

# Cyber Security

# Grundlagen der Programmierung

# Input und Output



# Ausgabe von Text

Der letzte Step im EVA-Prinzip ist die Ausgabe.

- Mit dem **print** Befehl werden Daten auf der sogenannten Standardausgabe ausgegeben.
  - **print("Das ist ein Test.")**
  - **print('Das ist ein Test.')**
- Der Befehl **print()** erzeugt lediglich eine leere Zeile



# Ausgabe von Text

- Die Benutzung von einfachen Anführungszeichen hat den Vorteil im Text doppelte Anführungszeichen in der Ausgabe nutzen zu können.
  - **`print('Das war ein richtiger „WOW“-Effekt!')`**

```
Das war ein richtiger „WOW“-Effekt!
```



# Ausgabe von Zahlen

- Dieses Beispielprogramm erzeugt folgende Ausgabe:

```
1 a=100
2 b=3.1415
3 c=200
4 print(a)
5 print(b)
6 print(a,c,b)
7 print(a+b)
```

```
100
3.1415
100 200 3.1415
103.1415
```



# Ausgabe von Zahlen

## Erklärung:

- In den Zeilen 1 bis 3 werden den Variablen a, b und c die Werte 100, 3.1415 und 200 zugewiesen.
- Zeile 4 gibt den Wert der Variable a aus, also 100
- Zeile 5 gibt den Wert der Variable b aus, also 3.1415
- Zeile 6 gibt die Werte der, im Befehl durch Komma getrennten, Variablen aus: 100, 200, 3.1415
- In Zeile 7 findet innerhalb des Befehls eine Berechnung statt:  $a+b$  (ergibt 103.1415)



# Ausgabe von Zahlen

**Der print-Befehl kann noch mehr:**

- Syntax: **`print(*args, sep=' ', end='\n', file=None, flush=False)`**

**Erklärung der Parameter:**

- **`*args`** übergibt die auszugebenden Daten an den print Befehl
- **`sep=' '`** definiert den Separator, durch den Werte bei Verwendung eines Kommas im print Befehl getrennt werden. (Beispiel: `print(a,b,c)`)





# Ausgabe von Zahlen

## Erklärung der Parameter:

- **end='\\n'** definiert, wie das Ende einer Zeile aussieht. Im Standardfall wird eine neue Zeile begonnen. ('\\n')
- **file=None** Dieser Parameter ist optional. Der Standardwert ist 'sys.stdout', somit erfolgt die Ausgabe auf dem Bildschirm. (Die Standardausgabe kann auch vom Bildschirm auf andere Geräte umgeleitet werden.)
- **'flush'** bezieht sich auf den file-Parameter und kann die Werte **True** oder **False** annehmen. Ist flush auf True gesetzt, erfolgt eine sofortige Ausgabe des Streams.



# Ausgabe von Zahlen

## Beispielprogramm für den 'sep'-Parameter

```
a=10
b=20
c=30

print(a,b,c)
print(a,b,c, sep=' ')
print(a,b,c, sep='#')
print(a,b,c, sep='|')
```

```
10 20 30
10 20 30
10#20#30
10|20|30
```



# Ausgabe von Zahlen

## Beispielprogramm für den 'end'-Parameter

```
a=10
b=20
c=30

print(a,b,c)
print(a,b,c, end='\n')
print(a,b,c, end=' ')
print(a,b,c)
```

```
10 20 30
10 20 30
10 20 3010 20 30
```



# Eingabe von Daten

Die gebräuchlichste Art Daten an ein Programm zu übergeben, geschieht über die Tastatur. Der Standard-Befehl dazu lautet **input**

```
Python Console>>> help(input)
Help on built-in function input in module builtins:

input(prompt=' ', /)
    Read a string from standard input. The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
    On *nix systems, readline is used if available.
```

oder etwas einfacher:

```
variable = input('Erklärender Text ... was eingegeben werden soll. Ihre Eingabe: ')
```



# Eingabe von Daten

Durch den Befehl erscheint der Text zwischen den Hochkommas auf der Std.-Ausgabe und der Interpreter wartet auf die Eingabe durch den Benutzer, welche mit der <Enter>-Taste abzuschließen ist.

Die Benutzereingabe wird in der Variablen **variable** gespeichert.



# Eingabe von Daten

## Beispiel

```
variable = input('Ihre Eingabe: ')
print('Sie haben "' + variable + '" eingegeben.')
```

## Erklärung:

- In der ersten Zeile wird eine beliebige Eingabe vom Benutzer entgegengenommen. Die Übergabe des Wertes an die Variable erfolgt stets als String!
- In der zweiten Zeile wird die Eingabe lediglich in Anführungszeichen ausgegeben.
- **Neu:** Die Ausgabe im print-Statement kann durch + verknüpft werden.



