8. Bedingte Anweisungen

!! Keine Schleife !!!

8.1 `if` -Anweisung

- = Bedingte Anweisung, Konditionale -Anweisungen werden nur unter bestimmten Bedingungen ausgeführt.
- Syntax:

```
if Bedingung :
    Anweisungen/Funktionen
```

Beispiel:

In [12]:
```
print("Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist")
zahl=13
a=input('Gib eine Zahl zwischen 0 und 20 ein')
a=int(a)
if a == 13:
    print("Richtig")
```

```
Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist
```

8.2 `if-else` -Anweisung

- zwischen zwei Blöcken kann mit einer Bedingung ausgewählt werden.
- Syntax:

```
if Bedingung :
    Anweisungsblock
else:
    Anweisungsblock
```

Beispiel:

In [13]:
```
print("Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist")
zahl=13
a=input('Gib eine Zahl zwischen 0 und 20 ein')
a=int(a)
if a == 13:
    print("Richtig")
else:
    print("Falsch")
```

```
Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist
Falsch
```

8.3 `if-elif-else` -Anweisung

- verkettete bedingte Anweisung
- zwischen mehreren Blöcken kann mit Bedingungen ausgewählt werden.
- Syntax

```
if Bedingung :
    Anweisungsblock
elif:
    Anweisungsblock
else:
    Anweisungsblock
```

Beispiel:

In [14]:
```
print("Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist")
zahl=13
a=input('Gib eine Zahl zwischen 0 und 20 ein')
a=int(a)
if a == 13:
    print("Richtig")
elif a<13:
    print("Deine Zahl ist zu klein")
else:
    print("Diene Zahl ist zu groß")
```

```
Ich denke mir eine Zahl zwischen 0 und 20 . Errate, welche es ist
Deine Zahl ist zu klein
```

8.4 verschachtelte Konditionale

- Bedingte Anweisungen können verschachtelt werden
- durch die Einrücktiefe ist immer klar, zu welchem Block die Anweisungen gehören.

Beispiel:

In [15]:
```
# Funktionsdefinition
def myf (var1):
    if var1 > 10:
        if var1%2==0:
            print("Die Zahl ist größer als 10 und gerade ")
        else:
            print("Die Zahl ist größer als 10 und ungerade")
    else:
        print("Die Zahl ist kleiner oder gleich 10")

# Eingabe der Zahl
a= input("Gib die Zahl ein")
a=int(a)
#Aufruf der Funktion
myf(a)
```

```
Die Zahl ist kleiner oder gleich 10
```

8.4 Bedingungen

8.4.1 Datentyp `bool`

- Bolsche Ausdrücke haben entweder den Wert `True` oder `False`.
- zum Vergleich wird der Vergleichsoperator == verwendet: `a == b`
- Arithmeitsche Opeationen konvertieren Boolsche Werte nach int: `False` wird 0, `True` wird 1
- Beispiel:

In [16]:
```
assert 13==13
assert not ('veggie'=='meatie')
assert type ('veggie'=='meatie')==bool
```

- Vergleichsoperatoren:

Syntax	Bedeutung
a == b	Ist a gleich b
a !=b	Ist a ungleich b?
a < b	Ist a echt kleiner als b?
a>b	Ist a echt größer als b?
a <= b	Ist a kleiner oder gleich b?
a >= b	Ist a echt größer als b?

- Bemerkungen:

In [17]:
```
# Gleitkommazahlen werden nicht exakt dargestellt
assert 2.1 - 2.0 > 0.1

assert not (2-1<1)

assert False < True
```

8.4.2 Der `String`-Vergleich

- Character werden anhand des Ergebnisses der `ord`-Funktion verlichen.
- Strings werden anhand der **lexikographische Ordnung** verglichen. (Telefonbuch)
- Beispiele

In [18]:
```
'paul' < 'paula'
# paul ist ein Präfix von antonia
```

Out[18]:
```
True
```

In [19]:
```
'anton' < 'berta'
#Vergleich der einander entsprechenden Buchstaben von links nach rechts
```

Out[19]: True

In [20]:
```
'anton'<'urs'
```

Out[20]: True

In [21]:
```
'antonia' < 'antonella'
```

Out[21]: False

8.4.3 Vergleich ungleicher Typen

- Werte unvergleichbarer Typen sind ungleich.
- Zahlen (`int` , `float` , `complex` , `bool`) nicht über '==' mit Strings vergleichbar
- `False` gibt es direkt als Ergebnis
- Bei der Anordnungsrelation gibt es eine Fehlermeldung
- Beispiel:

In [22]:
```
13 == 'dreizehn'
```

Out[22]: False

In [23]:
```
41 < '43'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/martin/Workspace/Jupyter_Notebooks/Informatik_KS/1_Python_Programmieren/1-8_Bedingte_Anweisungen/1_8_Bedingte_Anwweisungen.ipynb Cell 23 line 1
----> <a href='vscode-notebook-cell:/Users/martin/Workspace/Jupyter_Notebooks/Informatik_KS/1_Python_Programmieren/1-8_Bedingte_Anweisungen/1_8_Bedingte_Anwweisungen.ipynb#X35sZmlsZQ%3D%3D?line=0'>1</a> 41 < '43'

TypeError: '<' not supported between instances of 'int' and 'str'
```

8.4.4 Logische Operatoren auf `bool``

- `or` , `and` , `not wirken wie die Bitoperationen
- Beispiel: `x<10 or y>100` Hat den Wert `True` , wenn x kleiner 10 ist oder wenn y größer als 100 ist.
- Beispiel: `1<= x and x <= 10` hat den Wert `True` , wenn x zwischen 1 und 10 (inklusive) liegt.
- Beispiel: `1<=x<=10` ist alternativ auch möglich.
- Beipsiel: `not(x<y)` ist dann `True` , wenn x nicht kleiner als y ist.
- Nullwerte, wie `None` , `0` , `0.0` , und `''` werden als `False` behandelt, alle anderen Werte als `True`

- Kurzschlussauswertung: Die Auswertung der logischen Operatoren wird beendet, wenn das Ergebnis klar ist (short-cut evaluation)
- Beipsiele:

In [26]:
```
1 < 5 < 10
```

Out[26]: True

In [27]:
```
5 < 1 or 'text'<'abc'
```

Out[27]: False

In [25]:
```
'text' or True
```

Out[25]: 'text'

In [28]:
```
'' or 'default'
```

Out[28]: 'default'

In [29]:
```
'good night' and 'thank you'
```

Out[29]: 'thank you'

In [30]:
```
0 and 10 < 100
# 0 zuählt als False
```

Out[30]: 0

In [32]:
```
not 'text' and (None or 0.0)
# 'text' ist True, not 'text ist somit False
```

Out[32]: False