第 6 回　繰り返し処理

前回は、条件分岐の if 文について学んだ。今回は、繰り返し処理について学ぶ[1]。

【for 文を使った繰返し処理】
for 文を使った繰り返し処理の構造を示す。

```
for 変数名 in イテラブルオブジェクト:
    繰り返す処理
```

ここで「イテラブルオブジェクト」とは、「繰り返し可能なオブジェクト」ということである。繰り返す(iterate)ことが可能という意味の形容詞が iterable である。「イテラブルオブジェクト 」という言葉は少し難しいので、具体例で考える。

for 文の例（0,1,2,3,4 を順番に表示する）

```
for i in range(5):
    print(i)
```

ここで使われている range() という関数が、連続した数値の「イテラブルオブジェクト」を生成する関数である。整数 n に対して range(n)とすることで、0 から n-1 までの順番に連続した整数を生成する。したがって、range(5)では 0, 1, 2, 3, 4 という順番に連続した整数が生成される。この連続した整数を、順番に i という変数に代入して「繰り返す処理」である print(i)を実行する。ここで、range(5)は 0 から 4 までの整数なので、全部で 5 個あり、繰り返しは 5 回実行される。
range(n)は 0 から n-1 までの数を返すが、range(a,b)とすれば a から b-1 までの数を返す。また、range(a,b,c)とすると、a から b に届く前までを c おきに返す。

例：range(2,5) ： 2,3,4
　　range(2,7,2) ： 2,4,6
　　range(2,8,2) ： 2,4,6
　　range(8,2,-1) ： 8,7,6,5,4,3
　　range(8,2,-2) ： 8,6,4
　　range(8,3,-2) ： 8,6,4

★次のプログラムを実行してみましょう。入力には整数（たとえば 10）を 1 つ入れます。

```
n = int(input())
sum = 0
for i in range(1,n+1):
    sum += i
print(sum)
```

[1] コンピュータの制御構文（プログラムの実行順序を制御する方法）の中で、最も基本的なものが条件分岐（選択）と繰り返し（ループ）である。条件分岐と繰り返しがあれば、あらゆるプログラムの構造を書き換えることができ、条件分岐は繰り返しで書き換え可能であるため実は繰り返しだけでも良い。

これは、何をするプログラムでしょうか？プログラムと実行結果を見て、考えてみよう。

【while 文を使った繰返し処理】

繰り返し処理をする方法には、for 文だけでなく、while 文を使う方法がある。そこで、次に while 文について学ぶ。

while 文の構造

```
while 条件式:
    繰り返す処理
```

「条件式」は、繰り返しを続ける条件を示した式である。ここに書いた式が満たされている間に限り（条件式の評価結果が True になっている間に限り）、繰り返しの処理が実行される。前回学習した、if 文の構造と見比べてみよう。

```
if 条件式:
    条件が成り立つ時の処理
```

while と if の比較

（共通点）while 文も、if 文も「条件式」の判断をして、その値が True になっている間に限り処理が実行される。
（相違点）while 文では、条件式が成り立つ間、何回でも処理が繰り返される。if 文では、処理が実行されるのが 1 回だけ。

while 文の例（0,1,2,3,4 を表示する）

```
i = 0
while i < 5:
    print(i)
    i += 1
```

1. i = 0 で、i に整数 0 が代入される。
2. 条件式(i<5)を評価する。正しい(True)ので、繰り返し処理が実行され、i の値 0 が表示される。
3. i += 1 が実行される。これは「変数 i の値を 1 増やす」という意味で、i の値が 1 になる。
4. while 文に戻って条件式(i<5)を評価する。正しい(True)ので、繰り返し処理が実行され、i の値 1 が表示される。
5. i += 1 が実行される。i の値が 2 になる。
6. while 文に戻って条件式(i<5)を評価する。正しい(True)ので、繰り返し処理が実行され、i の値 2 が表示される。
7. i += 1 が実行される。i の値が 3 になる。
8. while 文に戻って条件式(i<5)を評価する。正しい(True)ので、繰り返し処理が実行され、i の値 3 が表示される。
9. i += 1 が実行される。i の値が 4 になる。
10. while 文に戻って条件式(i<5)を評価する。正しい(True)ので、繰り返し処理が実行され、i の値 4 が表示される。
11. i += 1 が実行される。i の値が 5 になる。

