守護神でもできるプログラミング講座

No.6

D3 村尾響

今日の目標

- 複数の値をまとめて扱えるようになろう
- 繰り返した処理ができるようになろう
- 処理をまとめてコードをすっきりさせよう

OUTLINE

1. リスト
2. ループ
3. 関数(補足)
4. 告知

1. リスト

リストとは？

Pythonで複数の数値や文字列をまとめて扱うことができるもの

C言語の配列のようなもの

1. リスト

リストを作成するには

数値を , で区切って[ ]で囲む

a = [1, 2, 3, 4]

とすると1や2や3や4といった数値(データ)が入ったリスト変数aが作成できる

文字列も同様

Task = [“Twitter”, “レッポヨ”, “プロロアミンg”,]

1. リスト

リストの中身が1つでも無くても大丈夫

one = [1.5]

zero = [ ]

のようにすることができる

また、数値と文字列を混ぜてリストに入れても扱える

mix = [“0”, 0, “Python”, “お給料ほしい///”]

1. リスト

x = [1, 2, 3, 4]
とすると
1が **0** 番目　　2が **1** 番目
3が **2** 番目　　4が **3** 番目
のデータになる

0から始まることに注意！

x[i]　　(この場合のiは0～3)とすると
xの中のi番目のデータを得ることができる

1. リスト

x = [1, 2, 3, 4]

print(x)
とすると
[1, 2, 3, 4]
のように
xの中のデータ全てが
表示される

print(x[2])
とすると
3
のように
xの中の2番目のデータ
(3)が表示される

1. リスト

x = [1, 2, 3]　　のとき、

x[1] = -2　　とすると、

xの1番目のデータ(2)が-2に変更される

このとき、xは

[1, -2, 3]

になる

1. リスト

補足: リスト変数のキーボードから入力

キーボードから入力した値からリスト変数を作るには、文字列のときは

lista = input().split()

整数の時は

listb = list(map(int, input().split()))　とすると

〇1 〇2 〇3 〇4　　(〇1～4は文字列か整数)

と入力したときに　[〇1, 〇2, 〇3, 〇4]

というリストが作成される

1. リスト

補足: リストの機能

リストには
途中からデータを追加するappendメゾット
中のとあるデータを削除するremoveメゾット
要素を検索するin演算子
要素の数を取得するlen関数がある

気になったらググってみよう

演習1. リスト

1. x = [1, 2, 3]　というリストを用意して、xをまとめて表示したり一つずつ表示したりしよう

2. x = [3, 2, 1]　というリストを用意して、
   xを[1, 2, 3]にして表示しよう

2. ループ

ループとは？

ループとは繰り返しのことである
プログラミングのループ文の代表として
forループとwhileループがあげられる

実際にループを使えるようになろう

2. ループ

for文とwhile文

```
for  i  in  range(10):
      print(“www”)
```

とすると10回ループして”www”と出力する

```
x  =  0
while  x  <  10:
      print(x)
      x  =  x + 1
```

とするとxが10未満の間ループしてxの値を出力する

2. ループ

for文

```
for  i  in  range(10):
        print(i)
```

とすると、iの値が1ずつ増えているのが確認できる

実際にやってみよう

2. ループ

iの値1つずつ増えていることを利用して、リストのデータを1つずつ出力することができる

```
x = [“一夜漬け”, “赤点”, “留年”]
for i in range(3):
    print(x[i])
```

とすると

xの0番目、xの1番目、xの2番目

と順番に出力される

2. ループ

これと同じことを以下のようにしてもできる

```
x = [“夏休みも”, “勉強”, “しようね！”]
for i in x:
    print(i)
```

とすると

iにxの0番目のデータ、xの1番目のデータ、xの2番目のデータが順番に代入され、

それが出力される

2. ループ

while文

```
a = 1
while a != 0:
    a = int(input())
    print(a)
```

とすると、aに0が入力されない限り永遠とループをし続ける

実際にやってみよう

2. ループ

前のコードを実行すると最後に0を入力したときに0が出力されるが以下のようにすると最後の0が出力されなくなる

```
a = 1
while True:
    a = int(input())
    if a == 0:
        break
    print(a)
```

2. ループ

前のコードで

break

というものがあったが、これはbreakが呼ばれた時点でループを強制終了するものである

他にも

continue

というものがあり、これはcontinueが呼ばれた時点で強制的にループの1番上に戻ることである

2. ループ

continueの意味合いは異なってくるが、

breakやcontinueはfor文でも使える

breakやcontinueはループされない、無限ループの可能性を引き起こすので必ずif文を使って呼び出そう

演習2. ループ

1. for文を使って2から順番に4,6,…と2ずつ増やして表示するプログラムを作成しよう

2. x = [3, 6, 9]　というリストとfor文を使って順番に表示するプログラムを作成しよう

3. while文を使ってaに0が代入されるまで入力された整数値の絶対値を表示し続けるプログラムを作成しよう

3. 関数

以降補足なのでやりたい人は見てください

3. 関数

関数とは？

複数の処理をまとめたもの

・まとめた処理を簡単に呼び出すことができる

・まとめたものをほかのファイルに保存して
ファイルを分けることができる

3. 関数

関数の作り方

以下のようにして関数を作ることができる

```
def 関数名():
    処理1
    処理2...
```

3. 関数

```
def  print_hello():
        print(“Hello”)
        print(“Python3!”)
```

というものを作っておけばこの後から

print_hello()

を呼び出すだけで

”Hello”と”Python3”が出力される

3. 関数

```
def  print_hello():
        print(“Hello”)
        print(“Python3!”)
print_hello()
```

3. 関数

戻り値

任意のの2つの数値を足し算する関数を考える

足し算の結果を後で使いたいとき、関数を呼び出したところに値を渡さなければならない

この渡す値を戻り値といい

return 戻り値

のように書く

3. 関数

```
def  add(x, y):
      z  =  x + y
      return  z
a  =  add(2, 6)
b  =  12
c  =  add(a, b)
```

とすると1回目にaddが呼び出されたときはxに2が、yに6が代入されてzは8になり、これがaに代入される

4. 告知

図書館で本借りよう！！