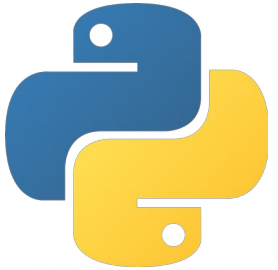
Python

Anbindung an andere Programmiersprachen



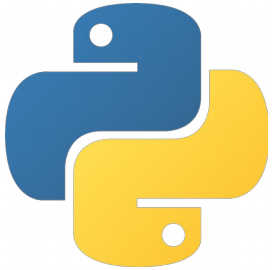
- Ausschließlich an der Sprache C
  - Dabei gibt es zwei wesentliche Szenarien zu beachten

    1. In einem Python-Programm soll C-Code ausgeführt werden

    2. In einem C-Programm soll ein Python-Script ausgeführt werden

Python

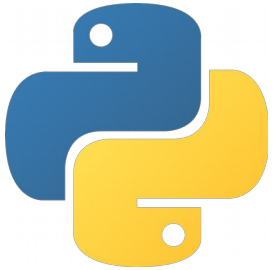
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
  - Zu 1.
    - In einem größeren Projekt kann Python als komfortable und gut zu wartende Sprache beispielsweise für die Programmlogik eingesetzt werden, während man einige wenige zeitkritische Algorithmen des Projekts aus Effizienzgründen in einer nicht interpretierten Sprache wie C oder C++ schreibt
    - Schauen wir wie Sie mit dem Modul ctypes der Standardbibliothek auf dynamische Bibliotheken, beispielsweise Windows-DLLs, zugreifen können
    - Der zweite Abschnitt zeigt Möglichkeiten auf, C- oder C++-Code direkt in den Python-Quelltext einzubetten

Python

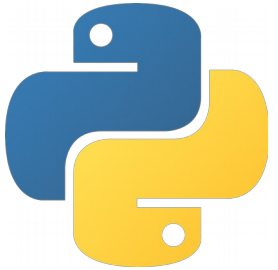
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
  - Zu 2.
    - Häufig möchte man auch den umgekehrten Weg beschreiten und in einem größeren C/C++-Projekt Python als eingebettete Scriptsprache für dynamische Elemente des Programms verwenden
    - In einem Computerspiel könnte beispielsweise C/C++ für die hauptsächlich laufzeitkritische Hauptanwendung und die gesamte Algorithmik verwendet werden, während Python für die dynamischen Elemente des Spiels, beispielsweise Ereignisse in bestimmten Levels oder das Verhalten verschiedener Spielfiguren, genutzt wird

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes

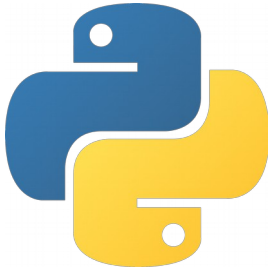
  - Mit dem Modul ctypes ist es möglich, Funktionen einer sogenannten dynamisch ladbaren Bibliothek, im Folgenden dynamische Bibliothek genannt, aufzurufen

  - Zu solchen dynamischen Bibliotheken zählen beispielsweise DLL-Dateien (Dynamic Link Library) unter Windows oder SO-Dateien (Shared Object) unter Linux bzw. Unix

  - Das Aufrufen von Funktionen einer dynamischen Bibliothek ist besonders dann sinnvoll, wenn bestimmte laufzeitkritische Teile eines Python-Programms in einer hardwarenäheren und damit effizienteren Programmiersprache geschrieben werden sollen oder wenn Sie schlicht eine in C oder C++ geschriebene Bibliothek in Python nutzen möchten

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes

  - Beachten Sie grundsätzlich, dass das Erstellen einer dynamischen Bibliothek keine Eigenschaft der Programmiersprache C ist

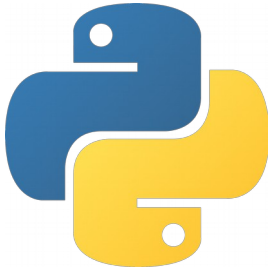
  - Im Gegenteil: Eine dynamische Bibliothek kann als eine sprachunabhängige Schnittstelle zwischen verschiedenen Programmen betrachtet werden

