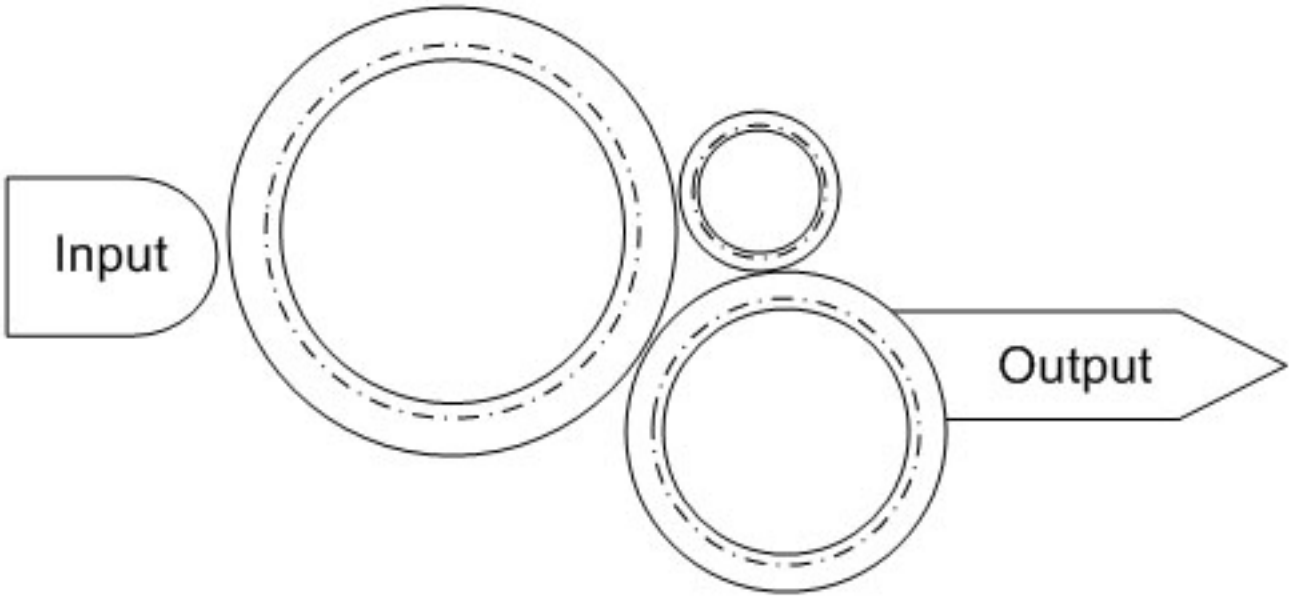
Python
„Modularisierung“



Modularisierung

- Das Projekt wird in verschiedene kleine Aufgaben zerlegt.
- Logische Gliederung von Programmteilen mit Hilfe von Namensräumen.
- Aufteilung von einem sehr großen Programm in mehrere Quelltext-Dateien.

Module

- Dateien mit der Endung „.py“
- Klassen werden in Modulen definiert.
- Die Standardmodule werden automatisiert bei der Installation von Python installiert.
- Module können selbstgeschrieben oder von Drittanbietern bereitgestellt werden.

Standardmodule

- Sortiert nach Kategorien.
  https://docs.python.org/3.6/library/.
- Alphabetisch sortiert.
  https://docs.python.org/3/py-modindex.html.

Benutzerdefinierte Module

- Kundenspezifische Programmierung.
- Eindeutiger Dateiname in einem Namensraum.
- Die Dateiendung regelt die Nutzung des Moduls. „.py“ enthalten Konsolen-Programme. „.pyw“ nutzen eine grafische Oberfläche für die Ein- und Ausgabe von Daten.

Regeln für den Datennamen

- Der Dateiname sollte mit einem Buchstaben beginnen.
- Ein Dateiname besteht aus den lateinischen Groß- und Kleinbuchstaben und den Ziffern.
- Ein Dateiname enthält als Sonderzeichen nur den Unterstrich.
- Der Dateiname darf kein Punkt enthalten. Der Punkt trennt den Dateinamen vom Dateityp.
- Die Groß- und Kleinschreibung wird beachtet.
- Namen von Standardbibliotheken sind nicht erlaubt.

