Coding

Datentypen und Kontrollstrukturen

Quelle: https://images.app.goo.gl/BPeK8166dgCBJJwk9

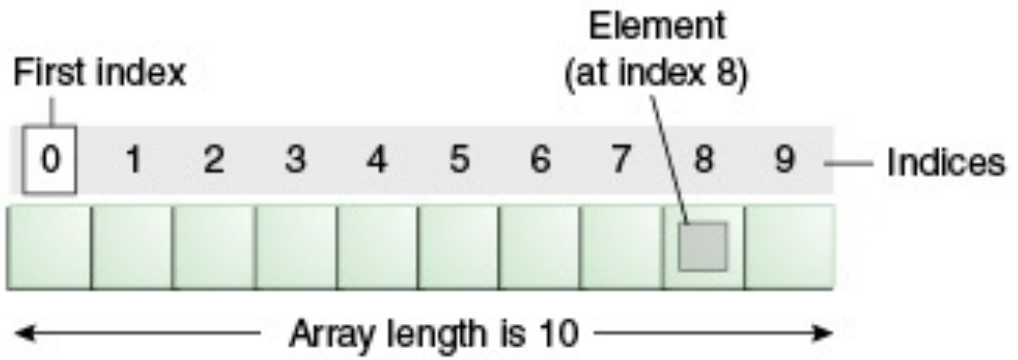
Datentypen

- Informationsverarbeitung in einem Programm → Speicherplätze des Rechners werden zur Aufnahme dieser Informationen reserviert (Variablen)
- Bevor Variablen benutzt werden können, müssen sie zuerst deklariert/initialisiert werden
- Regeln für Variablennamen:
  - Der Name kann aus Buchstaben, Zahlen und Bei- bzw. Unterstrichen bestehen
  - Variablennamen werden üblicherweise klein geschrieben
  - Das erste Zeichen muss ein Buchstabe sein

TYP	BESCHREIBUNG	BEISPIEL	RANGE
Boolean	Wahrheitswerte	true oder false	0 – 1
Integer	Ganzzahl	5	-32.768 – 32.767
Float	Fließkommazahl	4,36	1.17E-38 – 3.4E38
String	Zeichenkette	hallo	-128 – 127

Arrays

- Viele gleichartig strukturierte Daten werden in einer Liste gespeichert
- Zugriff auf bestimmte Inhalte des Feldes (Array) mittels Indizes



Quelle: https://images.app.goo.gl/tEarmi7Rx5grHhtj8

Kontrollstrukturen

- Kontrollstrukturen kontrollieren Programmablauf
- Programmteile werden nur unter gewissen Bedingungen ausgeführt oder wiederholt

- Problem wird übersichtlich und strukturiert formuliert
- Wichtigste Kontrollelemente:
  - Verzweigungen (Entscheidungen)
  - Schleifen
- Es können mehrere Kontrollstrukturen miteinander kombiniert werden
  - z.B. mehrere Schleifen ineinander verschachtelt
- Programmcode wird immer **VON OBEN NACH UNTEN** abgearbeitet!

Verzweigungen

IF/ELSE

- Damit ermöglichen Kontrollstrukturen die Auswahl zwischen zwei Szenarien
- Basiert auf Bedingungen, die entweder wahr oder falsch sind

```
if car.isBatteryLow():
    car.charge()
else:
    car.drive()
```

IF/ELIF/ELIF/ELSE

- Else-If-Bedingungen wahr, wenn If-Bedingungen falsch sind
- Else-Bedingungen wahr, wenn auch Else-If-Bedingungen falsch sind

```
if car.isBatteryLow() && car.isInDockingStation():
    car.charge()
elif car.isBatteryLow():
    car.driveToDockingStation()
else:
    car.drive()
```

Schleifen

→ Wiederholungsanweisungen

- **WHILE** Schleifen

```
i = 0
while i<10:
    print(i)
    i += 1
```

- **FOR** Schleifen

```
languages = [‘DE‘, ‘EN‘, ‘IT‘]
for x in languages:
    print(x)
```

- Fortfahren bzw. Abbrechen mit:
  - **continue** bzw. **break**

Operatoren

- Schlüsselwörter die in Programmiersprachen für die Zuweisung von Werten zu Variablen verantwortlich sind
- Arithmetische Operatoren (* vor / und Punkt- vor Strichrechnung):
  - + Addition
  - - Subtraktion
  - * Multiplikation
  - / Division
- Typen von Operatoren:
  - Zuweisungsoperatoren z.B. x = 1
  - Vergleichsoperatoren z.B. x == 1
  - Berechnungsoperatoren z.B. x = a + b
  - Logische Operatoren z.B. a == 1 && b == 2
  - Bit-Operatoren z.B. x | | y
  - Zeichenketten-Operatoren z.B. print(3 * ‘ab‘ + ‘cd‘)

Algorithmus

- Algorithmus → zentrales Werkzeug der Informatik
  - Bestandteil eines jeden Computerprogramms
- **Jeder Algorithmus ist eine spezifische Methode zur Lösung eines bestimmten Problems!**
- Nutzung mathematischer Verfahren zur Lösung komplexer Probleme
- Vollständige Formulierung einer exakten Verfahrensvorschrift
- Lesbarkeit für den Computer:
  - Programmierer schreibt lesbarem Quellcode (Syntax)
  - Compiler übersetzt Quellcode in Maschinensprache (Assembler Code)
  - Assembler besteht aus Bits und Bytes → also Nullen und Einsen
- Beispiele: Bubble Sort, Quick Sort, Merge Sort, Heap Sort, Hashing usw.

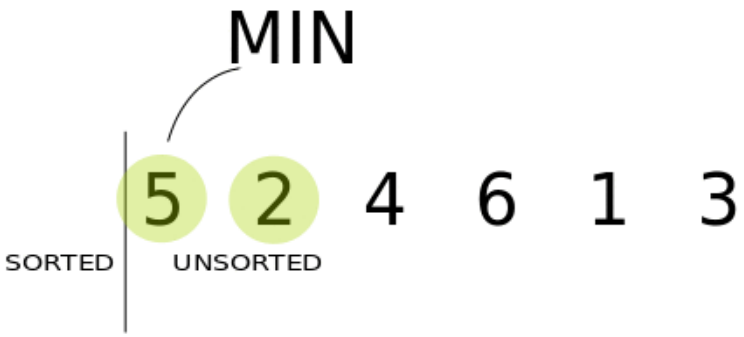
Sortieren

- Sortierverfahren unterscheiden durch Arbeitsweise:
  - **Vergleichsbasierte** Sortierverfahren → Vergleiche von Elementen der Liste
  - **Nicht-Vergleichsbasierte** Sortierverfahren → Fokus auf der konditionierten Eingabe
- Zusätzliche Unterscheidung:
  - **Stabil** → Reihenfolge der Datensätze bleibt gleich, wo Sortierschlüssel auch gleich sind
  - **Instabil** → Verschiedene Endergebnisse nach einem Sortiervorgang

Beispiel „Stabiles Sortierverfahren“ – Insertion-Sort:

Quelle: https://images.app.goo.gl/JnX8fAFo9dPSyhuQ9

Beispiel „Instabiles Sortierverfahren“ – Selection-Sort:



Quelle: https://images.app.goo.gl/HyaF1sBkUht7pkyr9