문제해결프로그래밍실습(CSE4152)

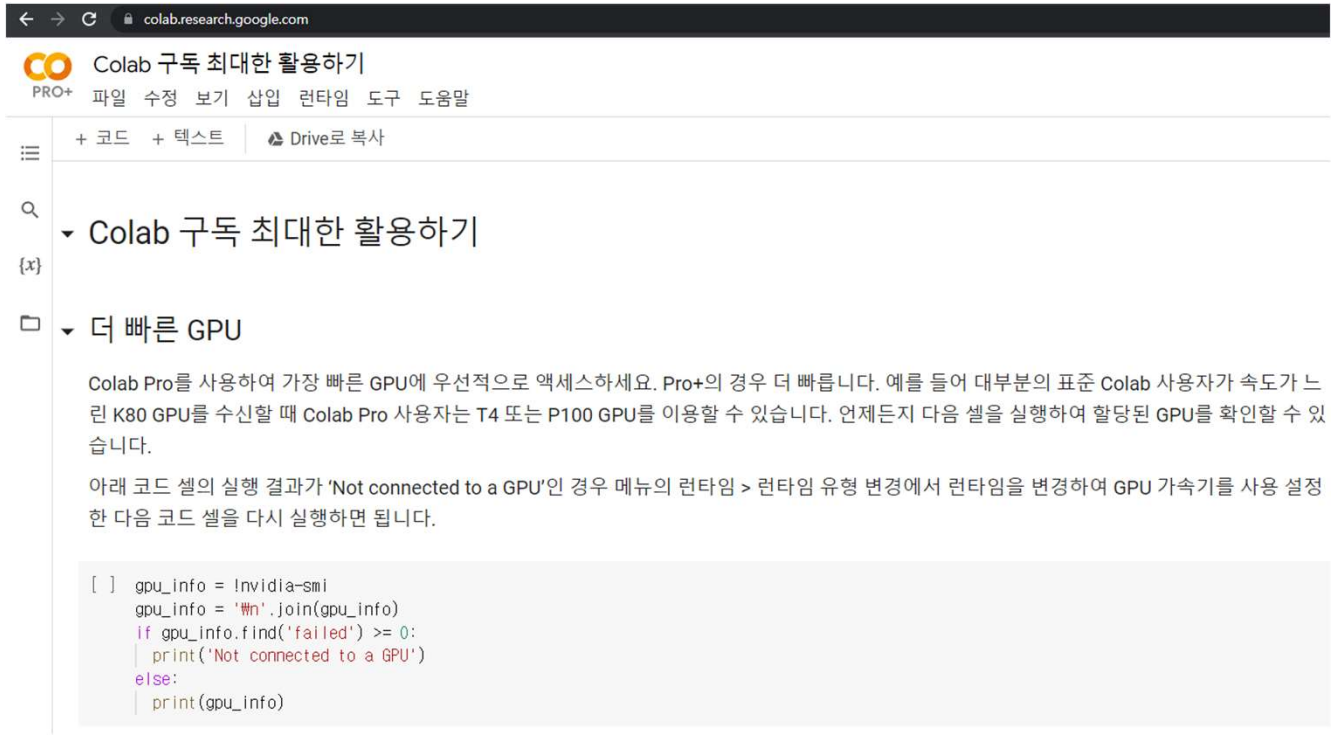
9주차

Monte Carlo Simulation

목차

실험환경 – Google Colaboratory

- Google Colaboratory
- K80 GPU 제공
- Jupyter Notebook 기반의 파이썬 환경 무료 제공



실험환경 – Google Colaboratory

- 파일 -> 업로드에 본 실습을 위해 제공되는 Jupyter Notebook 업로드

실습 파일 실행 예시

Monte Carlo Simulation

- 확정 모형(deterministic model)이 아닌 확률 모형(stochastic model)에서는 분석적인 방법으로 해를 찾을 수 없음
- 몬테카를로 시뮬레이션(Monte Carlo simulation)은 랜덤 샘플링 기법을 반복하여 시뮬레이션을 수행하고, 원하는 수치적 결과를 전체 확률 분포에서 계산해내는 계산 알고리즘의 한 종류
- 몬테카를로 시뮬레이션은 수치 적분, 확률 분포 계산, 확률 기반 최적화 등의 분야에서 사용