12. while 文に戻って条件式(i<5)を評価する。誤っている(False)ので、繰り返し処理は実行されず、while 文が終了する。

★次のプログラムが、先ほどのプログラムと同じように動作することを確認してみよう。

```
n = int(input())
sum = 0
i = 0
while i <= n:
    sum += i
    i += 1
print(sum)
```

【複合文のネスト】

if, for, while などの複合文は、複合文のブロックの中に別の複合文を書くことで、多重の入れ子構造にすることができる。このような入れ子構造のことを「ネスト」と言う。for, while 文をネストすることで、多重ループによる繰り返し処理ができる。

★次の掛け算九九を表示するプログラムを実行してみましょう。

```
for i in range(1,10):
    for j in range(1,10):
        print("{0} * {1} = {2}".format(i,j,i*j))
```

【ループを途中で抜ける】

ループを途中で抜けるには、break を使う。

```
for i in range(10):
    if i == 3:
        break
    print(i)
```

【三角形を表示するプログラム】

★次の左下が直角の三角形を表示するプログラムを実行してみましょう。

```
n = int(input())
for i in range(1,n+1):
    print ("*"*i)
```

入力：5
出力：*
**
***
****
*****

ここで、print ('*'*i) という文では、文字列の繰り返し演算が使われている。

“文字列”＊ 回数

と書くことで、同じ文字列を指定回数だけ繰り返した文字列となる。

【★課題】
直前のプログラムの赤字の箇所を修正して，左上が直角の三角形を表示するプログラムを作成して､そのプログラムをToyoNet-Aceから提出して下さい｡3以外の入力も試すこと。

入力：3
出力：***
　　**
　　*

【発展：ドキュメントと外部モジュール】
Pythonのプログラミングでわからないところは、まずは公式のドキュメントを確認する。公式のドキュメントは、かなりの部分が日本語に訳されている[1]ので、この授業でPythonの扱い方に慣れてきたら、まずは「チュートリアル」を一通り読むことから始めると良いであろう。具体的な関数の使い方などについて詳しく知りたいときには、「ライブラリーリファレンス」「言語リファレンス」から該当する説明を探すのが良い。この授業でこれまでに解説してきた事項も、これらのドキュメントの該当する箇所を読むことで、さらに理解が深まるであろう。あるいは、ドキュメントを読んで理解できるようになることが、この授業の目標到達点であるとも言える。
Python は標準ライブラリだけでもかなり便利に使うことができるが、外部モジュールを読み込むことでさらに便利に使うことができる。外部モジュールは、まずはシステムにインストールしてから、プログラム内で import して使う。モジュールのインストールには、pip というシステムを使う。詳しくは「Python モジュールのインストール」を参照。Paiza を使う場合には、Paiza にインストールされているモジュールでなければ直接 import して使うことはできない。Python は世界中で多くの人が使っているため、質の高いモジュールが数多くあり、多くの人たちによってメンテナンスされている。せっかく Python を使うのであれば、そのようなモジュールをうまく活用して効率良くプログラミングすることでそのメリットを享受できる。そのような外部モジュールについては、詳しくはモジュールのドキュメントを確認することになるので、ドキュメントを探して読み解くことのできる能力は、Python プログラミングにおいてとても重要である。
公式のドキュメントを読んでも十分にわからないことは、インターネットで検索することで解決することもある。なるべく、知りたいことを具体的にしぼってキーワードで検索すると良い。たとえば、エラーメッセージがある場合には、エラーメッセージをそのまま入れると良い検索結果が得られることがある。特にPythonは世界中に多くのユーザーがいるため、日本語で検索しても十分に良い検索結果が得られなくても、英語で検索すると良い結果が得られることがある。英語に自信がない人も、最近は機械翻訳の精度が上がっているため、機械翻訳を使えば十分に理解ができるであろう。ただし、いきなりネットで検索してもあまり質の高い結果が得られない場合があるので、まずは公式ドキュメントを読んで関連する基本事項をよく理解することが重要である。

[1] https://docs.python.org/ja/3/