【発展：ウェブアプリケーション開発】

ウェブでアプリケーションを実行するための技術には様々なものがある。大きく分けると、サーバー側（サーバーサイド）で動かすプログラムと、ウェブブラウザ（クライアントサイド）で動かすプログラムがある。

Python でサーバーサイドの Web アプリを開発するときには、Django（ジャンゴ）、Flask（フラスク）などの「フレームワーク」を導入するのが一般的である。Python のフレームワークには、ログイン機能などのウェブアプリで標準的に使われる機能が備わっているため、効率的にアプリを開発することができる。

サーバーにデータを保存する必要がなければ、サーバーの環境を選ばないクライアントサイドプログラミングが手軽である。クライアントサイドプログラミングには、歴史的には Java アプレットや Adobe Flash など様々な技術があった。JavaScript は当初はブラウザによる実装が異なるという問題もあったが、今では多くのブラウザで ECMAScript 標準に準拠する JavaScript が動作することから、クライアントサイドプログラミングといえばほぼ JavaScript 一択となっている。サーバーサイドプログラミングでも、フォームに入力するときの入力チェックなど、必要に応じて JavaScript が併用されるため、ウェブアプリを開発するのであれば JavaScript は HTML, CSS と並んで必修科目である。

ウェブブラウザで直接開発することができる点も手軽である。たとえば Chrome ではブラウザを右クリックして「検証」を選ぶことで Console が表示され、直接 Javascript を実行することができる。

私が JavaScript で作成したパズルを 2 つ紹介する。

(1) 15 パズル[1]

ソースコード[2]を GitHub で読むことができる。

(2) ナンプレ[3][4]

Python でナンプレを解き、問題を作成するプログラムを作成し、そのプログラムで自動作成した問題を難易度別問題集として JavaScript プログラムで出題している。

[1] https://sekika.github.io/2020/01/17/15Puzzle/
[2] https://github.com/sekika/sekika.github.io/blob/master/js/15.js
[3] https://sekika.github.io/kaidoku/
[4] ナンプレは世界的には sudoku として有名であるが、「数独」という名称はニコリが商標登録しているため、日本国内ではニコリが関与しないものはナンプレと表記される。

第 14 回　数値計算アルゴリズム（モンテカルロ法）

今回と次回は、数式などの近似解をコンピュータで求める手法を学ぶ。

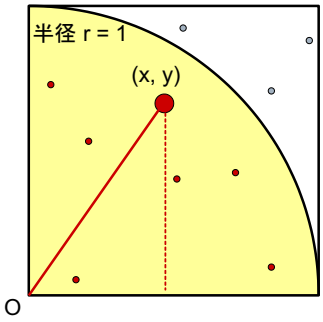
【モンテカルロ法】

まずは、シミュレーションにより近似解を求める「モンテカルロ法」について学ぶ。モンテカルロ法は、多くの回数シミュレーションを実行することで、解析的に解くことが困難な問題であっても近似的な解を求める手法である。

モンテカルロ法は物理学、統計学、工学、生物学などの分野で使われる。金融の分野では、コーポレート・ファイナンスや数理ファイナンスの分野で使われる[1]。

ここでは、円周率πをモンテカルロ法により次の手順で求めるプログラムを考える。

(1) 辺の長さが 1 の正方形を考え、この正方形の内側に n 個の点をランダムに打つ。
(2) この正方形の左下の点を中心とする半径 1 の円の 1/4 部分を考え、この 1/4 円の中に入った点の数 count を数える。
(3) count を n で割った値は、1/4 円の面積π/4 の近似値と考えられる。この関係を利用してπを近似的に求める。



【★課題】

モンテカルロ法によりπを求めるプログラムを????の箇所を修正させることによって完成させて、ToyoNet-ACE から提出してください。モンテカルロシミュレーションでπの近似をするという趣旨のプログラムであるから、pi = のところに直接πの近似値を書くような方法ではこの課題を解いたことにはならず、不正解となる。

```
import random
n = int(input())
count = 0
for i in range(n):
    x = random.random()
    y = random.random()
    dist = ????  # (x,y)の原点からの距離の2乗
    if dist <= 1:
        count += 1
pi = ????
print(pi)
```

入力：10
出力：3.2（乱数が使われているので、実行するたびに値が変わる）

変数 dist は原点からの距離ではなくて、その 2 乗としているが、dist <= 1 で「距離の 2 乗が 1 以下」つまり「距離が 1 以下」となる。Random モジュールの random.random 関数は、53 ビット精度の浮動小数点で 0 以上 1 未満の一様乱数を返す関数である[2]。

---

[1] https://en.wikipedia.org/wiki/Monte_Carlo_methods_in_finance
[2] https://docs.python.org/ja/3/library/random.html