Objektorientierung mit Python

Python
ist ...

Objektorientierung mit Python

Python ist interaktiv

- IDLE bietet die Möglichkeit, Anweisungen direkt auswerten zu lassen und das Ergebnis angezeigt zu bekommen
- Der Editor ist ein struktureller Editor, er berücksichtigt also die Python-Regeln
- Ein Python-Programm kann sofort ausgetestet werden

Objektorientierung mit Python

Python ist typgebunden

- Variable kennen ihren Typ
- Eine Deklaration von Typen ist nicht notwendig
- a=10
  definiert eine Variable a und weist ihr den Wert 10 zu, der damit Zahltyp ist
- a="10" [oder a='10' mit derselben Wirkung]
  definiert den string

Objektorientierung mit Python

Python ist prozedural

- Anweisungen werden sequenziell ausgeführt
- Mit der Anweisung
  def <Name>(<par>):
      <Anweisung 1>
      <Anweisung 2>
  können Prozeduren definiert werden

Objektorientierung mit Python

Python ist funktional

- Mit der Definition
  def <Name>(<par>):
      return <Rückgabewert>
  können Funktionen definiert werden
- Funktionsschachtelungen werden "normal" ausgewertet
- Rekursion ist möglich

Objektorientierung mit Python

Python ist voll funktional

- Es gibt die Möglichkeit anonyme Funktionen zu definieren
  lambda x: x*x
  ist die Quadratfunktion
- apply(lambda x,y: (x+y)/2, (3,5))
  wendet die Mittelwertfunktion auf das Tupel an
- Funktionen können Parameter sein

Objektorientierung mit Python

Python ist voll funktional mit einem Mangel

- Python erkennt keine Endrekursion!
- Das führt leider unnötigerweise oft zu Rekursionsabbruch mit Fehler

Objektorientierung mit Python

Python ist objektorientiert

- Ein großer Teil der Sprache ist objektorientiert definiert
- Das hat auch Probleme zur Folge:
  - Listen sind Objekte
  - call-by-reference
  - Methoden verändern das Objekt dauerhaft
  - Zuweisungen erzeugen keine neuen Objekte