20211014_Informationsmanagement_FAT1

October 14, 2021

[1]:
```
# "for" loops (Iteration über Sequenzen -- Listen, Tuples, Dicts, Strings,...)
```

[3]:
```
fruits = ['banana', 'apple', 'cherry']

for x in fruits: # keine Initialisierung der Variable "x" notwendig bei "for"␣
 ↪loops
    print(x)
```

```
banana
apple
cherry
```

[5]:
```
for x in fruits:
    print(x)
    if x == 'apple':
        break # stop befehl
```

```
banana
apple
```

[6]:
```
for x in fruits:
    if x == 'apple':
        break # stop befehl vor dem print
    print(x)
```

```
banana
```

[7]:
```
# range Funktion

for x in range(9): # natürliche Zahlen zw. 0 und die Zahl in Klammer
    print(x)
```

```
0
1
2
3
4
5
```

```
6
7
8
```

[8]:
```
for x in range(2,6): # ich kann den "range" auch mit Anfang und Ende
    print(x)
```

```
2
3
4
5
```

[9]:
```
for x in range(2,30,3): # fange bitte bei 2 an, bis 30, und jede dritte Zahl
    print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

[10]:
```
# "else" mit "for" kann kombiniert werden

for x in range(6):
    print(x)
else:
    print('Endlich sind wir fertig!')
```

```
0
1
2
3
4
5
Endlich sind wir fertig!
```

[11]:
```
# "nested for loop" das sind for loops in for loops

Adj = ['red', 'big', 'tasty']

for x in Adj:
    for y in fruits:
        print(x,y)
```

```
red banana
red apple
red cherry
big banana
big apple
big cherry
tasty banana
tasty apple
tasty cherry
```

[17]:
```python
# nested loop für die Folgen 1 bis 10
# outer loop

for i in range(1, 11):
    # nested loop
    # to iterate from 1 to 10
    for j in range(1, 11):
        # print multiplication
        print(i * j, end='-')
    print()
```

```
1-2-3-4-5-6-7-8-9-10-
2-4-6-8-10-12-14-16-18-20-
3-6-9-12-15-18-21-24-27-30-
4-8-12-16-20-24-28-32-36-40-
5-10-15-20-25-30-35-40-45-50-
6-12-18-24-30-36-42-48-54-60-
7-14-21-28-35-42-49-56-63-70-
8-16-24-32-40-48-56-64-72-80-
9-18-27-36-45-54-63-72-81-90-
10-20-30-40-50-60-70-80-90-100-
```

[18]:
```python
# *
# * *
# * * *
# * * * *
# * * * * *
```

[20]:
```python
rows = 5

for i in range(1, rows + 1):
    # inner loop
    for j in range(1, i + 1):
        print("*", end=" ")
    print('')
```

```
*
* *
```

```
* * *
* * * *
* * * * *
```

[21]:
```
###
```

[22]:
```
# numerisches Python "numpy"
```

[23]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (1.19.5)
```

[24]:
```
import numpy as np # ich lade mein package "numpy" und nenne es "np"
```

[25]:
```
arr = np.array([1,2,3,4,5]) # arrays sind nummerische Listen
print(arr)
```

```
[1 2 3 4 5]
```

[26]:
```
# Arrays werden genutzt um mehrere Werte von einer Variable zu nennen

cars = ['Ford', 'Volvo', 'BMW']
x = cars[0]
print(x)
```

```
Ford
```

[29]:
```
x = len(cars) # länge vom Array
print(x)
```

```
3
```

[30]:
```
# append fügt neue Elemente in dem Array ein

cars.append('Honda')
print(cars)
```

```
['Ford', 'Volvo', 'BMW', 'Honda']
```

[31]:
```
# Dimensionen von Arrays
```

[32]:
```
# 0-D Array ist ein Punkt

arr_0D=np.array(7.9)
print(arr_0D)
```

```
7.9
```

[33]:
```
# 1-D Array ist ein Vektor

array_1D=np.array([1,2,3,4])
print(array_1D)
```

```
[1 2 3 4]
```

[35]:
```
# 2-D Array sind Matrizen

array_2D=np.array([[1,2,3],[3,4,5]])
print(array_2D)
```

```
[[1 2 3]
 [3 4 5]]
```

