Erste Schritte mit Python

Wir wollen mehr über Python erfahren

1.

Syntax & Data Types

Wir beginnen mit den Datentypen (data types)

Werte (Literale) und Datentypen

- Beispiele für Werte in Python:
  - **1.23, 2000**,
  - **'Hello, world!'**,
  - **True**,
  - und mehr
- Der Datentyp bestimmt die Struktur eines Objekts
  - Welche Werte sind erlaubt?
  - Welche Operationen unterstützt das Objekt? Hat es zum Beispiel eine **`Länge`**? Kann ich das Objekt zu einem anderen Objekt hinzufügen?

Objekte in Python

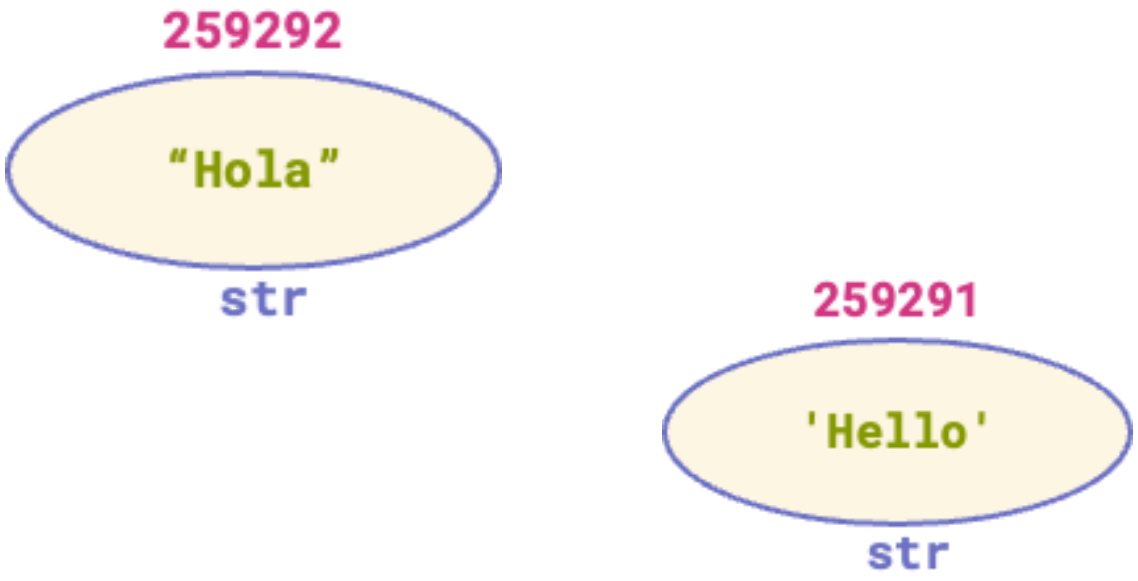
- Darstellung der Daten
- Jedes Objekt in Python hat einen Datentyp, einen Wert und eine Identität

Objekte in Python

- Darstellung der Daten
- Jedes Objekt in Python hat einen Datentyp, einen Wert und eine Identität

Objekte in Python

- Darstellung der Daten
- Jedes Objekt in Python hat einen Datentyp, einen Wert und eine Identität

