Dit document geeft Python codes voor het simuleren van zowel het verjaardagsprobleem als het bijna-verjaardagsprobleem uit paragraaf 5.2.2 van het boek.

Python code voor het verjaardagsprobleem

```
import numpy as np
import numpy.random as rnd

def randomverjaardagen(k):
    '''
    Doel: genereer een rij met random gekozen verjaardagsdagen.
    input: k = lengte rij;
    output: de rij.
    '''
    return [rnd.randint(365) for i in range(k)]

def simulatierun(verjaardag,k):
    '''
    Doel: controleer of in een rij verjaardagen er minstens
          twee op dezelfde vallen (noem dat succes).
    input: verjaardag = rij verjaardagen;
           k = lengte rij;
    output: 1 als succes;
            0 anders.
    '''
    for i in range(k-1):
        for j in range(i+1,k):
            if verjaardag[i] == verjaardag[j]:
                return 1
    return 0

def simulaties(n,k):
    '''
    Doel: voer herhaaldelijk het experiment uit van
          1) genereer een rij van random gekozen verjaardagsdagen;
          2) controleer of er succes is.
    input: n = aantal herhalingen;
           k = lengte rij;
    output: y = rij van lengte n met op de i-de positie een 1
            als het i-de experiment succes heeft, en 0 anders.
    '''
    y = np.zeros(n)
    for i in range(n):
        verjaardag = randomverjaardagen(k)
        y[i] = simulatierun(verjaardag,k)
    return y
```

```
def statistieken(y,n):
    '''
    Doel: van een rij simulatie-uitkomsten puntschatting
          en 95\% betrouwbaarheidsinterval te geven.
    '''
    p = np.mean(y)         # gemiddelde = schatting
    S2 = np.var(y,ddof=1)  # variantie
    sf = np.sqrt(S2/n)     # standaard fout
    print('geschatte kans:', p)
    print('95 % BTI      :(',p-1.96*sf,',',p+1.96*sf,')')

def main():
    '''
    Doel: geef grootte van de groep en aantal experimenten; daarna
          wordt de simulatie uitgevoerd en de statistieken geprint.
    '''
    k = 23    # aantal personen
    n = 10000 # aantal simulatieruns
    y = simulaties(n,k)
    statistieken(y,n)

if __name__ == '__main__':
    '''
    Doel : voer main uit.
    '''
    main()
```

Dit herhaaldelijk uitvoeren (voor 23 personen) geeft voor het verjaardagsprobleem:

```
geschatte kans: 0.5087
95 % BTI      :( 0.4989 , 0.5185)

geschatte kans: 0.5056
95 % BTI      :( 0.49589 , 0.5154 )

geschatte kans: 0.5073
95 % BTI      :( 0.4975 , 0.5171 )
```

Python code voor het bijna-verjaardagsprobleem

Voor dit probleem hoeft alleen de functie `simulatierun()` aangepast te worden.

```
def simulatierun(verjaardag,k):
    # Doel: controleer of in een rij verjaardagen er minstens
```

```
    #       een keer een verschil van hoogstens 1 is (succes).
    # input: verjaardag = rij verjaardagen;
    #        k = lengte rij;
    # output: 1 als succes;
    #         0 anders.
    for i in range(k-1):
        for j in range(i+1,k):
            dif = abs(verjaardag[i]-verjaardag[j])
            if dif < 1.1 or dif == 364:
                return 1
    return 0
```

Dit weer herhaaldelijk uitvoeren (voor 23 personen) geeft voor het bijna-verjaardagsprobleem:

```
geschatte kans: 0.8886
95 % BTI      :( 0.8824 , 0.8948 )

geschatte kans: 0.8901
95 % BTI      :( 0.8840 , 0.8962 )

geschatte kans: 0.8924
95 % BTI      :( 0.8863 , 0.89847 )
```