  - Es ist absolut möglich, ein Python-Programm zu schreiben, das auf eine in C geschriebene dynamische Bibliothek zugreift, die ihrerseits auf eine dynamische Bibliothek zugreift, die in Pascal geschrieben wurde

  - Dies gilt allerdings nur für Sprachen, die sich zu einer dynamischen Bibliothek kompilieren lassen – PHP bliebe beispielsweise außen vor

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes

  - Zum Einstieg in das Modul ctypes dient ein einfaches Beispiel

  - Im Beispielprogramm soll die dynamische Bibliothek der C Runtime Library eingebunden und die darin enthaltene Funktion printf dazu genutzt werden, den Text »Hallo Welt« auszugeben

  - Die C Runtime Library ist unter Windows unter dem Namen msvcrt.dll und unter Unix-ähnlichen Systemen unter dem Namen libc.so.6 zu finden

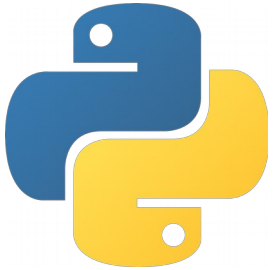
  - Dazu betrachten wir zunächst den Quellcode des Beispielprogramms

```
import ctypes

bibliothek = ctypes.CDLL("MSVCRT")
bibliothek.printf(b"Hallo Welt\n")
```

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes

  - Zunächst wird das Modul ctypes eingebunden und dann eine Instanz der Klasse CDLL erzeugt

  - Eine Instanz dieser Klasse repräsentiert eine geladene dynamische Bibliothek

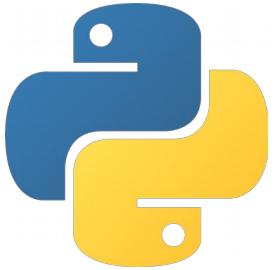
  - Beachten Sie, dass die Dateiendung .dll unter Windows weggelassen werden muss, wenn eine System-Bibliothek geladen werden soll

  - Unter Linux sieht die Instanziierung der Klasse CDLL folgendermaßen aus

    bibliothek = ctypes.CDLL("libc.so.6")

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes

  - Nachdem die dynamische Bibliothek eingebunden worden ist, kann die Funktion printf wie eine Methode der CDLL-Instanz aufgerufen werden

  - Die normale printf-Funktion erwartet einen bytes-String

  - Um mit richtigen Strings zu arbeiten, kann ihr Pendant wprintf gerufen werden

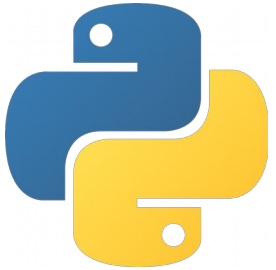
    bibliothek.wprintf("Hallo Welt\n")

  - Behalten Sie bei der Arbeit mit ctypes aber immer im Hinterkopf, dass es teilweise große Unterschiede zwischen den Datentypen von C/C++ und denen von Python gibt

  - Es können also nicht alle Funktionen so einfach verwendet werden

Python

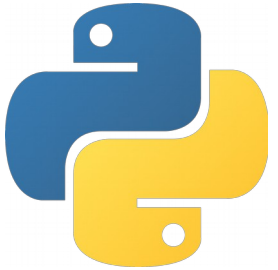
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Die eigene Bibliothek
    - An dieser Stelle soll eine dynamische Bibliothek erstellt werden, auf die wir dann in den folgenden Abschnitten zugreifen werden
    - Die Bibliothek ist in C geschrieben und enthält drei Funktionen mit unterschiedlich komplexer Schnittstelle
    - Wir werden gleich den Quelltext der drei Funktionen sehen, auf die wir uns in den folgenden Beispielen beziehen
    - Das erstellen einer dynamischen Bibliothek ist hier nicht Bestandteil, wollen Sie sich trotzdem damit befassen, so müssen Sie Recherche im Internet durchführen

