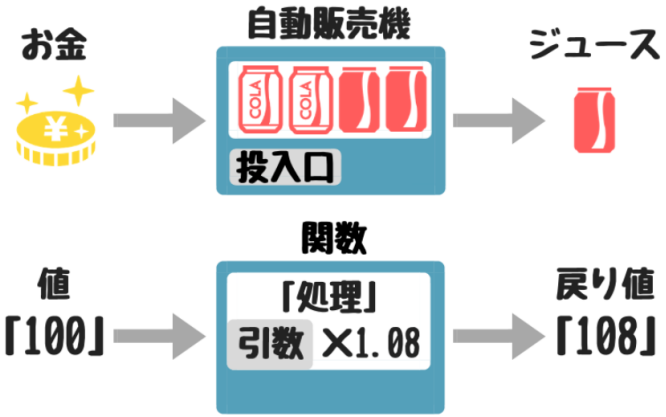
② 処理

① ③

出力結果

```
108.0
```

① 変数aに100を代入
② 処理
　1) 100*1.08を計算
　2) 結果をresultに格納
③ resultを返り値として返して、bに格納する



関数を使う利点

１.定義しておけば使いまわせる

例えば、100円の物の税込み価格と、250円の物の税込み価格を計算したいときにあらかじめ定義しておけば同じような文を2回書く必要がない。

関数を定義した場合

```
def tax(a):
        result = a*1.08
        return result
def main() :
        b = tax(100)
        c = tax(250)
        print(b)
        print(c)
main()
```

関数を定義しない場合

```
def main() :
        b = 100*1.08
        c =  250*1.08
        print(b)
        print(c)
main()
```

同じ計算を2回記述する必要がある

（簡単な計算の場合その利点が分かりにくいが複雑な計算をする場合には関数を定義しておいたほうが良い場合が多い）

関数を使う利点

（前ページよりは）利点が分かりやすい例（最大値を求める）

関数を定義した場合

```
def m(a, b, c):
        maximum = 0
        if (maximum < a):
                maximum = a
        if (maximum < b):
                maximum = b
        if (maximum < c):
                maximum = c
        return maximum
def main() :
        b = m(3,10,1)
        c = m(86,2,40)
        print(b)
        print(c)
main()
```

関数を定義しない場合

同じ計算を2回記述する必要がある

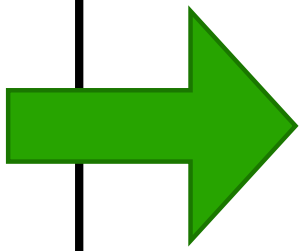
```
def main() :
        maximum = 0; a = 3; b = 10; c = 1
        if (maximum < a):
                maimum = a
        if (maximum < b):
                maximum = b
        if (maximum < c):
                maximum = c
                print(maximum)

        maximum = 0; d = 86; e = 2; f = 40
        if (maximum < d):
                maimum = a
        if (maximum < e):
                maximum = b
        if (maximum < f):
                maximum = c
                print(maximum)
```

関数を使う利点

2.main文を見るだけでやっているおおまかな内容が分かる

```
def tax(a):
        result = a*1.08
        return result
def main() :
        b = tax(100)
        c = tax(250)
        print(b)
        print(c)
main()
```



関数名がtax
↓
税金の計算をしていると予想できる

プログラムを後で見返すときや、共同でプログラムを作るときに便利

コメントアウト

プログラムの説明をするために**コメントアウト**を用いる

コメントアウトされた行はプログラム実行時には反映されない

方法：コメントアウトしたい行の頭に#（シャープ）をつける

```
#8%の税込み価格を計算する
def tax(a):
    result = a*1.08
    return result
```

この行はプログラム実行時には、反映されない

後で見返す時や
共同で作業をする時に便利

https://blog.codecamp.jp/python-comment

コメントアウト

また、一時的にコードを無効化したい時（試行錯誤したい時など）

にもコメントアウトは用いられる

```
def tax(a):
  result = a*1.08


    return result
```

https://blog.codecamp.jp/python-comment

コメントアウト

また、一時的にコードを無効化したい時（試行錯誤したい時など）にもコメントアウトは用いられる

```
def tax(a):
  #result = a*1.08
  result = a*1.10
    return result
```

違う計算も試してみたいので
一時的にコメントアウト

https://blog.codecamp.jp/python-comment

コメントアウト

複数行コメントアウトしたい時は　“（ダブルクォーテーション）

もしくは‘（シングルクォーテーション）3つで囲う

```
"""
def tax(a):
    result = a*1.08
    return result
"""
```