[36]:
```
# Arrays ab 3 Dimensionen sind Tensoren

arr = np.array([[[1,2,3],
                 [4,5,6]],
                [[7,8,9],
                 [10,11,12]]])
print(arr)
```

```
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

[41]:
```
# Zufallszahlen generieren mit numpy und "random"

from numpy import random

x = random.randint(100) # "randint" generiert ein Zufalls--Integer--zahl zw. 0␣
 ↪und die Zahl in Klammer
print(x)
```

```
62
```

[42]:
```
# mit "rand" kann ich eine Zufallsreele Zahl (float) zw. 0 und 1 generieren

x = random.rand()

print(x)
```

```
0.7961416460210097
```

[43]:
```
# wir können auch den "Shape" (Dimensionen) von dem Zufallszahl eingeben

# Generieren Sie ein 1-D Array mit 5 random Integerzahlen zw. 0 und 100

from numpy import random

x = random.randint(100, size=(5))

print(x)
```

```
[40 84 58 38 11]
```

[44]:
```
# Generieren Sie ein 2-D Array mit 5 Zeilen und 6 Säulen mit random
↪Integerzahlen zw. 0 und 80

from numpy import random

x = random.randint(80, size=(5,6))

print(x)
```

```
[[29 54 37 10 45  9]
 [60 37 16 78 56 64]
 [60 13 35 15 11 63]
 [34 73 37 44 13 66]
 [42 74 77 16 74 16]]
```

[45]:
```
# Generieren Sie ein 3-D Array mit Dimensionen 4,6,7 mit
# Zufallsintegerzahlen zw. 0 und 100

x = random.randint(100, size=(5,6,7))

print(x)
```

```
[[[95 92 24 46 81 83 86]
  [70 18 46 48 27 54 82]
  [57 89 67 79 70 14 85]
  [55 75 30 91 36 43 57]
  [40 71 54 37 36 15 29]
  [14  1  8 90  7 52 65]]

 [[26 33 75 53 26 73 71]
  [20  7 71 22 34 20 59]
  [ 2 73 28 67  7 62  1]
  [35 71 47 81 17 90  5]
  [64 73 27  2 53 91 57]
  [14 86 34 91 43  1 72]]
```

```
[[10 33 92 67 46  1 30]
 [70 92 18 69  7 47  8]
 [31 10 94 54 36 27 87]
 [85 48 23 21 84 91 32]
 [21 95 41 35 30 63 48]
 [11 71 50 56 91 94 28]]

[[56 10 28 52 30 91  1]
 [40 85 70 90 16 48 25]
 [31 59  7 99 65 99 73]
 [93 80 64 46  1 93 81]
 [18 18 51 29 86 84 20]
 [35 50 26 32 44 85 36]]

[[36 77 13 52 47 62  1]
 [81 36 76 68 13 90 23]
 [81 27 78 91 28 40 71]
 [87 16 53 83 79 70 67]
 [11 35  6 32 54 91 59]
 [24 10 85 87  3 49  5]]]
```

[46]:
```
# Generieren Sie ein 4-D Array mit Dimensionen 4,6,7,9 mit
# Zufallsintegerzahlen zw. 0 und 100

x = random.randint(100, size=(5,6,7,9))

print(x)
```

```
[[[[67 88 36 … 13  7 34]
   [90 65 21 … 87 20 10]
   [53 57 15 … 67 32 87]
   …
   [82 55 92 … 94  1  3]
   [86 57 18 … 58 58 51]
   [67 26 47 … 24 68 90]]

  [[82 11 83 … 29 73 71]
   [18 82 79 … 29 76 40]
   [44 13 57 …  9 86 53]
   …
   [ 4 56 21 … 68 88 89]
   [33 93 50 … 42 67 61]
   [84  5 57 …  7 42 47]]

  [[77 61 25 … 22 59 78]
   [14 29 47 … 21 48 57]
   [95 89 11 … 92 17 94]
   …
```

```
  [86 66 70 … 89 82 22]
  [47 62 40 … 27 88 59]
  [88 58 17 … 43 98 24]]