Python

Anbindung an andere Programmiersprachen



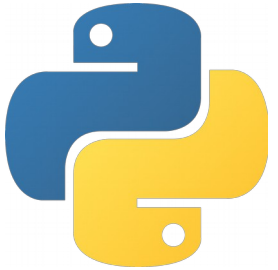
- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Die eigene Bibliothek

    - Inhalt der DLL

```
// Berechnet die Fakultaet einer ganzen Zahl
int fakultaet(int n)
{
	int i;
	int ret = 1;
	for(i = 2; i <= n; i++)
		ret *= i;
	return ret;
}
// Berechnet die Laenge eines Vektors im R^3
double veclen(double x, double y, double z)
{
	return sqrt(x*x + y*y + z*z);
}
```

Python

Anbindung an andere Programmiersprachen



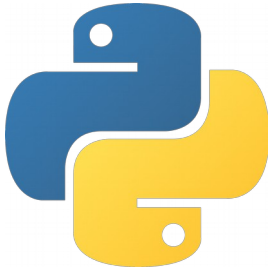
- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Die eigene Bibliothek

    - Inhalt der DLL

```
// Bubblesort
void sortiere(int *array, int len){
      int i, j, tmp;
      for(i = 0; i < len; i++){
         for(j = 0; j < i; j++){
              if(array[j] > array[i]){
                  tmp = array[j];
                  array[j] = array[i];
                  array[i] = tmp;
              }
         }
    }
}
```

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Die eigene Bibliothek

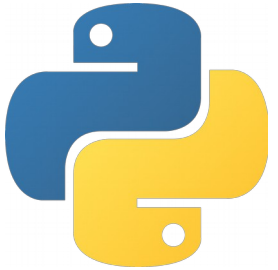
    - Die erste Funktion, fakultaet, berechnet die Fakultät einer ganzen Zahl und gibt das Ergebnis ebenfalls in Form einer ganzen Zahl zurück

    - Die zweite Funktion, veclen, errechnet die Länge eines dreidimensionalen Vektors
      Sie bekommt dazu die drei Koordinaten des Vektors in Form von drei Gleitkommazahlen übergeben und gibt die Länge des Vektors ebenfalls in Form einer Gleitkommazahl zurück

    - Die dritte, etwas komplexere Funktion, sortiere , implementiert den sogenannten Bubblesort-Algorithmus, um ein Array von beliebig vielen ganzen Zahlen aufsteigend zu sortieren
      Dazu übergeben wir der Funktion einen Pointer auf das erste Element sowie die Anzahl der Elemente des Arrays

Python

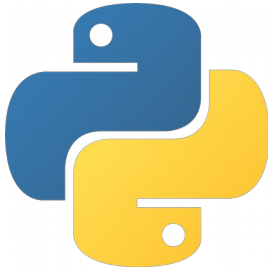
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - An dieser Stelle haben wir eine fertige dynamische Bibliothek mit drei Funktionen, die wir jetzt mit ctypes aus einem Python-Programm heraus aufrufen
    - Der praktischen Umsetzung dieses Vorhabens stehen jedoch die teilweise inkompatiblen Datentypen von C und Python im Weg
    - Solange Instanzen der Datentypen int, str, bytes oder NoneType übergeben werden, funktioniert der Funktionsaufruf einwandfrei, denn diese Instanzen können eins zu eins nach C übertragen werden

Python

Anbindung an andere Programmiersprachen



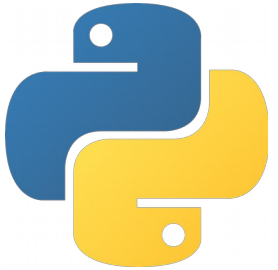
- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - So ist beispielsweise der Aufruf der Funktion fakultaet mit keinerlei Problemen behaftet

```
>>> from ctypes import CDLL
>>> bib = CDLL("bibliothek.dll")
>>> print(bib.fakultaet(5))
120
```

    - Doch bereits das Übergeben einer float-Instanz scheitert
    - Für diesen und andere Datentypen von C implementiert das Modul ctypes entsprechende Gegenstücke in Python, deren Instanzen über die Schnittstelle geschickt werden können