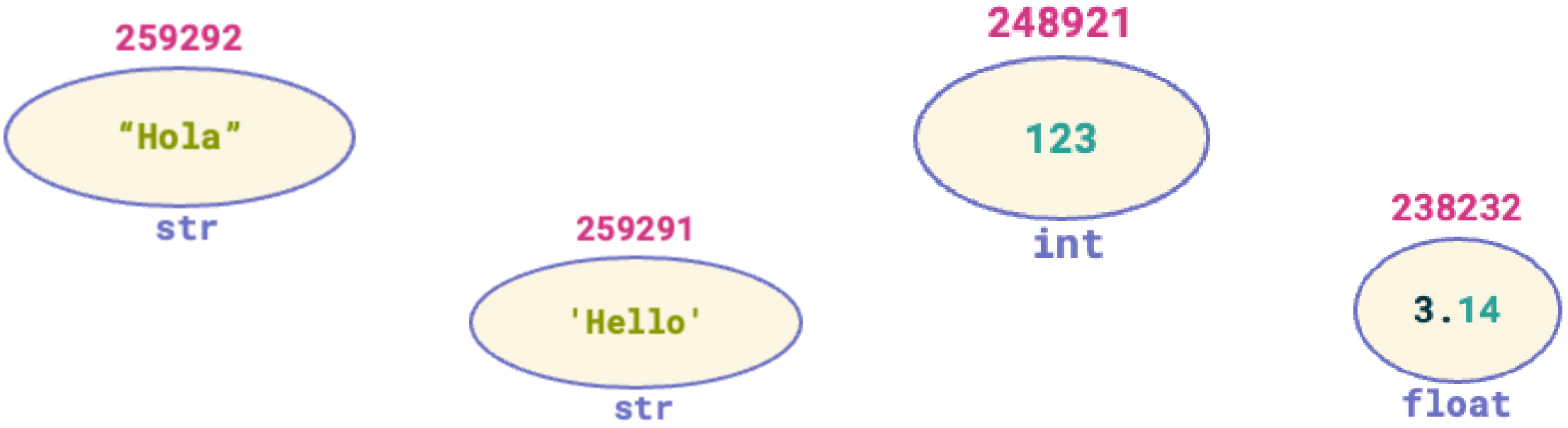


Objekte in Python

- Darstellung der Daten
- Jedes Objekt in Python hat einen Datentyp, einen Wert und eine Identität

Objekte in Python

- Darstellung der Daten
- Jedes Objekt in Python hat einen Datentyp, einen Wert und eine Identität



Datentypen - Zahlen

- Datentyp **`int`**
  - Stellt ganze Zahlen dar
  - Beispiele: 2, 3, -100, 2000

- Datentyp **`float`**
  - Stellt Gleitkommazahlen dar
  - Beispiele: 1.2, 3.14, 100.12, -13.23
    (bitte Punkt beachten)

Datentypen - Zeichenketten

- Datentyp **`str`**
  - Stellt textuelle Daten dar:

```
1 language = "Python"
2
3 course = 'Basics'
4
5 text = """Lorem ipsum dolor sit amet"""
```

Quiz: Welcher Datentyp?

Literal	Data Type
'Hello, there!'	?
'4123.314123'	?
3.14	?
9000	?

Arithmetische Operatoren

Reihenfolge	Operator	Operation	Beispiel
1	(...)	Klammern	(2 * 3) * 7
2	**	Exponential	2 ** 3
3	%	Modulo	10 % 2
4	/	Division	12 / 4
5	*	Multiplikation	3 * 4
6	-	Subtraktion	10 - 8
7	+	Addition	100 + 200

Der Modulo-Operator

- Der Modulo-Operator **`(%)`** wird verwendet, um den Rest einer Division zu erhalten
  - Das (ganzzahlige) Ergebnis von **`15 geteilt durch 4`** ist 12 und der Rest ist 3

```
1 print(15 % 4) # Output: 3
2 print(17 % 12) # Output: 5
3 print(240 % 13) # Output: 6
```

Probieren Sie es aus: Arithmetische Operatoren

- Die Bedeutung des Operators hängt vom Datentyp der Operanden ab.
  - Bitte erstellen Sie eine neue Datei (in Ihrem userfolder)

```
1 nr_1 = 4 * 6 + 4
2
3 text_1 = "Hello, " + "World"
4
5 text_2 = 3 * "Tatütata"
```

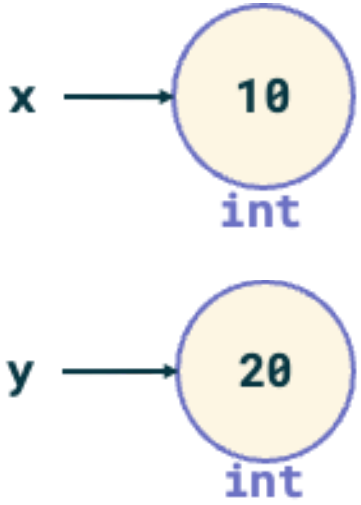
Quiz: Operator Reihenfolge

Ausdruck	Resultat
2 ** 3 + 2 * 3	?
3 * (4 + 3)	?
2 ** (1 + 1) / 4	?