 [[71 46 65 … 30 79 53]
  [66 32 11 … 75  4 26]
  [92 90  7 … 77 25 23]
  …
  [77 83  1 … 30 77 90]
  [52 83 80 … 94 83  6]
  [51 78 56 … 94  0 97]]

 [[21 84 37 … 12 55 71]
  [38 51 74 … 49  3 24]
  [28  1 31 … 15 31 58]
  …
  [67 19 21 … 25 88 23]
  [ 9 50 27 … 72 51 99]
  [60 50 40 … 60  8 88]]

 [[59 41 62 … 44 38 84]
  [83 23 29 … 16 63 58]
  [27  0 24 …  0 73 23]
  …
  [67 29 43 … 45 52 69]
  [65 85 42 … 37 41 73]
  [52 58 95 … 12 53 56]]]


[[[54 65 88 … 86 46 80]
  [35 94 99 … 20 84  6]
  [52 74 46 … 69 56 53]
  …
  [20 37 89 … 33 56 48]
  [26 89 72 … 13 98  8]
  [35 68 16 … 24 28 94]]

 [[10 16 22 … 87 69 16]
  [21  4 62 … 78 20 10]
  [87 28 43 … 12 13 93]
  …
  [66 84 72 … 67 58 58]
  [45 21  4 … 79 95 99]
  [61 56 73 … 22 91 90]]

 [[39 31 97 … 38 31 23]
  [49 75  9 … 56  1 72]
  [54 99  1 … 67 87 39]
```

```
  …
   [22 83 21 … 33 71 87]
   [12 73 76 … 81 29 24]
   [87 90 38 … 98 71 41]]

  [[68 43 42 … 20 14 37]
   [46  6 29 … 49 61 95]
   [85  2 29 … 38 29 74]
  …
   [98  8 58 … 27 45 37]
   [ 1 90 70 … 18 39 41]
   [79 56 38 … 80 77 37]]

  [[94 72 99 … 28 33 18]
   [12 17 31 … 96 40 49]
   [29 97 43 … 29 98  3]
  …
   [43 54 83 … 41 12 61]
   [54 82 76 … 68  3 34]
   [60  1 68 … 68 74 43]]

  [[82 24 10 … 17 69 24]
   [ 1 47  6 … 16 84 77]
   [33 71  1 … 99 16 61]
  …
   [67 13 10 …  9 67 67]
   [65 34 70 … 80 75 90]
   [95 88 93 … 91 28 89]]]


[[[55 24 62 … 13 38 30]
  [53 26 36 … 26 72  0]
  [11 77 28 … 66 73 75]
  …
  [96 28 53 …  1 24  8]
  [94 23 44 … 38 71 29]
  [56 85 91 …  1 62 92]]

  [[96 67 72 …  2 37 98]
   [ 8  8 99 … 51 88 96]
   [97 84 34 … 50 37 62]
  …
   [11 41  4 … 75 73 20]
   [11 40 60 … 53 92 64]
   [74 13 91 … 55 24 91]]

  [[68 83 87 … 23  7 41]
   [86 67 33 …  3 97 77]
```

```
  [87 41 34 … 43 60 36]
  …
  [53 78 31 … 64  0 22]
  [42 41 13 … 36  2 51]
  [90  1 67 … 60 60 22]]

 [[89 59 86 … 38 22 25]
  [85 62 82 … 82 34 15]
  [88 52 55 … 44 17 30]
  …
  [69 45 19 … 60 42 11]
  [73  2 62 … 37 38 92]
  [41 59 56 …  9  2 98]]

 [[44 18 36 … 85 22 78]
  [64 82  5 … 95 63  0]
  [27  0  8 …  4 66 31]
  …
  [71 48 69 … 41 91 12]
  [67 62 15 … 25 30 19]
  [30 92 29 … 28 87 31]]

 [[81 28 53 … 27 52 40]
  [72 87 97 … 96 53 84]
  [97 57 23 … 44 25 37]
  …
  [16 37 11 … 59 35 90]
  [69 47  0 … 30 60 71]
  [ 8 30 86 … 93 36 29]]]


