Programming in Python[1]

Mattia Monga

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

Academic year 2022/23, I semester

[1] ©️ 2022 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. http://creativecommons.org/licenses/by-sa/4.0/deed.it

Lecture X: Random numbers

Random numbers

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

```
from random import randint

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

```
from random import randint

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

unless you know the seed.

```
from random import seed, randint

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

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...".

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: it should now ask until the user guess the number correctly, giving hints ("higher...", "lower...").

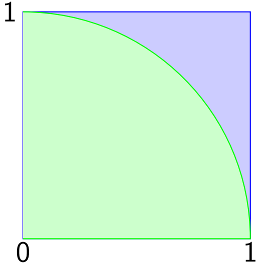
Exercise

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: it should now ask until the user guess the number correctly, giving hints ("higher...", "lower...").
How many tries in the worst case? Can you write a program guessing a number between 1 and `int(1e32)`

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

```python
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
    """
    assert tries > 0
    within_circle = 0
    for i in range (0, tries):
        x = random() # range [0,1)
        y = random()
        if x**2 + y**2 < 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
    """
    assert tries > 0
    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, getstate, setstate

class DriftSimulation:
    def __init__(self, sim_seed: int = 232943) -> None:
        self.population = ['\N{MONKEY}', '\N{TIGER}', '\N{BUTTERFLY}', '\N{LIZARD}',
        ↪  '\N{SNAIL}']
        seed(sim_seed)
        self.r_state = getstate()

    def offspring(self) -> None:
        setstate(self.r_state)
        new = self.population[randint(0, len(self.population)-1)]
        self.population[randint(0, len(self.population)-1)] = new
        self.r_state = getstate()

    def simulate(self, generations: int) -> None:
        for i in range(0, generations):
            self.offspring()

a = DriftSimulation()
b = DriftSimulation()
a.simulate(2)
b.simulate(2)
```