- Simulation: 동전 던지기를 모사하기 위하여 [0,1] 범위에서 random value를 추출하여 0.5 미만을 tail로, 0.5 이상을 head로 간주
- Monte Carlo method: 동전을 한 박스 쏟아서 head와 tail의 숫자를 세고, 이로부터 head가 나올 확률을 구함
- Monte Carlo simulation: [0,1] 범위에서 random value를 반복해서 추출, 이로부터 head를 얻을 확률을 구함

Monte Carlo Simulation

- Monte Carlo simulation의 일반적인 패턴은 다음과 같다.

1. 샘플링을 진행할 영역 정의
2. 정의한 영역에 대한 랜덤 샘플링
3. 수집한 샘플들에 대해 결정론적 계산(deterministic computation) 수행
4. 결과를 집계하여 근사치 도출

Monte Carlo Simulation

- 몬테카를로 방법의 간단한 예시로 주사위를 두 번 던져 8의 합이 나오는 경우의 확률을 구해볼 수 있음

1. [1,6] 범위의 난수를 두 개 생성하여, 주사위를 두 번 던지는 시행, 100번을 랜덤하게 실행하는 시뮬레이션을 진행
2. 두 난수의 합이 8이면 hit, 아닐 경우 miss라고 가정하고 hit와 miss의 횟수 집계
3. 전체 시행 횟수에 대한 hit의 비율 계산
4. 수학적으로 계산한 확률과 비교

Monte Carlo Simulation

- 몬테카를로 방법의 또 다른 예시로 원주율을 구하는 것을 들 수 있음

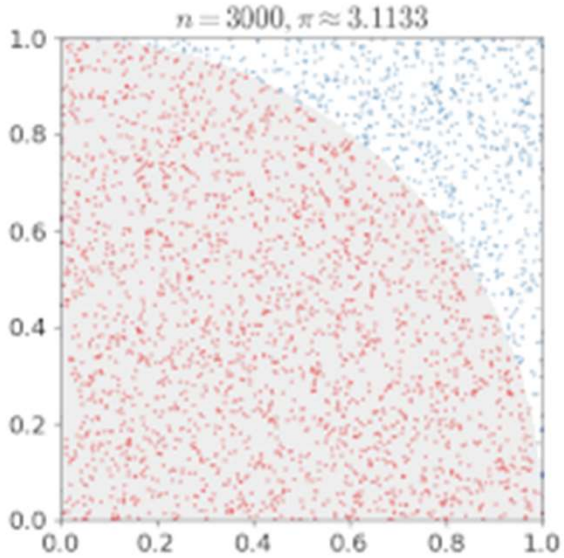


그림 1. 몬테카를로 방법으로 원주율을 계산하는 과정
(https://commons.wikimedia.org/wiki/File:Pi_30K.gif#/media/파일:Pi_30K.gif)

Monte Carlo Simulation

- $x^2 + y^2 = 1$ 으로 표현되는 원을 이용하여 원주율 계산
- 이 원은 $-1 \leq x \leq 1, -1 \leq y \leq 1$ 로 표현되는 넓이가 4인 정사각형 공간 안에 포함됨
- $0 \leq x \leq 1, 0 \leq y \leq 1$ 범위로 제한
- 이 공간 안에서 n개의 난수 순서쌍 (x, y) 추출
- 추출한 점 중 원 내부의 점의 개수 집계
- 원 내부와 전체 점의 개수 비율을 이용하여 π를 구함

$$\frac{\frac{1}{4}\pi r^2}{r^2} = \frac{\square}{\square}$$

Monte carlo simulation을 위한 난수 생성 – Pseudo Random Number Genarator (PRNG)