# Eingabe von Daten

**Weiteres Beispiel (eine Variable kann mehrfach genutzt werden):**

```
a = input("Ihre Eingabe: ")  
print(a)  
  
a = input("Weitere Eingabe: ")  
print(a)
```

**Erklärung:**

- Der Variablen **a** wird durch den input-Befehl (In Zeile 1) ein Wert zugewiesen und mittels **print** (In Zeile 2) ausgegeben.
- In Zeile 4 wird der Variablen **a** ein neuer Wert zugewiesen und im Folgenden ausgegeben.
- Die Deklaration einer neuen Variablen ist also nicht erforderlich, wenn der Inhalt nicht weiter benötigt wird.



# Eingabe von Daten

## input und der Datentyp ...

- Die Eingabe durch den ‚input‘-Befehl wird stets als Datentyp ‚String‘ übergeben:

```
a = input('Ihre Eingabe: ')\nprint(type(a))    # Der Befehl input übergibt immer (!) einen String
```

```
Ihre Eingabe: 3\n<class 'str'>
```

- In dem Beispiel wurde die Zahl drei eingegeben und der Variablentyp von **a** mit dem Schlüsselwort **type** innerhalb von **print** ausgegeben.





# Eingabe von Daten

## Umwandlung von Datentypen: casting

```
a = '5'  
print(type(a))  
  
b = int(a)  
print(type(b))
```

- Der Variablen **a** wird der Wert **5** zugewiesen. Die Anführungszeichen deklarieren die Variable als String. Siehe Kontrolle durch den print Befehl.
- Der Variablen **b** wird der Wert von **a** zugewiesen. Durch die Verwendung von **int( )** wird eine sogenannte Typumwandlung nach Integer vorgenommen und dadurch der Wert 5 nicht als String übergeben.



# Eingabe von Daten

- Casten während einer Eingabe, um direkt zum Zieldatentyp zu gelangen

```
1 print("Dieses Programm addiert zwei Ganzzahlen")
2 print()
3 a = int(input('Zahl 1: '))
4 b = int(input('Zahl 2: '))
5 c = a + b
6 print("Das Ergebnis der Addition ist: ", end='')
7 print(c)
```

- In Zeile 3 und 4 wird mittels Input die Benutzereingabe entgegengenommen.
- Die Eingabe wird mittels **int(...)** direkt in den Datentyp Integer umgewandelt und wird in Zeile 5 für eine Berechnung genutzt.



# Eingabe von Daten

Die Eingabe muss nicht zwingend über die Tastatur erfolgen.

## **Datei**

- Befindet sich auf der Festplatte
- Wird zum Schreiben oder lesen geöffnet

## **Stream**

- Nach dem Öffnen der Datei besteht die Möglichkeit einen Datenstrom aus der Datei zu lesen oder dort hinzusenden. (Dies ist i.d.R. lediglich eine Zuweisung)

## **Variable**

- Auf einer der Seiten befindet sich eine Variable von der aus zugewiesen wird bzw. welche den Datenstrom empfängt.



# ASCII-Tabelle

## **ASCII ist die Abkürzung für:**

- American Standard Code for Information Interchange

## **Deutsche Entsprechung:**

- Amerikanischer Standard-Code für den Informationsaustausch
- Der ASCII-Code bildet die Grundlage für eine standardisierte Kommunikation
- Einführung im Jahr 1963.
- Umfang 128 sichtbare und nicht sichtbare Zeichen (die Steuerzeichen)