Python

Anbindung an andere Programmiersprachen

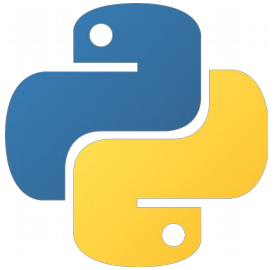


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen

Datentyp (ctypes)	Datentyp (C)	Datentyp (Python)
c_char	char	bytes (ein Zeichen)
c_wchar	wchar_t	str (ein Zeichen)
c_byte	char	int
c_wchar_p	wchar_t *	str, None
c_void_p	void *	int, None

Python

Anbindung an andere Programmiersprachen

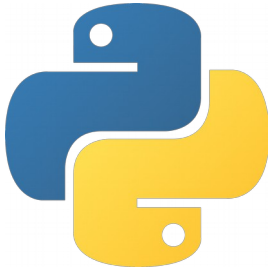


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen

Datentyp (ctypes)	Datentyp (C)	Datentyp (Python)
c_ubyte	unsigned char	int
c_short	short	int
c_ushort	unsigned short	int
c_int	int	int
c_uint	unsigned int	int
c_long	long	int

Python

Anbindung an andere Programmiersprachen

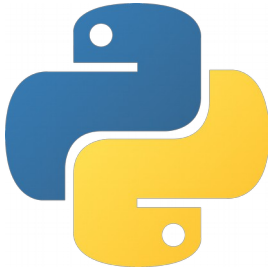


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen

Datentyp (ctypes)	Datentyp (C)	Datentyp (Python)
c_ulong	unsigned long	int
c_longlong	__int64, long long	int
c_ulonglong	unsigned __int64, unsigned long long	int
c_float	float	float
c_double	double	float
c_char_p	char *	bytes, None

Python

Anbindung an andere Programmiersprachen

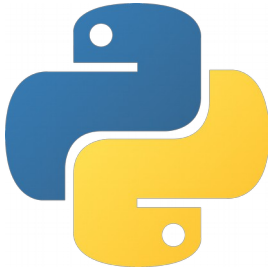


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - All diese ctypes -Datentypen können durch Aufruf ihres Konstruktors instanziiert und mit einer Instanz des angegebenen Python-Datentyps initialisiert werden
    - Über das Attribut value lässt sich der jeweilige Wert eines ctypes-Datentyps verändern

```
>>> import ctypes
>>> f = ctypes.c_float(1.337)
>>> f
c_float(1.3370000123977661)
```

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen

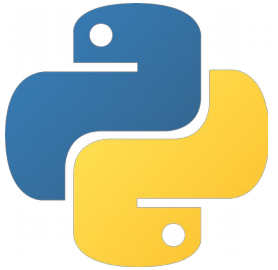
```
>>> d = ctypes.c_double(1.337)
>>> d
c_double(1.337)

>>> s = ctypes.c_char_p(b"Hallo Welt")
>>> s.value
'Hallo Welt'

>>> null = ctypes.c_void_p(None)
>>> null
c_void_p(None)
```

Python

Anbindung an andere Programmiersprachen

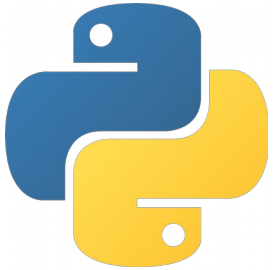


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - Um ein Array eines bestimmten Datentyps anzulegen, wird der zugrundeliegende Datentyp mit der Anzahl der Elemente, die er aufnehmen soll, multipliziert
    - Das Ergebnis ist ein Datentyp, der das gewünschte Array speichert

```
>>> arraytyp = ctypes.c_int * 5
>>> a = arraytyp(1, 5, 2, 1, 9)
>>> a
<__main__.c_long_Array_5 object at 0xb7af82fc>
```