- 난수 생성 방식은 대표적으로 두 가지를 들 수 있음

1. Linear congruential generator

2. Mersenne Twister

Linear Congruential Generator (LCG, 선형합동법)

- Linear Congruential Generator(LCG)는 다음의 점화식을 따라 난수가 생성되는 알고리즘
- $X_{n+1} = (aX_n + c) \bmod m$
- LCG는 다음과 같은 인자들로 유일하게 결정됨

m : 나눔수, modulus
a : 곱함수, multiplier
c : 더함수, increment
X_0 : 초기값, seed

☞ $0<m$, $0<a<m$, $0 \le c<m$, $0 \le X_0 < m$ (비주얼베이직6의 경우 $m=2^{24}$, $a=43FD43FD_{16}$, $c=C39EC3_{16}$)

- 최대 주기 m을 가지기 위한 필요충분조건

☞ c 와 m이 서로소, a-1이 m의 모든 소인수로 나뉨, m이 4의 배수면 a-1도 4의 배수

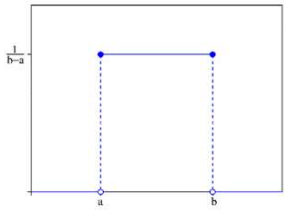
- 작은 메모리로 난수 생성이 가능하여 임베디드 환경에서 적합.
- 또한 상관 관계에 대한 고려가 필요하지 않은 경우에도 LCG 적용 가능
- LCG는 인자들과 마지막 생성된 난수를 알면 그 뒤에 만들어질 모든 난수를 예측 가능하므로 암호학적으로 안전한 난수 생성기라고 볼 수 없음

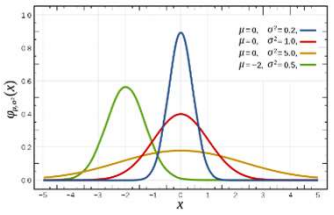
Mersenne Twister

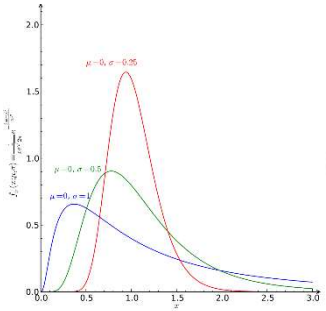
- 1997년에 개발된 유사난수 생성기
- 기존 난수 생성기들의 문제점을 피하면서 매우 질이 좋은 난수를 빠르게 생성할 수 있도록 설계되었음
- 난수의 반복 주기가 메르센 소수($2^n - 1$ 중 소수)인 데에서 유래
- 주기가 $2^{19937} - 1$인 MT19937 사용. 주기가 $2^{19937} - 1$으로 매우 크다.
  - 624개의 integer 사용, 624*32-31=19937
- 다이하드 테스트를 비롯한 다양한 확률적 시험을 통과
- 비트 연산만으로 알고리즘의 구현이 가능하기 때문에 매우 빠름
- 생성기의 상태가 비교적 크다. 최소한 624개의 숫자를 담을 공간이 할당되어야 함. 제한된 용량만을 활용하는 임베디드 환경에서 큰 단점이 될 수 있음
- 암호학적으로 안전하게 설계되어 있지 않음. 난수의 특성을 알고 있을 때 유한한 난수(624개)만으로 현재 생성기의 상태를 알 수 있으며 그 뒤에 나올 난수도 예측 가능함