https://blog.codecamp.jp/python-comment

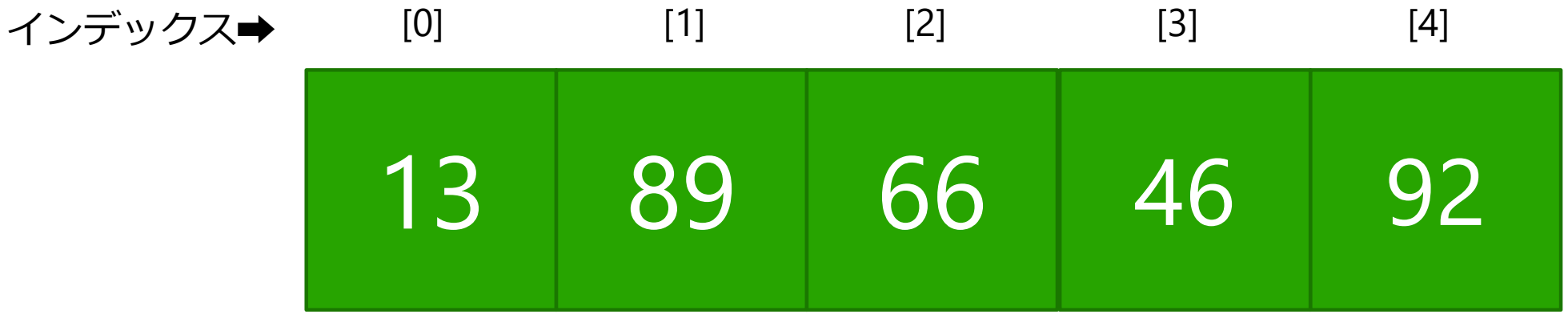
list

配列：同じ型の要素を一列に並べたもの

要素を格納した順番に0番目、1番目、2番目・・・というふうに番号がつけられる

この番号は**インデックス**と呼ばれる

（下の図の場合0番目の要素が13、1番目の要素が89、2番目の要素が66・・・というふうになる）



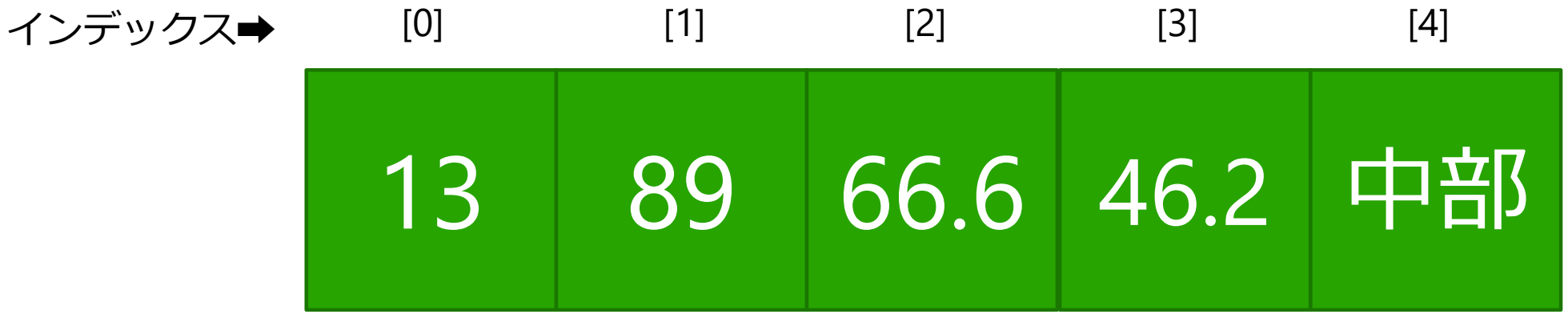
https://www.sejuku.net/blog/32681

list

pythonでは配列ではなく**list**と呼ばれる

配列との違い：list内を構成する要素は同じ型でなくても成り立つ

（下の図では0•1番目の要素はint（整数）型、2•3番目の要素はfloat（浮動小数点）型、4番目の要素はstr(文字列）型になる）



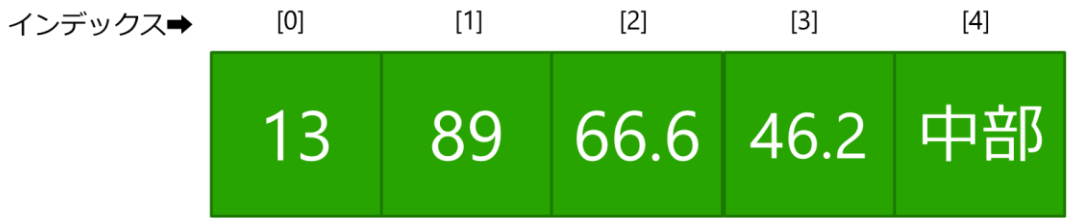
https://www.sejuku.net/blog/32681

list

例）下記はaというlist内に格納されている0番目の要素、2番目の要素、4番目の要素を出力したもの

```
a = [13, 89, 66.6, 46.2, “中部”]

print(a[0],a[2],a[4])
```



出力結果

```
13 66.6 中部
```

https://www.sejuku.net/blog/32681

変数と配列（*list*）の違い

変数

配列

配列は変数を一列に並べたもの

list 要素の追加と削除

既存のlistに新たな要素を追加するときは**appendメソッド**を使用する

（下記はaというlistに新たな要素「8」を追加した※末尾に追加される）

```
a = [1, 2, 3, 4, 5]
a.append(8)
print(a)
```

追加する値を指定

出力結果

```
[1, 2, 3, 4, 5, 8]
```

https://www.sejuku.net/blog/32681

list 要素の追加と削除

既存のlistの**任意の位置**に新たな要素を使用するときは**insertメソッド**を使用する

（下記はaというlistの3番目に100を追加した）

```
a = [1, 2, 3, 4, 5]
a.insert(3, 100)
print(a)
```

(追加する位置,　追加する値)の順番で指定

出力結果

```
[1, 2, 3, 100, 4, 5]
```

https://www.sejuku.net/blog/32681

list 要素の追加と削除

既存のlistの要素を削除するときは**removeメソッド**を使用する

（下記はaというlist内の要素30を削除した）

```
a = [88, 22, 33, 7, 11]
a.remove(22)
print(a)
```

削除したい要素の値を指定

出力結果

```
[88, 33, 7, 11]
```

https://www.sejuku.net/blog/32681

list その他色々

- すべての要素を削除： clear
- list並び順を反転：reverse
- 要素数（長さ）を取得：len
- 2つのlistを結合：extend
- 要素の検索：in　　など

自分の実現したいプログラムによって使用するメソッドなども変わるので自分で色々調べてみましょう

https://udemy.benesse.co.jp/development/web/python-list.html