Import des gesamten Moduls

```
import math

def getSinus():
    sinus =  math.sin(math.pi / 2)
    print("Sinus: ", sinus)
```

Weiteres Beispiel

```
import modul_classPunkt

class ClsKreis(object):
    def __init__(self, radius = 1, x = 0, y = 0):
        self.__radius = radius
        self.__position = modul_classPunkt.ClsPunkt(x, y)
```

import-Anweisung

import	math
import	Modul

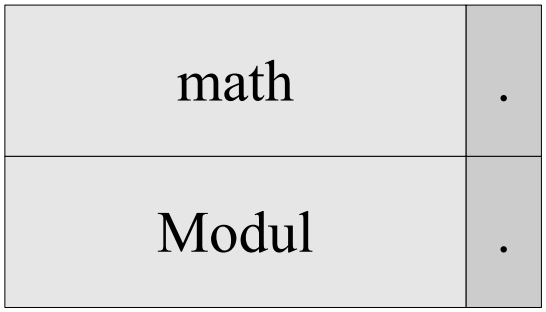
- Mit Hilfe des Schlüsselwortes import wird ein Standardmodul oder ein benutzerdefiniertes Modul in eine Code-Datei eingebunden.
- Dem Schlüsselwort folgt der Dateiname des Moduls ohne Angabe der Dateiendung.
- Bereitstellung einer Klasse Funktionen, Methoden und Datentypen aus einem Modul.

Nutzung von Funktionen

math	.	sin	(	math.pi / 2	)
Modul	.	Funktion	(	parameter	)

- Das Modul ist vollständig in die Code-Datei importiert.
- Der Aufruf der Funktion erfolgt über einen qualifizierten Namen.
- Mit Hilfe des Punktoperators wird die Funktion mit dem dazugehörigen Modul verbunden. Die angegebene Funktion ist in dem Modul definiert.

Wo ist das Element definiert?



- Links vom Punkt wird der Name des Moduls angegeben.
- In diesem Modul ist das gewünschte Element definiert.
- Datei mit der Endung „.py“, in dem die Funktion abgelegt ist.

Welches Element wird angesprochen?

math	.	pi
modul	.	elment

- Rechts vom Punkt steht der Name der aufzurufenden Funktion. Die Funktion wird entsprechend ihres Funktionskopfes aufgerufen.
- Rechts vom Punkt steht der Name einer definierten Variablen. Der Wert der Variablen wird im Ausdruck genutzt.

Nutzung von Klassen

modul_classPunkt	.	clsPunkt	(	x	,	y	)
modul	.	Klasse	(	parameterliste			)

- Das Modul ist vollständig in die Code-Datei importiert.
- Die Erzeugung der Instanz erfolgt über einen qualifizierten Namen.
- Mit Hilfe des Punktoperators wird die Klasse mit dem dazugehörigen Modul verbunden. Die angegebene Klasse ist in dem Modul definiert und dient als Vorlage für die Instanz.

Wo ist die Klasse definiert?

modul_classPunkt	.
modul	.

- Links vom Punkt wird der Name des Moduls angegeben.
- In diesem Modul ist die gewünschte Klasse definiert.
- Die Vorlage ist in einer Datei mit der Endung „.py“ abgelegt.

Von welcher Klasse wird eine Instanz erzeugt?

modul_classPunkt	.	clsPunkt	(	x	,	y	)
modul	.	Klasse	(	parameterliste			)

- Rechts vom Punkt steht der Name einer Klasse. Diese Klasse dient als Vorlage für die zu erzeugenden Instanz.
- Die Instanz wird zum Beispiel in einer anderen Klasse als Attribut genutzt. Zum Beispiel wird in der Initialisierungsmethode clsKreis eine Attribut clsPunkt erzeugt, der Mittelpunkt eines Kreises beschreibt.