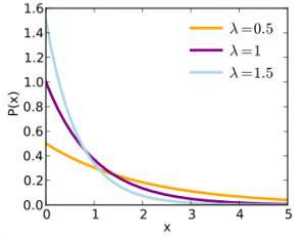
파이썬의 난수 생성 함수 – random

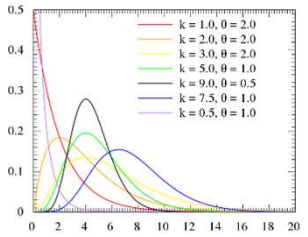
- 파이썬의 난수 생성기는 메르센 트위스터(Mersenne Twister)를 기본으로 탑재하고 있음
- 32비트 정밀도의 float을 생성하며 주기는 $2^{19937} - 1$ 임
- Uniform, Normal, lognormal, negative exponential, gamma, beta 분포를 추출할 수 있는 함수 제공

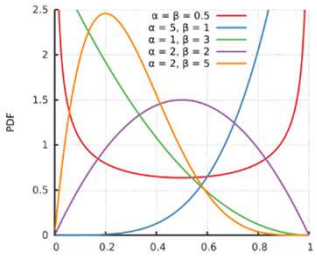












파이썬의 random 함수 예시

```
[1]  import random
```

```
[7]  #random.random() -> 0.0에서 1.0사이의 실수 중에서 난수값 리턴
     print(random.random())
```

```
0.18025638856727455
```

```
[8]  #random.uniform(a, b) -> 괄호 안 두 수 사이의 실수 중에서 난수값을 리턴
     print(random.uniform(10, 30))
```

```
25.974314197832875
```

```
[9]  #random.randint(a, b) -> 괄호 안 두 수 사이의 정수 중에서 난수값을 리턴
     print(random.randint(100,200))
```

```
176
```

```
[10] #random.choice(sample) 함수 -> sample에서 무작위로 하나를 선택하여 리턴
     data = [1, 2, 3, 4, 5, 6, 7]
     print(random.choice(data))
```

```
4
```

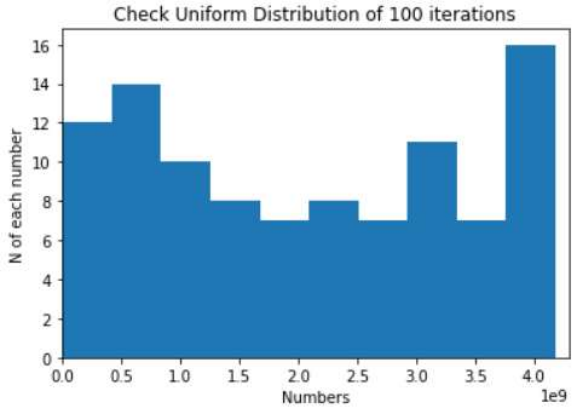
```
     #random.sample(sample, n) -> 입력으로 받은 sample 에서 정한 개수만큼  무작위로 뽑아 리턴함
     data = [1, 2, 3, 4, 'apple', 'banna']
     print(random.sample(data, 3))
     print(random.sample(data, 3))
     print(random.sample(data, 5))
```

```
[1, 4, 'banna']
['banna', 3, 2]
['banna', 4, 1, 2, 'apple']
```

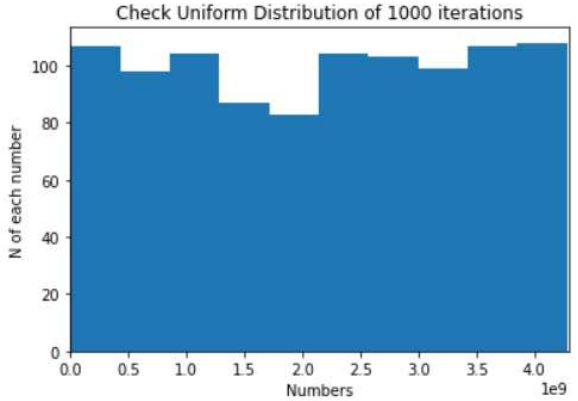
실습 1. 난수 히스토그램 그리기
– Linear Congruential Generator(LCG) 구현

