Python (Listen und Tupel) **Prüfungsstoff**

1. Einfache und verschachtelte Listen definieren:

```
a =[9,3,4,2,7,5]
b =[[5,3,2], [1,7], [9,4,8]]
```

2. Zugriff auf Listenelemente (auch bei verschachtelten Listen):

```
a =[9,3,4,2,7,5]
b =[[5,3,2], [1,7], [9,4,8]]
print(a[4])    # => 7
print(b[2][0]) # => 9
```

3. Zugriff auf Listenelemente mit negativen Indizes:

```
a =[9,3,4,2,7,5]
print(a[-2]) # => 7
```

4. Die Länge einer Liste bestimmen:

```
a =[9,3,4,2,7,5]
b =[[5,3,2], [1,7], [9,4,8]]
print(len(a)) # => 6
print(len(b)) # => 3 # 'innere' Elemente werden nicht gezählt
```

5. Einzelne Elemente einer Liste verändern:

```
c = [8,1,4,9]
c[2] = 7
print(c)     # => [8,1,7,9]
```

6. Elemente am Ende einer Liste hinzufügen:

```
d = [5,2,3]
d.append(1)
print(d)   # =>  [5,2,3,1]
```

7. Ein Element vom Ende der Liste löschen:

```
e = [7,9,5,8,2]
x = e.pop()
print(e) # =>  [7,9,5,8]
print(x) # =>  2
```

8. Ein Element von der i-ten Position löschen:

```
f = [7,9,5,8,2]
x = f.pop(2)
print(f) # => [7,9,8,2]
print(x) # => 5
```

9. Listen verketten:

```
print([9,3,2] + [4,1]) # => [9,3,2,4,1]
```

10. Listen multiplizieren:

```
print(3 * [7,2]) # => [7,2,7,2,7,2]
```

11. Zugriff auf Teillisten („slices“):

```
g = [7,9,5,8,2,1,4,3]
print(g[2:5]) # => [5,8,2]
print(g[:3])  # => [7,9,5]
print(g[6:])  # => [4,3]
```

12. Kopieren von Listen:

```
h = [6,1,2,7,9]
i = h     # (Kopie der Referenz auf h)
j = h[:]  # (echte Kopie)
h[3]=0    # (verändern von h)
print(i)  # => [6,1,2,0,9]
print(j)  # => [6,1,2,7,9]
```

13. Die `reverse`-Methode:

```
k = [6,1,2,7,9]
k.reverse()
print(k) # => [9,7,2,1,6]
```

14. Die `sort`-Methode:

```
m = [6,1,2,7,9]
m.sort()
print(m) # => [1,2,6,7,9]
```

15. Die `sum`-Funktion:

```
n = [6,4,2,8]
x = sum(n)
print(x) # => 20
```

16. Mehrfachzuweisungen:

```
[y, x, z] = [5, 3, 1]
print(x) # => 3
```

17. Listen elementweise durchlaufen:

```
p = [9, 2, 4, 1, 8]
s = 0
for element in p:
    s += element

print(s) # => 24
```

18. Listen indexgesteuert durchlaufen:

```
p = [9, 2, 4, 1, 8]
s = 0
for i in range(0, len(p)):
    s += p[i]

print(s) # => 24
```

19. Definition von Tupeln:

```
p = (9,3,4,2,7,5)
```

20. Zugriff auf Tupel-Elemente:

```
p = (9,3,4,2,7,5)
print(p[1]) # => 3
```

21. Operationen und Funktionen für Tupel:

    Bei Tupeln sind alle Listenoperationen erlaubt, die das Original-Tupel nicht verändern.

```
p = (9,3,4,2,7,5)
len(p)  # => 6
p[3]=0  # => Fehler!
p[1:4]  # => (3,4,2)
        # usw.
```