Python

Anbindung an andere Programmiersprachen



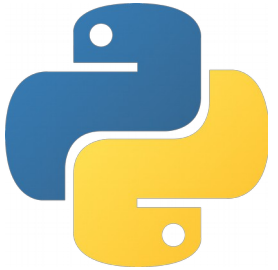
- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - Einen solchen Array-Typ können wir beispielsweise der Funktion sortiere übergeben, die ein Array von ganzen Zahlen sortiert

```
from ctypes import CDLL, c_int

bib = CDLL("bibliothek.dll")
arraytyp = c_int * 10
a = arraytyp(0,2,5,2,8,1,4,7,3,8)
print("Vorher: ", [i for i in a])
bib.sortiere(a, 10)
print("Nachher: ", [i for i in a])
```

Python

Anbindung an andere Programmiersprachen

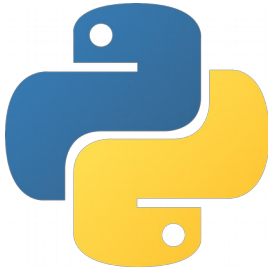


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Datentypen
    - Die Ausgabe dieses Beispielprogramms lautet

```
Vorher: [0, 2, 5, 2, 8, 1, 4, 7, 3, 8]
Nachher: [0, 1, 2, 2, 3, 4, 5, 7, 8, 8]
```

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Im folgenden Beispiel sollen die Parameter der Funktion veclen, wie von der Funktion verlangt, als Gleitkommazahlen übergeben werden

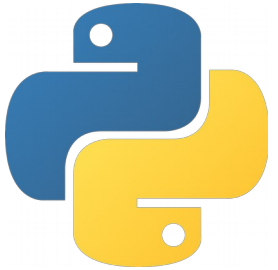
```
from ctypes import CDLL, c_double

bib = CDLL("bibliothek.dll")
print(bib.veclen(c_double(1.5), c_double(2.7), c_double(3.9)))
```

    - Wird dieser Code ausgeführt, so erhalten wir 1077412479
    - als Ergebnis, was nun wirklich nicht der gesuchten Vektorlänge entspricht

Python

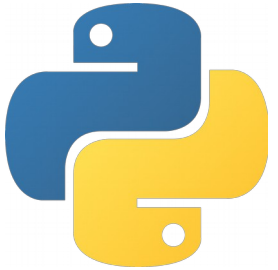
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Der Rückgabewert und die Parameter einer Funktion, also ihre Schnittstelle, sind in C anders als in Python an bestimmte Datentypen gebunden
    - Der eben beschriebene Problemfall resultiert daraus, dass nach dem Laden einer dynamischen Bibliothek von ctypes angenommen wird, dass jede enthaltene C-Funktion eine ganze Zahl zurückgibt, was natürlich in vielen Fällen falsch ist
    - Die eigentliche Gleitkommazahl wurde also aus Unwissenheit als ganze Zahl interpretiert und entsprechend ausgegeben

Python

Anbindung an andere Programmiersprachen



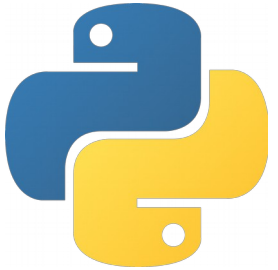
- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Damit dies in Zukunft nicht mehr passiert, können wir über das Attribut restype eines Funktionsobjekts der Datentyp des Rückgabewertes explizit angeben

```
from ctypes import CDLL, c_double

bib = CDLL("bibliothek.dll")
bib.veclen.restype = c_double
print(bib.veclen(c_double(1.5), c_double(2.7), c_double(3.9)))
```

Python

Anbindung an andere Programmiersprachen

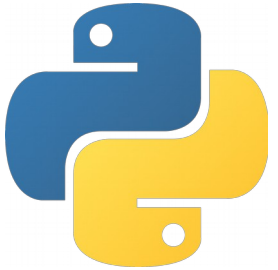


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Bei diesem Beispielprogramm wird der Rückgabewert der C-Funktion korrekt interpretiert, wie die Ausgabe zeigt

      4.97493718553
    - Es wurde angesprochen, dass auch die Parameter einer Funktion in C an einen bestimmten Datentyp gebunden sind
    - Übergäben Sie beispielsweise im obigen Programm statt des Datentyps c_double Instanzen des Datentyps c_float, so würde bereits bei der Parameterübergabe ein Fehler in der Interpretation der Daten passieren, der letztlich in einen falschen Rückgabewert mündet