[[[16 17 51 … 88 38 56]
  [22 55 82 … 18 15  4]
  [49 35 71 … 96 91 75]
  …
  [36 25 73 … 68  9 94]
  [21 20 37 … 94 31 29]
  [47 76 66 … 92 76 70]]

 [[63 25 16 …  4 72 10]
  [30 37 11 … 12 91 39]
  [64  2  9 … 50 52 52]
  …
  [51 49 19 … 20 48 94]
  [50 76  2 … 53 78  1]
  [92 33 82 … 15 80 17]]

 [[ 8 24  9 … 13 75 87]
```

```
  [26 90 28 … 91  7 63]
  [52 65 93 … 97 58  4]
  …
  [26 74 63 … 39 58 92]
  [30  4 45 … 96 53 66]
  [ 5 11 93 …  5 17 47]]

 [[15 71 51 … 33 36 26]
  [85 79  0 … 33 78 67]
  [44 82 75 … 26 46 63]
  …
  [ 0 17 69 … 76 10 79]
  [41 19 72 … 41  2 44]
  [97 81 23 … 33 28 13]]

 [[89 52 36 … 10 76 48]
  [45 39 96 … 33 48 30]
  [62 57 24 … 57 89 54]
  …
  [56 84 59 … 91 19 86]
  [40 80 21 … 62 41 12]
  [25 47  1 … 67  9 69]]

 [[88 16 24 … 44 62 99]
  [ 1 54 64 … 73 78 68]
  [24 12 54 … 48 11 79]
  …
  [58 23 93 … 83  0  3]
  [33 46 40 … 46 17 15]
  [ 4 89  7 … 90 94  7]]]


[[[44 66 59 … 59 73  8]
  [72 92 84 … 32 64 49]
  [22 13 81 … 37 23 30]
  …
  [59 81 92 … 57 84 13]
  [35 74 55 … 83 21  8]
  [36 25 24 … 87 72 88]]

 [[88  1 20 … 46 14 12]
  [84 47 17 … 49 55 80]
  [51 76 87 … 82 45 65]
  …
  [11  6 24 … 59 96 88]
  [47 95 66 … 71 72 17]
  [27  1 50 …  7 39 14]]
```

```
[[54 32  8 …  3 54 97]
 [70 89 35 …  1 25 67]
 [75 29  2 …  5 71 40]
 …
 [16 63 44 … 22 46 24]
 [27  1 46 … 56  2 12]
 [99 58 77 … 28 24 53]]

[[11 87 54 …  3 22 33]
 [27 69 19 … 55 12 27]
 [20 58 81 … 28 57 50]
 …
 [15 85 79 … 20 77 10]
 [58 17 96 … 25 81 98]
 [46  8 65 … 17 87 66]]

[[95 66 60 …  9 78 78]
 [29 95 51 … 18 81 22]
 [50 92 29 … 23  9 95]
 …
 [41 27 26 … 83 28 66]
 [65 82 96 … 16 72 48]
 [27 33 38 …  4  1 49]]

[[42 20 56 … 65 94 55]
 [62 68 96 … 20 84 17]
 [72 15 68 …  5 64 54]
 …
 [19 63 63 …  5 30 98]
 [26 53 95 … 39  4 50]
 [26 34 88 … 37 69 34]]]]
```

[49]:
```
# "choice" erlaubt die generation von Zufallszahlen basiert auf einem
 ↪bestimmten Datensatz

x = random.choice([12, 24, 35, 56, 67])

print(x)
```

```
56
```

[50]:
```
# Generieren Sie einen 1--D Array mit 100 Elementen
# Alle Elemente müssen 3,5,7, oder 9 sein.
# Die Wahrscheinlichkeit von 3 ist 0.1
# Die Wahrscheinlichkeit von 5 ist 0.3
# Die Wahrscheinlichkeit von 7 ist 0.6
# Die Wahrscheinlichkeit von 9 ist 0
```