- 실습은 Colaboratory에서 Python을 활용하여 진행
- LCG 구현
- [0,99] 범위에서 난수 생성
- 구현된 LCG를 사용하여 난수 생성하는 과정을 여러 번 반복하고, 각각의 경우에서 생성된 난수에 대한 분포를 확인
- 파이썬 Matplotlib을 사용하여 분포를 히스토그램 형식으로 나타내고, 분포가 uniform distribution의 형태를 보이는지 확인

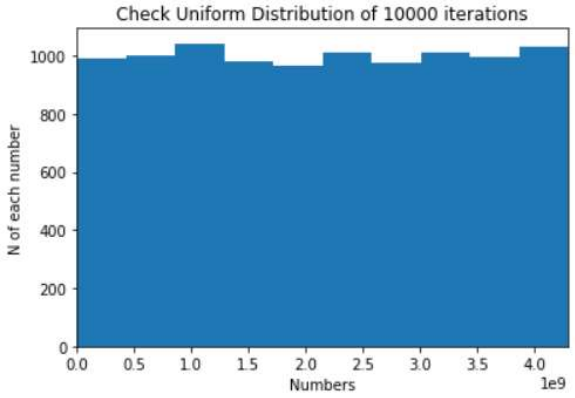
실습 1. 결과 예시



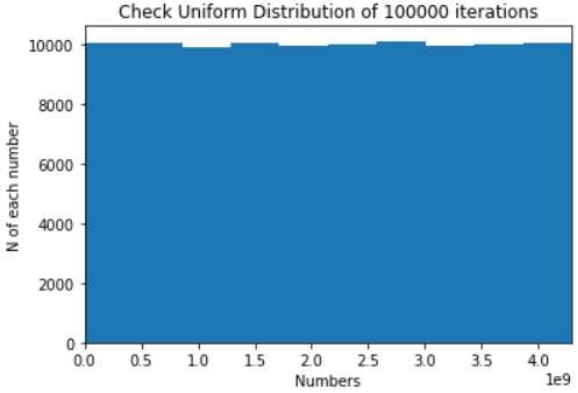
100번 반복



1000번 반복



10000번 반복



100000번 반복

실습 2. LCG와 Mersenne Twister 중에서 어느 것이 Uniform Distribution에 가까운지 확인

- 실습은 Colaboratory에서 Python을 활용하여 진행
- LCG 와 Mersenne Twister(파이썬 기본 random 모듈)을 활용하여 분포를 그려보고, 어느 난수 추출기가 uniform distribution에 가까운지 확인

실습 2. 결과 예시

```
LCG 시행횟수 1000에서의 uniform distribution 오차 : 0.00016992187500000007
LCG 시행횟수 10000에서의 uniform distribution 오차 : 1.1718750000000173e-05
LCG 시행횟수 100000에서의 uniform distribution 오차 : 4.687500000000243e-06
```

```
메르센 트위스터 시행횟수 1000에서의 uniform distribution 오차 : 0.00218
메르센 트위스터 시행횟수 10000에서의 uniform distribution 오차 : 0.000956
메르센 트위스터 시행횟수 100000에서의 uniform distribution 오차 : 0.00024920000000000004
```

실습 결과 예시

실습 3. 주사위 던지기

- 실습은 Colaboratory 환경에서 Python을 활용하여 진행
- random.randint(1,6) 를 이용하여 주사위 눈 값을 추출
- 이때 첫번째 던진 주사위의 눈의 값과 두번째로 던진 주사위의 눈의 값의 합이 8인 경우를 hit으로 간주
- 몬테카를로 시뮬레이션을 사용하여 두 번 주사위를 던질 때 눈의 합이 총 8이 될 확률을 구하고 오차율을 확인