Python

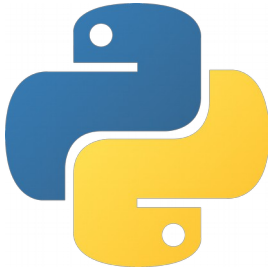
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Python bietet Ihnen an, über das Attribut argtypes die Datentypen der Parameter festzulegen
    - Wenn das gemacht wird, werden übergebene Instanzen eines falschen Datentyps in den korrekten Datentyp konvertiert, oder es wird, wenn dies nicht möglich ist, eine ctypes.ArgumentError-Exception geworfen
    - Im folgenden Programm (auf der nächsten Folie) wird die vollständige Schnittstelle der Funktion veclen vorgegeben

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung

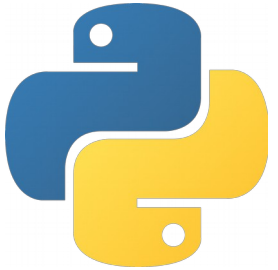
```
from ctypes import CDLL, c_double

bib = CDLL("bibliothek.dll")
bib.veclen.restype = c_double
bib.veclen.argtypes = [c_double, c_double, c_double]
print(bib.veclen(c_double(1.5), c_double(2.7), c_double(3.9)))
```

  - Zwar werden unter Verwendung des Moduls ctypes in vielen Fehlerfällen Exceptions geworfen, beispielsweise wenn zu viele, zu wenige oder die falschen Parameter übereben werden, doch Sie sollten sich immer vergegenwärtigen, dass Sie mit ctypes viele Schutzmechanismen von Python umgehen und möglicherweise direkt im Speicher arbeiten

Python

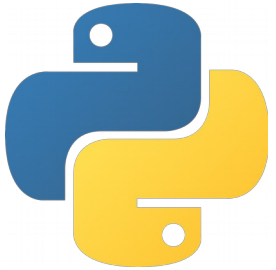
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Wir gehen davon aus, der Name der Bibliothek lautet 'bibliothek.dll'
  - Schnittstellenbeschreibung
    - Es ist also durchaus möglich, unter Verwendung von ctypes den Python-Interpreter zum Absturz zu bringen
    - Und mit »Absturz« ist keine Exception im bisherigen Sinne gemeint, sondern ein tatsächlicher Absturz, beispielsweise durch einen Speicherzugriffsfehler

Python

Anbindung an andere Programmiersprachen

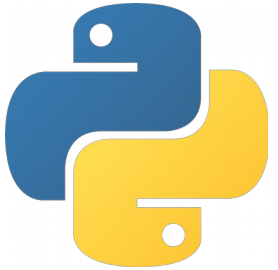


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - An dieser Stelle möchten wir noch einen kurzen Überblick über die wichtigsten im Modul ctypes enthaltenen Funktionen bieten, die Ihnen die Arbeit mit C-Funktionen erleichtern

Funktion	Bedeutung
addressof(obj)	Gibt die Speicheradresse der Instanz *obj* zurück. Für den Parameter *obj* muss eine Instanz eines ctypes-Datentyps übergeben werden.
byref(obj[, offset])	Erzeugt einen Pointer auf die Instanz *obj* eines ctypes-Datentyps. Der zurückgegebene Pointer kann einer C-Funktion übergeben werden.
cast(obj, type)	Bildet den Cast-Operator von C in Python ab.
create_string_buffer(init_or_size[, size])	Erzeugt einen veränderlichen String-Buffer, in den von einer C-Funktion heraus geschrieben werden kann.