```
# "choice" und parameter "p"= Wahrscheinlichkeit

x = random.choice([3,5,7,9],
                  p=[0.1, 0.3, 0.6, 0],
                  size=(100))

print(x)
```

```
[7 7 7 7 7 7 7 7 7 7 5 5 7 3 5 7 7 5 7 7 3 5 7 5 7 7 7 7 5 7 7 7 7 5 5 5
 7 5 7 5 7 7 7 7 5 5 7 3 7 5 7 7 7 7 5 7 7 7 7 7 7 7 7 5 7 7 5 7 5 7 7 7 7
 7 7 3 7 7 7 7 7 7 7 5 5 7 5 3 7 7 7 5 5 7 7 5 7 7 3]
```

[54]:
```
# "random.shuffle" Funktion bittet uns die Möglichkeit Array Elementen␣
 ↪durcheinander zu würfeln

arr = np.array([1,2,3,4,5])

random.shuffle(arr)

print(arr)
```

```
[2 5 3 1 4]
```

[63]:
```
# Generation einer Normalen Distribution

# Mittelwert "loc"
# Standard Abweichung "scale"

from numpy import random

x = random.normal(loc=1, scale=2, size=(2,100))

print(x)
```

```
[[ 2.47207194 -0.98786173 -3.81258445  3.2186694   2.31773681  2.44966699
   1.27356593  2.65050775  1.62321442  0.70663874  0.27782325  2.97543071
   0.6047513   4.09330527  0.49585611 -0.5498974   3.31511489  3.59209078
   3.69739649  2.07867028  1.23412792  1.08075382  0.60033097  2.80759715
  -1.30632599  0.12426724 -0.63863464 -0.774001    3.059473    1.54092368
   1.00516743  2.87990615  0.71220142 -0.24556279 -0.15484963 -0.30261409
   4.06596809  2.45109958  0.18939105  1.14097151  5.42547903 -1.3533851
   0.51844097  0.85714211  0.74401179  2.4359261  -0.90995758  7.27687955
   2.32728576  1.03883139  0.46546739 -1.03504888 -0.75069999  1.11948939
  -1.77211466  0.16377773  1.39588218 -2.59988697 -2.09501901  0.8597387
  -1.23261328  1.17056386 -1.34155325  2.18339343 -0.09459875 -0.3957591
  -0.64645052 -1.35085395  0.58420041  4.62016145  0.63451045 -0.68451757
   0.83569995  2.40893035 -0.55123386  0.94892281  0.31926639 -0.63594268
```

```
 -0.18299917  4.09637377  0.07600243  1.01783277  3.26705289 -0.32024757
  3.66013639  0.45589269  3.6016015   1.73170175  1.81380108  3.58666871
  0.2435078  -0.10779648  0.55299612  0.91817112  3.64164142  0.5576051
  1.61665372  1.73931029  1.61182666  3.76594186]
[-1.6165759   2.42479304  0.17119193  0.73880508  4.7608194   4.59128534
  3.15933893 -0.77106256  2.31730904  0.32106384  1.27422028  3.93542804
  0.6310644   3.36669809 -1.95602401  0.59512331  0.28175351  1.69582425
 -0.39000405  3.92332265  0.72844713  1.10038607  3.25289021 -2.10642334
  1.72269464  2.96706286  0.31445748  0.0935014   1.44782926 -1.26320356
 -1.71942569 -1.91480944 -1.79485039  0.6290107   0.73784478  0.08382941
  4.12649712  1.55004703 -0.72491738  1.85815899  4.16935213  0.23461398
  0.55644289 -1.08538356  3.61126726  3.415501   -1.10188635  4.31199515
  2.03550056  2.50991294  0.42814706 -1.58629561  3.51062419  3.58144944
  0.74930662 -0.57935062 -0.07267195  3.50347128 -1.32038638  0.1462143
  2.85952613 -0.4733034  -0.85535289  0.64249903 -0.5174839  -0.57358428
  0.70097644 -1.47690957  2.0141549   0.75668853 -1.89855044  1.19048861
  1.73211811  0.77638183 -0.08562325 -0.38898754  2.04539978  1.95026384
 -3.5600392  -0.0822582  -0.07841709  1.45806966  2.84345362  2.06035045
  0.40147942  0.77962689  3.39724454  1.64180526  2.97910894 -0.35897432
  1.73405681  0.76266104  1.26577436  0.24855307 -0.10399493  1.13610823
 -3.47856888  1.83071066 -0.01884854  1.34027757]]
```

[ ]: