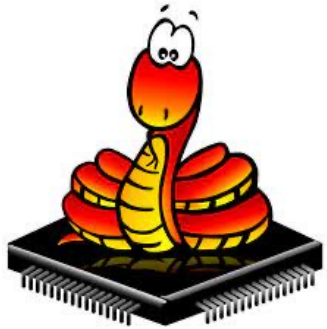
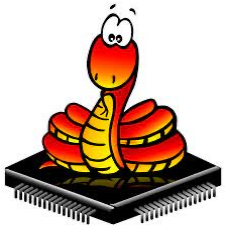
Kapitel 4 - Mikrocontroller
Python für Mikrocontroller: *MicroPython*

Reinhard Weiß, 19.03.2021



Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Grundlagen.



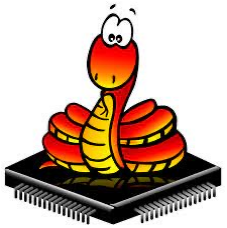
Was ist objektorientierte Programmierung?

Daten und Funktionen gehören zusammen. Wenn Sie eine Liste erzeugen, so haben Sie automatisch Zugriff auf die Methoden der Liste.

In [ ]:

```
1 a = [3,5,2,1,4,6,4,6,4,6,3,28,6,4]
2 type(a)
3 a.index(max(a))
```

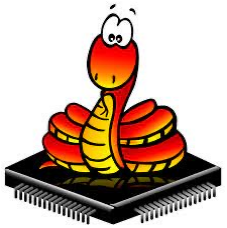
Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Grundlagen.



Datenstrukturen einer Liste (`list`):

```
list.append(x)
list.insert(i, x)
list.remove(x)
list.pop([i])
list.clear()
list.index(x[, start[, end]])
list.count(x)
list.sort(*, key=None, reverse=False)
list.reverse()
list.copy()
```

Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Grundlagen.



Zwei Methoden mit demselben Namen:

```
1 a = [3,5,2,1,4,6,4,6,4,6,3,28,6,4]
2 type(a)
```

```
list
```

```
1 b = (3,5,2,1,4,6,4,6,4,6,3,28,6,4)
2 type(b)
```

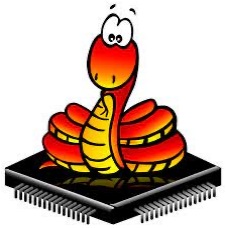
```
tuple
```

```
1 a.index(6)
```

```
5
```

```
1 b.index(6)
```

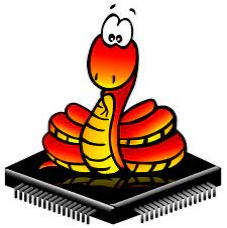
```
5
```



Zwei Methoden mit demselben Namen:

In einer nicht objektorientierten Programmiersprache würde es nur eine Funktion mit dem Namen „`index`“ geben.

Aufruf: `index(a,6)`

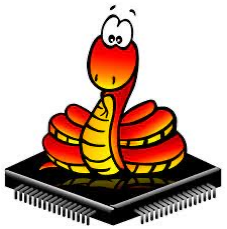


Zwei Methoden mit demselben Namen:

Ziel einer objektorientierten Programmiersprache ist es, die Komplexität der Objekte vor dem Benutzer zu verbergen und stattdessen klar definierte Schnittstellen zur Verfügung zu stellen.

Das geschieht über definierte Klassen. In einer Klasse wird festgelegt, wie die Daten intern aufgebaut sind und welche Methoden zur Verfügung stehen.

Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Definition von Klassen.



Beispiel Planeten:

- Definition der Klasse:

```
1  class Koerper:
2      def __init__(self, name, masse):
3          self.name = name
4          self.masse = masse
```

- Erzeugung eines Elements der Klasse `Koerper`:

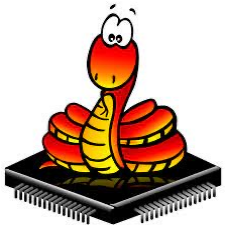
```
1  k = Koerper('Jupiter', 1.89813e27)
```

- Aufrufen der Attribute des Elements:

```
1  print(k.masse)
```

```
1.89813e+27
```

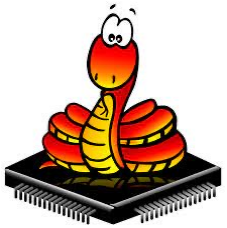
Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Definition von Klassen.



Weitere Methoden:

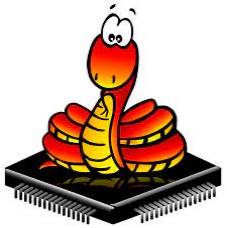
```
1  class Koerper:
2      def __init__(self, name, masse):
3          self.name = name
4          self.masse = masse
5      def erdmassen(self):
6          return self.masse / 5.9722e24
7      def __str__(self):
8          return f'Körper {self.name}: m = {self.masse} kg'
```

Kapitel 4 - Mikrocontroller
Objektorientierte Programmierung
Vererbung



Vererbung:

```
1  import math
2  from klasse2 import Koerper
3
4  class Planet(Koerper):
5      def __init__(self, name, masse, radius):
6          super().__init__(name,masse)
7          self.radius = radius
8      def volumen(self):
9          return 4/3 * math.pi * self.radius ** 3
10     def dichte(self):
11         return self.masse / self.volumen()
12
```



Überschreiben:

```python
import math
from klasse3 import Planet

class Planet2(Planet):
    def __str__(self):
        return f'Planet {self.name}: m = {self.masse} kg'

if __name__ == '__main__':
    jupiter = Planet2('Jupiter', 1.89813e27, 6.9911e7)
    rho = jupiter.dichte()

    print(jupiter)
    print(f'Dichte: {rho/1e3} g/cm³')
```

Vielen Dank
für Ihre Aufmerksamkeit.