Python

Anbindung an andere Programmiersprachen

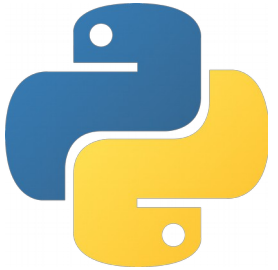


- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - An dieser Stelle möchten wir noch einen kurzen Überblick über die wichtigsten im Modul ctypes enthaltenen Funktionen bieten, die Ihnen die Arbeit mit C-Funktionen erleichtern

Funktion	Bedeutung
create_unicode_buffer(init_or_size[, size])	wie *create_string_buffer*, mit dem Unterschied, dass ein veränderliches Unicode-Array erzeugt wird (Datentyp `c_wchar`)
sizeof(obj_or_type)	Bildet den `sizeof`-Operator von C auf Python ab.
string_at(address[, size])	Gibt den String zurück, der an der Speicheradresse *address* steht.
wstring_at(address[, size])	wie *string_at*, mit dem Unterschied, dass ein Unicode-String zurückgegeben wird

Python

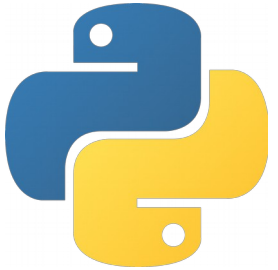
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
  - ctypes.byref(obj[, offset])
    - Diese Funtkion erzeugt einen Pointer auf die Instanz obj eines ctypes-Datentyps
    - Der zurückgegebene Pointer kann einer C-Funktion übergeben werden
    - Ein für offset übergebenes ganzzahliges Offset wird auf den Pointer aufaddiert

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext

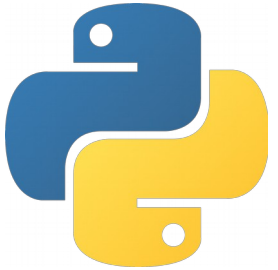
```
>>> import ctypes
>>> bib = ctypes.CDLL("MSVCRT")
```

  - ctypes.byref(obj[, offset])

```
>>> x = ctypes.c_int()
>>> bib.scanf(b"%d", ctypes.byref(x))
27
1                ← Rückgabewert von scanf
>>> x
c_int(27)
```

Python

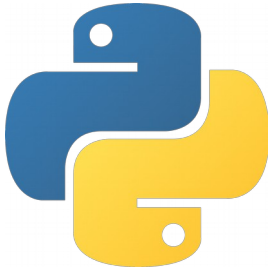
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
  - ctypes.byref(obj[, offset])
    - Im Beispiel wurde die Funktion scanf der Standard C Library verwendet, um eine ganze Zahl vom Benutzer einzulesen
    - Dazu muss ein Pointer auf eine int-Variable übergeben werden
    - Dieser wird über die Funktion byref erzeugt
    - Wie Sie sehen, ist die eingegebene 27 tatsächlich in die Variable x geschrieben worden

Python

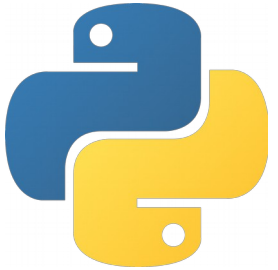
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    ```
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
    ```
  - ctypes.cast(obj, type)
    - Die Funktion cast bildet den Cast-Operator von C in Python ab
    - Die Funktion gibt eine neue Instanz des ctypes-Pointer-Datentyps type zurück, die auf die gleiche Speicherstelle verweist wie obj

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext

```
>>> import ctypes
>>> bib = ctypes.CDLL("MSVCRT")
```

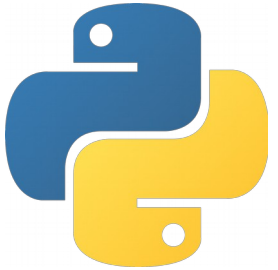
  - ctypes.cast(obj, type)

```
>>> s = ctypes.c_char_p(b"Hallo Welt")
>>> vp = ctypes.cast(s, ctypes.c_void_p)
>>> vp.value
140011580525696
>>> ctypes.cast(vp, ctypes.c_char_p).value
b'Hallo Welt'
```

    - Hier wurde ein char-Pointer in einen void-Pointer und dieser wieder zurück in einen char-Pointer gecastet