Import von Funktionen aus einem Modul

```
form math import sin, pi

def getSinus():
    sinus =  sin(pi / 2)
    print("Sinus: ", sinus)
```

import-Anweisung

from	math	import	sin	,	pi
from	modul	import	Funktion	,	Funktion

- Bestimmte Funktionen werden aus einem Modul benötigt.
- Von dem Modul importiere Funktion A, Funktion B und so weiter.
- Dem Schlüsselwort import folgt eine Liste von Funktionsnamen. Die Namen werden durch ein Komma getrennt.

Aufruf der importierten Funktionen

sin	(	math.pi / 2	)
Funktion	(	parameter	)

- Die importierte Funktion wird mit einem unqualifizierten Namen aufgerufen.
- Der Aufruf entspricht dem Funktionskopf der Funktion.

Modul sys

```
import sys

print("Version: ", sys.version)
print("Plattform: ", sys.platform)
```

- Das Modul sys enthält Informationen zum aktuellen Python-System.
- Mit Hilfe von dir(sys) können alle Funktionen, Konstanten etc. aus dem Modul angezeigt werden.

Version und Plattform

```
import sys

print("Version: ", sys.version)
print("Plattform: ", sys.platform)
```

- sys.version zeigt die Python-Version an.
- sys.platform gibt das Betriebssystem, auf dem das Python-Programm läuft, zurück.

Systempfade

```
import sys

sys.path.append(r"C:\pythonkurs\\")

for verzeichnis in sys.path:
    print(verzeichnis)
```

- sys.path zeigt die System-Pfade von Python an.
- Diese Pfade werden nach den importierten Modulen durchsucht.
- Die Methode .append() fügt der Liste weitere Pfadangaben hinzu.

Dokumentation von Modulen

- Kurze Beschreibung einer Klasse
- Dokumentation der Argumente und des Rückgabewertes einer Methode.
- Mit Hilfe des Befehls help(klassenname) kann die Dokumentation zu einer Klasse angezeigt werden.

DocString

- Gestaltung der Hilfeseiten zu einem Modul.
- Direkt in der Zeile vor einem Klassenkopf, um die Klasse zu beschreiben.
- Direkt in der Zeile nach einem Methodenkopf, um die Parameter und das Verhalten der Methode zu erläutern.

Beispiel

```
""" Fläche berechnen

        Arguments:
            self : object

        Returns
            ClsKreis_object.getFlaeche() -> float
            pi * (pow(radius,2))
 """
```

Erläuterung

- Docstrings sind spezielle Kommentare. Die Kommentare beginnen und enden mit drei Anführungszeichen oder Apostrophs.
- Mit Hilfe des Befehls help(name) wird die Dokumentation zu zu einem Modul, Klasse oder Methode angezeigt. In Abhängigkeit der übergebenen Modul-, Klassen- oder Methodennamens wird eine Hilfeseite geöffnet.

Namensraum

- Gültigkeit einer Variablen, Methode etc.
- Vermeidung von Namenskonflikte bei benutzerdefinierten Elementen.
- In der realen Welt sind zum Beispiel Postleitzahlen oder Vorwahlnummern bei Telefonnummern Namensräume. Ein bestimmtes Gebiet wird gekennzeichnet.

Lokaler Namensraum

```
def  printInformation(self):
        koordinaten = self.__position.getPosition()
        xKoordinate = str(koordinaten[0])
        yKoordinate = str(koordinaten[1])

        strInfo = "Der Kreis hat einen Radius von " + str(self.radius)
        strInfo = strInfo +  "\nDer Kreis wurde an dem Punkt "
        strInfo = strInfo + xKoordinate
        strInfo = strInfo + "," + yKoordinate
        strInfo = strInfo + " abgelegt."
        print(strInfo)
```

Erläuterung

- Bezeichner von lokalen Variablen in einer Funktion oder Methode.
- Parameter einer Funktion oder Methode.
- Elemente, die nur in einer Funktion / Methode bekannt sind.

Modularer Namensraum

- Bezeichner von globalen Variablen in einem Modul. Globale Variablen sollten, wenn möglich nicht genutzt werden.
- Eine Klasse ist in einem Modul definiert. Der Klassenname kann in der Code-Datei überall genutzt werden.

Eingebauter Namensraum

- In Python definierte Namen sind überall gültig.