1 Python

1.1 Vergleiche

```python
# Instead of
if A == True:
    print("Die Aussage A ist wahr.")

if B == False:
    print("Die Aussage A ist falsch.")

# use
if A:
    print("Die Aussage A ist wahr.")

if not B:
    print("Die Aussage B ist falsch.")
```

1.2 Lists

```python
pokemons = {1: "Bisasam", 2: "Bisaknosp", 3: "Bisaflor"}
```

```python
# Instead of
if len(pokemons) != 0:
    print(pokemons)

# use
if pokemons:
    print(pokemons)
```

```python
# Instead of
for nr in pokemons:
    pokemon = pokemons[nr]
    print(f"{nr}:{pokemon}")

# use
for nr, pokemon in pokemons.items():
    print(f"{nr}:{pokemon}")
```

```python
# Instead of
for i in range(len(pokemons)):
    print(pokemons[i])

# use
for pokemon in pokemons:
    print(pokemons)
```

```python
# Instead of
nr = 1
for pokemon in pokemons:
    print(f"{nr}:{pokemon}")
    nr += 1

# use
nr = 1
```

```
for nr, pokemon in enumerate(pokemons):
    print(f"{nr+1}:{pokemon}")
```

1.3 Matrices

```
# Instead of
v = (1, 0, -1)
x = v[0]
y = v[1]
z = v[2]

print(f"x={x}, y={y}, z={z}")

# use
v = (1, 0, -1)
x, y, z = v

print(f"x={x}, y={y}, z={z}")
```

1.4 Context-Manager

```
# Instead of
f = open("file.txt", "w")
f.write("Hallo Welt")
f.close() #Closing manually is mandatory!

# use
with open("file.txt", "w") as f:
    f.write("Hallo Welt")
```

```
# Instead of
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((host, port))
finally:
    s.close()

# use
socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((host, port))
```

1.5 Handling NA or NaN in data frames

```
with open("example.csv", "w") as outfile:
    outfile.write(
        "first_name,last_name,age,country\n"
        "John,Smith,10,USA\n"
        "Billy,Joe,,USA\n"
    )

import csv
with open("example.csv") as infile:
```

```
    reader = csv.reader(infile)
    for line in reader:
        print(line)
```

```
import pandas as pd

#create NaN
df_import = pd.read_csv('example.csv')
print(df_import)

#create NaN
df_import = pd.read_csv('example.csv')
df_import.fillna(pd.NA)
print(df_import)

#create NaN
df_import = pd.read_csv('example.csv')
df_import.fillna('')
print(df_import)

#creates <NA>
df_import = pd.read_csv('example.csv',
                        dtype = {
                            'first_name': str,
                            'last_name': str,
                            'age': pd.Int64Dtype(),
                            'country': str}
                        )
print(df_import)
```

1.6 Import dataframe from document

```
import io

string_data="""first_name,last_name,age,country
John,Smith,10,USA
Billy,Joe,,USA
"""

#creates <NA>
df_import = pd.read_csv(io.StringIO(string_data),
                  sep=",",
                  dtype = {
                      'first_name': str,
                      'last_name': str,
                      'age': pd.Int64Dtype(),
                      'country': str}
                      )
print(df_import)
```

1.7 Different ways to print variables

```
print(name + ' ' + family)              # Concatenation
print(name, family)                     # print() takes multiple arguments
print(name, family, sep=' ')            # Can specify how to separate args
print(' '.join([name, familiy]))        # str.join works for lists of strs
print('%s %s' % (name, family))         # Similar to str.format()
print('{} {}'.format(name, family))     # str.format() gives you more control over the format of the var
print('{0} {1}'.format(name, family))   # Can specify which args to use where
print(f'{name} {family}')               # f-strings let you interpolate variables directly
```