Python

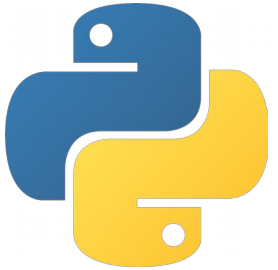
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
  - ctypes.create_string_buffer(init_or_size[, size])
    - Diese Funktion erzeugt einen veränderlichen String-Buffer, in den aus einer C-Funktion heraus geschrieben werden kann
    - Zurückgegeben wird ein Array von c_char-Instanzen
    - Für den ersten Parameter init_or_size können Sie entweder eine ganze Zahl übergeben, die die Länge des zu erzeugenden Buffers enthält, oder einen String, mit dem der Buffer initialisiert werden soll

Python

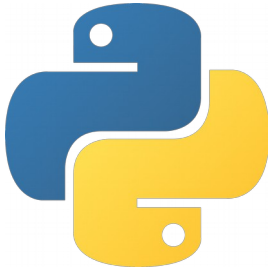
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
  - ctypes.create_string_buffer(init_or_size[, size])
    - Im Falle eines Strings wird der Buffer ein Zeichen größer gemacht als der String lang ist
    - In dieses letzte Zeichen wird der Terminator »\0« geschrieben
    - Wenn Sie für init_or_size einen String übergeben haben, können Sie über den Parameter size die Größe des Buffers festlegen, sofern nicht die Länge des Strings genommen werden soll

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext

```
>>> import ctypes
>>> bib = ctypes.CDLL("MSVCRT")
```

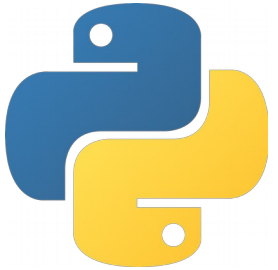
  - ctypes.create_string_buffer(init_or_size[, size])

```
>>> s = ctypes.create_string_buffer(20)
>>> bib.sprintf(s, b"Die Zahl ist: %d", 12)
16
>>> s.value
b'Die Zahl ist: 12'
```

    - Im Beispiel wurde ein String-Buffer der Länge 20 erzeugt und mittels sprintf ein formatierter Text hineingeschrieben

Python

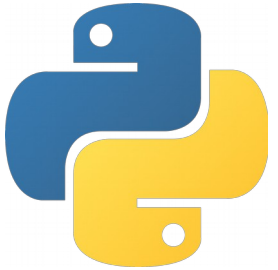
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    ```
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
    ```
  - ctypes.sizeof(obj_or_type)
    - Die Funktion sizeof bildet den sizeof-Operator von C auf Python ab
    - Zurückgegeben wird die Größe der übergebenen Instanz bzw. des übergebenen ctypes-Datentyps in Byte
      ```
      >>> ctypes.sizeof(ctypes.c_int)
      4
      >>> ctypes.sizeof(ctypes.c_char)
      1
      >>> ctypes.sizeof(ctypes.c_double(1.7))
      8
      ```

Python

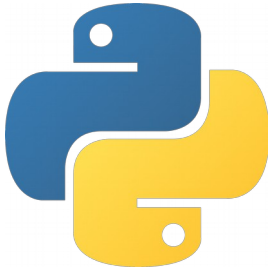
Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Verwendung des Moduls
  - Folgende Beispiele beziehen sich auf folgenden Kontext
    >>> import ctypes
    >>> bib = ctypes.CDLL("MSVCRT")
  - ctypes.string_at(address[, size])
    - Diese Funktion gibt den String zurück, der an der Speicheradresse address steht
    - Sollte der String im Speicher nicht null-terminiert sein, so kann über den Parameter size die genaue Länge des Strings übergeben werden
    - Für address muss eine ganze Zahl übergeben werden, die sinnvollerweise mit addressof geholt und verändert wurde

Python

Anbindung an andere Programmiersprachen



- Ausschließlich an der Sprache C
- Dynamisch ladbare Bibliotheken – ctypes
  - Das soll es soweit gewesen sein, es kann sich hier nur um eine Einführung handeln, welche die Sprache C nicht berücksichtigt
  - Als weiteres Thema ist es möglich selber Extensions zu schreiben, oder aber Python in C/C++-Programmen zu nutzen
  - Da auch hier schon Sprachkenntnisse von C benötigt werden, ist es sinnvoll sich den Teil erst nach einer Einarbeitung in C durch zu arbeiten