# ASCII-Tabelle

| <div> <div> <div>b<sub>7</sub>b<sub>6</sub>b<sub>5</sub></div> <div>b<sub>4</sub>b<sub>3</sub>b<sub>2</sub>b<sub>1</sub></div> </div> <div> <div>Column →</div> <div>Row ↓</div> </div> </div> |   |   |   |    | 0   | 1   | 2  | 3 | 4 | 5 | 6 | 7   |
|--|---|---|---|----|-----|-----|----|---|---|---|---|-----|
| 0  | 0 | 0 | 0 | 0  | NUL | DLE | SP | 0 | @ | P | ` | p   |
| 0  | 0 | 0 | 1 | 1  | SOH | DC1 | !  | 1 | A | Q | a | q   |
| 0  | 0 | 1 | 0 | 2  | STX | DC2 | "  | 2 | B | R | b | r   |
| 0  | 0 | 1 | 1 | 3  | ETX | DC3 | #  | 3 | C | S | c | s   |
| 0  | 1 | 0 | 0 | 4  | EOT | DC4 | \$ | 4 | D | T | d | t   |
| 0  | 1 | 0 | 1 | 5  | ENQ | NAK | %  | 5 | E | U | e | u   |
| 0  | 1 | 1 | 0 | 6  | ACK | SYN | &  | 6 | F | V | f | v   |
| 0  | 1 | 1 | 1 | 7  | BEL | ETB | '  | 7 | G | W | g | w   |
| 1  | 0 | 0 | 0 | 8  | BS  | CAN | (  | 8 | H | X | h | x   |
| 1  | 0 | 0 | 1 | 9  | HT  | EM  | )  | 9 | I | Y | i | y   |
| 1  | 0 | 1 | 0 | 10 | LF  | SUB | *  | : | J | Z | j | z   |
| 1  | 0 | 1 | 1 | 11 | VT  | ESC | +  | ; | K | [ | k | {   |
| 1  | 1 | 0 | 0 | 12 | FF  | FS  | ,  | < | L | \ | l |     |
| 1  | 1 | 0 | 1 | 13 | CR  | GS  | —  | = | M | ] | m | }   |
| 1  | 1 | 1 | 0 | 14 | SO  | RS  | .  | > | N | ^ | n | ~   |
| 1  | 1 | 1 | 1 | 15 | SI  | US  | /  | ? | O | _ | o | DEL |



# ASCII-Tabelle

## Interpretation:

- zuerst die Zeile lesen
- dann die Spalte

Das große **A** entspricht dem hexadezimalen Wert 0x41



# Die formatierte Ausgabe

Die Ausgabe von print ist mittels Pluszeichen kombinierbar:

```
vorname = input("Bitte geben Sie Ihren Vornamen ein: ")
nachname = input("Bitte geben Sie Ihren Nachnamen ein: ")
print('Herzlich willkommen ' + vorname + ' ' + nachname + '!')
```

Die Ausgabe des Programms führt zu folgender Ausgabe:

```
Bitte geben Sie Ihren Vornamen ein: Donald
Bitte geben Sie Ihren Nachnamen ein: Duck
Herzlich willkommen Donald Duck!

Process finished with exit code 0
```



# Die formatierte Ausgabe

Elegantere Art, um die gleiche Ausgabe zu erreichen: **F-String**

```
vorname = input("Bitte geben Sie Ihren Vornamen ein: ")  
nachname = input("Bitte geben Sie Ihren Nachnamen ein: ")  
print(f'Herzlich willkommen {vorname} {nachname}!')
```

Die Ausgabe des Programms führt zu folgender Ausgabe:

```
Bitte geben Sie Ihren Vornamen ein: Donald  
Bitte geben Sie Ihren Nachnamen ein: Duck  
Herzlich willkommen Donald Duck!  
  
Process finished with exit code 0
```





# Die formatierte Ausgabe

Formatierte Ausgabe mittels Formatbezeichner:

```
zahl_pi = 3.14159265359  
print("Die Zahl Pi auf drei Nachkommastellen gerundet lautet: %.3f" % (zahl_pi))
```

Die Ausgabe des Programms führt zu folgender Ausgabe:

```
Die Zahl Pi auf drei Nachkommastellen gerundet lautet: 3.142  
  
Process finished with exit code 0
```



# Die formatierte Ausgabe

Formatierung mit dem Steuerzeichen `\t` bzw. Tabulator:  
Beispielprogramm ohne Tabulator

```
var1 = 10  
var2 = 20  
print(f'Die erste Variable enthält den Wert: {var1}')
```

```
print(f'Die zweite Variable enthält den Wert: {var2}')
```

Ausgabe:

```
Die erste Variable enthält den Wert: 10  
Die zweite Variable enthält den Wert: 20  
  
Process finished with exit code 0
```



# Die formatierte Ausgabe

Formatierung mit dem Steuerzeichen `\t` bzw. Tabulator:  
Beispielprogramm mit Tabulator

```
var1 = 10  
var2 = 20  
print(f'Die erste Variable enthält den Wert:\t {var1}')
```

```
print(f'Die zweite Variable enthält den Wert:\t {var2}')
```

Ausgabe:

```
Die erste Variable enthält den Wert:      10  
Die zweite Variable enthält den Wert:     20  
  
Process finished with exit code 0
```





# CloudCommand