Programming in Python[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2020/21, II semester

Lecture IX: Random numbers

Exercise

Write a Python program which chooses an integer 1–10 and asks to the user to guess it

- if the number given by the user is not 1–10, it prints "Invalid";
- if the number is the chosen one, it prints "Yes!";
- otherwise "You didn't guess it...".

Evolve the program for asking until the user guess the number correctly giving hints ("higher...", "lower...").

```
from random import randomint

# To get a random integer in the set [1..10]
randint(1, 10)
```


Random numbers

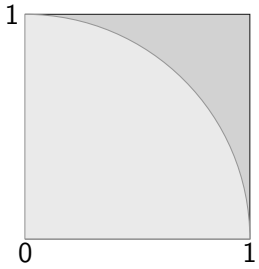
Pseudorandomness: the sequence of numbers is not predictable...

```
from random import randomint

for _ in range(0,10):
   print randomint(1, 100)
```

unless you know the seed.

```
from random import seed, randomint

seed(292)
for _ in range(0,10):
   print randomint(1, 100)
```

Example



- Blue square: 1
- Green area: $\frac{\pi}{4}$

The Monte Carlo method consists of choosing sample experiments at random from a large set and then making deductions on the basis of the probabilities estimated from frequency of occurrences.

Example

```
from random import random

def approx_pi(tries: int) -> float:
    """Return an approximation for pi.

    >>> from math import pi
    >>> from random import seed
    >>> seed(7897)  # Tests should be reproducible
    >>> abs(4*approx_pi(1000) - pi) < 10e-2
    True

    >>> abs(4*approx_pi(100000) - pi) < abs(approx_pi(1000) - pi)
    True
    """
    within_circle = 0
    for i in range (0, tries):
        x = random() # range [0,1)
        y = random()
        if (x**2+y**2)**0.5 < 1:
            within_circle += 1
    return within_circle / tries
```


Example

It's easy to extend to make this work for any function on $[0, 1)$.

```
from random import random
from typing import Callable

def approx_fun(predicate: Callable[[float, float], bool], tries:
↪  int) -> float:
    """Return an approximation for pi.

    >>> from math import pi
    >>> from random import seed
    >>> seed(7897)  # Tests should be reproducible
    >>> within_circle = lambda x, y: x**2 + y**2 < 1
    >>> abs(4*approx_fun(within_circle, 1000) - pi) < 10e-2
    True
    """
    true_cases = 0
    for i in range (0, tries):
        x = random() # range [0,1)
        y = random()
        if predicate(x, y):
            true_cases += 1
    return true_cases / tries
```


Simulations

Random number are useful also for *simulation*: for example, we could simulate evolutionary drift.

```
from random import seed, randint

class DriftSimulation:
    def __init__(self, sim_seed: int = 232943):
        self.population = ['\N{MONKEY}', '\N{TIGER}', '\N{BUTTERFLY}', '\N{LIZARD}',
        ↪  '\N{SNAIL}']
        seed(sim_seed)

    def offspring(self):
        new = self.population[randint(0, len(self.population)-1)]
        self.population[randint(0, len(self.population)-1)] = new

    def simulate(self, generations: int):
        for i in range(0, generations):
            self.offspring()

s = DriftSimulation()
for i in range(100):
    line = str(i)
    for a in s.population:
        line = line + ' ' + a
    print(line)
    if len(set(s.population)) == 1:
        break
    s.simulate(1)
```