실습 3. 결과 예시

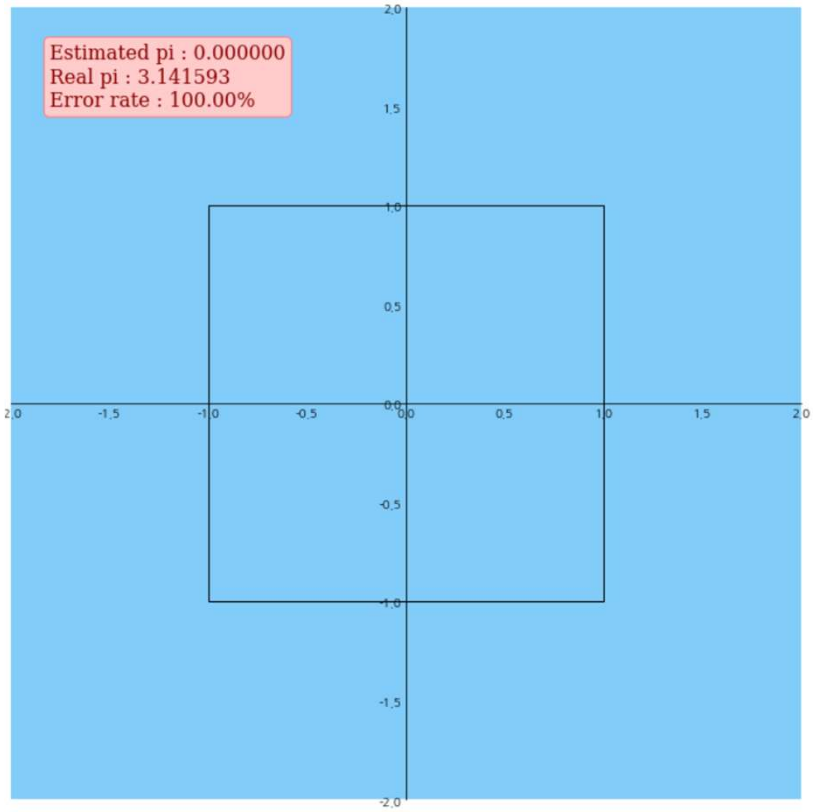
```
================================================================================================================
* 두 주사위의 합이 8인 경우에는 cyan 색의 음영이 들어가 있음 *
try  0 : 4 1 try  1 : 4 6 try  2 : 6 6 try  3 : 3 4 try  4 : 1 2
try  5 : 1 3 try  6 : 6 1 try  7 : 2 1 try  8 : 4 3 try  9 : 3 6
try 10 : 2 3 try 11 : 5 6 try 12 : 6 2 try 13 : 3 4 try 14 : 6 3
try 15 : 1 2 try 16 : 1 3 try 17 : 4 4 try 18 : 3 2 try 19 : 6 5
try 20 : 6 4 try 21 : 4 6 try 22 : 6 1 try 23 : 4 5 try 24 : 5 5
try 25 : 5 4 try 26 : 3 5 try 27 : 6 4 try 28 : 5 1 try 29 : 1 2
try 30 : 6 4 try 31 : 1 4 try 32 : 6 4 try 33 : 6 5 try 34 : 1 5
try 35 : 6 6 try 36 : 1 6 try 37 : 3 5 try 38 : 2 6 try 39 : 1 2
try 40 : 4 4 try 41 : 6 6 try 42 : 4 1 try 43 : 1 1 try 44 : 2 2
try 45 : 6 3 try 46 : 4 5 try 47 : 2 4 try 48 : 4 6 try 49 : 2 2
try 50 : 3 2 try 51 : 5 2 try 52 : 4 3 try 53 : 6 4 try 54 : 2 4
try 55 : 3 4 try 56 : 3 3 try 57 : 5 1 try 58 : 3 3 try 59 : 6 5
try 60 : 6 3 try 61 : 5 3 try 62 : 2 4 try 63 : 2 6 try 64 : 1 4
try 65 : 4 3 try 66 : 4 1 try 67 : 2 3 try 68 : 1 5 try 69 : 1 1
try 70 : 2 6 try 71 : 2 2 try 72 : 6 6 try 73 : 5 3 try 74 : 3 6
try 75 : 1 4 try 76 : 3 6 try 77 : 2 1 try 78 : 5 5 try 79 : 6 4
try 80 : 2 4 try 81 : 3 3 try 82 : 1 4 try 83 : 3 1 try 84 : 3 2
try 85 : 2 3 try 86 : 4 2 try 87 : 5 3 try 88 : 2 1 try 89 : 4 4
try 90 : 2 2 try 91 : 6 4 try 92 : 2 5 try 93 : 4 3 try 94 : 2 5
try 95 : 2 5 try 96 : 4 6 try 97 : 5 3 try 98 : 3 3 try 99 : 6 5

실제 값 : 0.138889
계산된 값 : 0.13
오차율 : 6.400000000000001 %
================================================================================================================
```

