Python によるプログラム

1 簡単なプログラム

簡単に Python の機能を確認したりする時は Jupyter Notebook が役に立ったが、実際のプログラムはファイルに書き込んでいく。

今回は 0~9 までの数の和を求めるプログラムを考える。

以下、ファイル名を指定するので、そのファイルを作成し、書き込んで実行してみて欲しい。

test_sum_10.py

```
x = 0
for i in range(10):
   x = x + i
print(x)
```

まずは上のプログラムを「python test_sum_10.py」と統合ターミナルに入力して実行してみて欲しい。0~9 までの和が出力されるはずだ。

さて、このプログラムは「0~9」までの和を求めるが、「0~20」までの和を求めるプログラムに変更したい。そのためにはどうすればよいだろうか？

例えば、以下のようなプログラムが考えられる。

test_sum_20.py

```
x = 0
for i in range(20):
   x = x + i
print(x)
```

しかし、「range」の中身を毎回書き換えるのはいかにもエレガントではない。ここにも変数を用いてはどうだろうか？

test_sum_n.py

```
n = 20
x = 0
for i in range(n):
    x = x + i
print(x)
```

しかし、この 「test_sum_n.py」を用いても、「n」の値を変更するには毎回プログラムを書き直す必要がある。 この問題を解決するには「input」関数を用いると良い。「input」関数を使ったバージョンに書き換えよう。

test_sum_n.py（書き換え）

```
n_str = input(">>>")
n = int(n_str)
x = 0
for i in range(n):
    x += i
print(x)
```

このプログラムを実行しても、最初「>>>」と表示されるだけである。ここで、例えば、「10」と打ち込み、エンターキーを押してみて欲しい。「45」と帰ってくるはずである。

1.1 課題 1

最初の行で `n_str = input(">>>")` と書かれているが、`n_str` の型と値を答えよ。また、`n` の型と値を答えよ。

ヒント: この問題を解くにあたって、`print` と `type` をプログラム中に埋め込んで実行してみると考えの助けになる。

2 function

さて、「input」関数の使い方が理解できたところで、さらなるプログラムの改良を行いたい。プログラムの見通しを良くするため、手続きに名前を付けて別のソースから隔離することがある。この機能を「関数」と言う。これまで用いてきた「range」や「print」、「input」も実は関数である。

sumdef.py

```
def summation(max_n):
    x = 0
    for i in range(max_n):
        x += i
    return x

n_str = input(">>>")
n = int(n_str)
s = summation(n)
print(s)
```

このプログラムは短くすると以下のように縮約できるので、プログラムを書き直して見比べて欲しい。

sumdef.py（変更）

```
def summation(max_n):
    x = 0
    for i in range(max_n):
        x += i
    return x

print(summation(int(input(">>>"))))
```

3 import

さて、「summation」という関数を定義したが、他のプログラムからこの関数を利用したいと思う日が来るだろう。このために「import」という機能が用意されている。とりあえず、import の機能を使ってみよう。

test_import.py

```
import sumdef
```

「test_import.py」を実行すると、「sumdef.py」を実行した時と同じになるだろう。そう、「import」することで別のファイルを実行することが出来るのである。

今回、「import」された「summation.py」がまるまる実行されたため、「`print(summation(int(input(">>>"))))`」の部分も実行された。

しかし、場合によってはここはいらず、「summation」関数のみを利用したと思うこともあるだろう。例えば以下のような場合だ。

test_import.py（変更）

```
import sumdef


for i in range(int(input(">>>"))):
    print(sumdef.summation(i))
```

ここで、sumdef.summation と言うのは、「sumdef.py の中の summation 関数」ということである。このファイルを実行してみて欲しい。何かがおかしいだろう。

そう、sumdef.py の input が実行されてしまっているがために、二回 input することを求められてしまうのだ。これは良くない挙動だ。これを回避するための方法が用意されている。

sumdef.py（変更）

```
def summation(max_n):
    x = 0
    for i in range(max_n):
        x += i
    return x

if __name__ == '__main__':
    print(summation(int(input(">>>"))))
```

「if __name__ == '__main__':」という一文を入れることで、その下の内容は import された場合は実行されなくなる。もちろん import せずに直接 sumdef.py を実行した場合はこの部分も実行される。

また、import の方法は多岐にわたる。「test_import.py」の別バージョンをいくつか紹介しておく。

test_import.py（別バージョン）

```
from sumdef import summation

for i in range(int(input(">>>"))):
    print(summation(i))
```

test_import.py（別バージョン）

```
import sumdef as sm

for i in range(int(input(">>>"))):
    print(sm.summation(i))
```

他にも以下の方法があるが、非推奨である。

```
from sumdef import *

for i in range(int(input(">>>"))):
    print(summation(i))
```

4 おまけ

help という関数が非常に便利である。

```
help(print)
```

英語であるが、関数等の説明を見ることが出来る。困ったら help してみよう。

5 おまけ 2

Python には sum という関数があり、これを使うことで、これまでに学んだ内容は

```
print(sum(range(int(input(">>>")))))
```

で表される。

5.1 課題

いまの実装ではsummation(10)は 45 を返す。つまり、10 は足されていない。プログラムを改造して、最後の数も足されるようにせよ。