Dit document geeft Python codes voor het simuleren van de afstand tussen twee random gekozen punten voor zowel het eenheidsvierkant als de eenheidscirkel, zie paragraaf 5.2.1 van het boek.

Python code voor het eenheidsvierkant

```
import numpy as np
import numpy.random as rnd
import numpy.linalg as la # voor lineaire algebra berekeningen

def simulatierun(rng):
    '''
    Deze functie simuleert twee random punten P en Q in het
    eenheidsvierkant, en retourneert hun (Euclidische) afstand.
    rng = instantie van random number generator.
    '''
    P = rng.random(2)
    Q = rng.random(2)
    return la.norm(P-Q)

def simulatie(nrun):
    '''
    Deze functie simuleert nrun keer twee random punten in het
    eenheidsvierkant, en houdt hun afstand bij. Vervolgens wordt
    de output statistiek uitgevoerd voor het schatten van
    de verwachte afstand.
    '''
    rng = rnd.default_rng()   # instantie van RNG
    Y = np.zeros(nrun)        # aanmaak output vector Y van afstanden

    for i in range(nrun):
        Y[i] = simulatierun(rng) # i-de afstand

    est = np.mean(Y)         # gemiddelde als schatting van verwachting
    S2 = np.var(Y,ddof=1)    # steekproefvariantie
    SE = np.sqrt(S2 / nrun)  # standaardfout
    print('gemiddelde: %.4f' %est)
    print('95%% BTI   : (%.4f,%.4f)' %(est-1.96*SE, est+1.96*SE))


def main():
    nrun = 1000000
    simulatie(nrun)

if __name__ == '__main__':
    main()
```

Uitvoer

```
simulatiewaarde: 0.5215
95% BTI   : (0.5210,0.5220)
exacte waarde: 0.5214
```

Python code voor de eenheidscirkel

```
import numpy as np
import numpy.random as rnd
import numpy.linalg as la # voor lineaire algebra berekeningen

def randompointinunitcirle(rng):
    '''
    Deze functie simuleert een random punt in de eenheidscirkel via
    de hit-en-mis methode door steeds een random punt in het
    omhullende vierkant te genereren.
    rng = instantie van random number generator.
    '''
    hit = 0
    while hit==0:  # nog geen hit
        P = -1 + 2*rng.random(2) # random punt in omhullend vierkant
        hit = la.norm(P) < 1     # check of punt in cirkel
    return P

def simulatierun(rng):
    '''
    Deze functie simuleert twee random punten P en Q in de
    eenheidscirkel, en retourneert hun (Euclidische) afstand.
    '''
    P = randompointinunitcirle(rng)
    Q = randompointinunitcirle(rng)
    return la.norm(P-Q)

def simulatie(nrun):
    '''
    Deze functie simuleert nrun keer twee random punten in de
    eenheidscirkel, en houdt hun afstand bij. Vervolgens wordt de
    output statistiek uitgevoerd voor het schatten van de
    verwachte afstand.
    '''
    rng = rnd.default_rng()   # instantie van RNG
    Y = np.zeros(nrun)        # aanmaak output vector Y van afstanden

    for i in range(nrun):
        Y[i] = simulatierun(rng) # i-de afstand

    est = np.mean(Y)         # gemiddelde als schatting van verwachting
    S2 = np.var(Y,ddof=1)    # steekproefvariantie
    SE = np.sqrt(S2 / nrun)  # standaardfout
    print('gemiddelde: %.4f' %est)
    print('95%% BTI   : (%.4f,%.4f)' %(est-1.96*SE, est+1.96*SE))


def main():
    nrun = 1000000
```

```
    simulatie(nrun)

if __name__ == '__main__':
    main()
```

Uitvoer

```
simulatiewaarde: 0.9059
95% BTI   : (0.9050,0.9067)
exacte waarde: 0.9054
```