실습 결과 예시

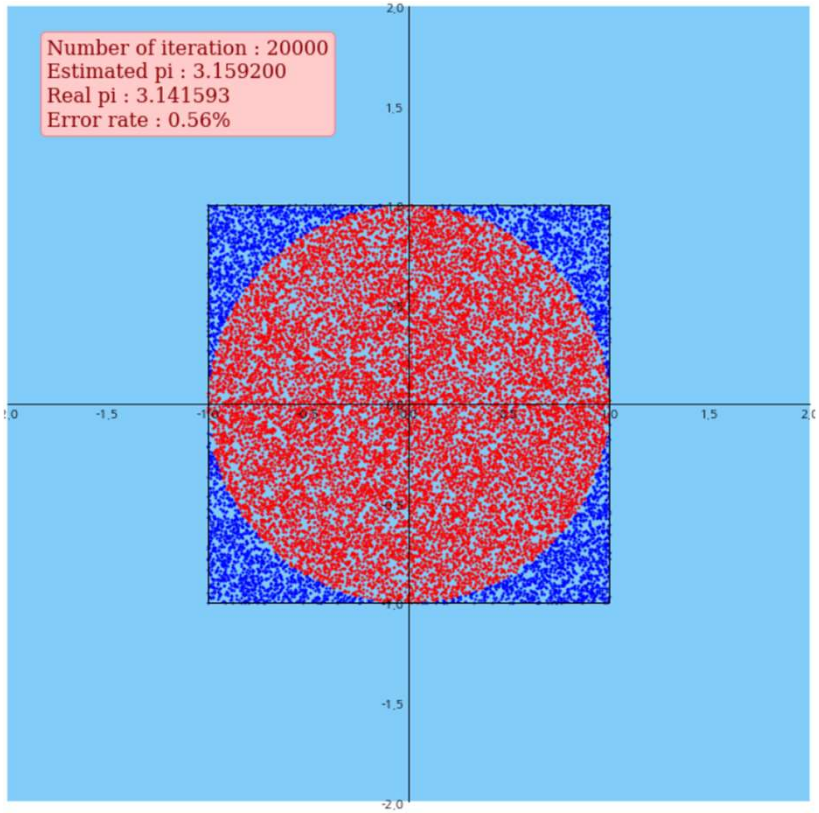
코딩 과제 6. 원주율 구하기

- Python의 random.uniform(-1, 1)을 활용해서 MAX_POINTS개 만큼 (x,y) 좌표의 난수 순서쌍을 생성
- 이때 주어진 변의 길이가 2인 정사각형 안에 접하는 중심이 (0,0)이고 반지름이 1인 원에 난수 순서쌍이 속할 경우와 아닐 경우를 색으로 구별하고 Matplotlib을 활용하여 그림
- 몬테카를로 시뮬레이션을 통하여 π를 구하고 오차율을 확인
- **[Elice]코딩 과제** : π의 오차가 0.1% 미만이 되도록 해볼 것

코딩 과제 6. 시각화 결과 예시



시각화 코드 실행 예시



시각화 코드 결과 예시

보고서 과제 6. LCG, MT 이외의 난수 생성 방식에 관하여 3가지 이상 열거하고 설명하시오. (폰트 10, 한페이지 분량, 다음 실습 전까지)