Variablen

- Eine **`Variable`** ist eine Bezeichnung, die auf ein Objekt verweist.
- Auf den Wert des Objekts kann mithilfe von **`Variablennamen`** zugegriffen werden
- Deklaration einer Variablen:

```
x = 10
y = 20
```



- Verwendung:

```
print(x + y)
```

Variablen

- Regeln für die Definition von Variablennamen beachten
- Wählen Sie aussagekräftige Variablennamen
- Mehrfachzuweisungen sind möglich
  - Bitte Komma beachten

```
string1 = "David"
name = "David"
```

```
first_name, last_name = "Monty", "Python"
```

Variablen - Reservierte Schlüsselwörter

- and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, yield

Variablen

- Mit **`print()`** können wir Objekte, Variablen, Literale usw. auf der Konsole ausgeben:

```
1 # 02-getting-started-output.py
2
3 print("Hello, world!")
4
5 nr_1= 10
6 nr_2 = 30
7 print("Result:", nr_1 + nr_2)
```

Quiz: Ausgabe

Wie lautet die Ausgabe, wenn der folgende Code ausgeführt wird?

```
1 nr_1 = 10
2 nr_1 = 20.123
3 nr_1 = "Hallo"
4
5 print(nr_1)
```

Ausdrücke

- Ein **`Ausdruck (expression)`** ist eine Kombination aus **`Werten, Variablen, Funktionsaufrufen und Operatoren`**
- Wichtig: Ein Ausdruck wird immer zu einem **`Wert`** (Literal) ausgewertet.

```
20 + nr_1 + math.sqrt(4)
```

Statement

- Eine Anweisung (statement) ist eine Instruktion, die der Python-Interpreter ausführen kann
- Beispiele für Anweisungen: **`Zuweisung von Variablen, Definition einer Funktion, while-Schleife, for-Schleife`**, usw.
- Mehrere Anweisungen in einer Zeile müssen durch ein **`Semikolon (;)`** getrennt werden

```
name = 'David'
nr_1 = 10; nr_2 = 20
result = nr_1 + nr_2 + 40
```

Quiz-Zeit: Ausgabe

Welchen Wert hat die Variable k in den Skripten **A** und **B** jeweils?

A

```
x = 30; y = 20; z = 10
k = x + 10 - 4 * (y + z)
```

B

```
a = "a"; b = "b"
k = a + b + a + b * 3
```

Benutzereingaben lesen

Wichtig: Die Eingabe wird immer als **`String`** ausgewertet, unabhängig davon, ob der Benutzer eine Zahl eingibt oder einen Text

```
name = input("Your name: ")
print("Hello there,", name)
```

Beispiel: Addition

Führen Sie den folgenden Code in PyCharm aus:

What is the problem with the following code?

```
nr_1 = 100
nr_2 = input("Enter a number: ")
print("Result =", nr_1 + nr_2)
```

Type Conversions (casting)

Umwandlung	**Code**
String → **Integer**	int('1234')
String → **Float**	float('3.14')
Float, Integer → **String**	str(3.14) *or* str(123)

Code Kommentare

- Kommentare werden mit einem Hashtag (#) erstellt
- **`Tip`**: Kommentare sollten nicht den Code erklären, vielmehr sollten sie den Sinn des Codes erklären

```
age = int(input("Age? "))
# Checks if age is greater than or equal to 18
# Only sell alcohol to people of legal age
if age >= 18:
  print("Enjoy your drink!")
else:
  print("Sorry, can't sell you a drink!")
```

Quiz: Reservierte Schlüsselwörter

- Welche davon haben wir bereits genutzt?

  and, **assert**, **break**, **class**, **continue**, **def**, **del**, **elif**, **else**, **except**, **exec**, **finally**, **for**, **from**, **global**, **if**, **import**, **in**, **is**, **lambda**, **not**, **or**, **pass**, **print**, **raise**, **return**, **try**, **while**, **yield**

Übung: Simple Calculation

Schreiben Sie ein Skript simple_calculation.py,
das die folgenden Anweisungen implementiert:

1. **x** hat den Wert **2**
2. **y** hat den Wert **3**
3. **z** ist die Summe von **x** und **y**
4. Verwenden Sie print() um das Ergebnis von (x * y) + z auszugeben

Lösung: Simple Calculation

```
1 x = 2
2 y = 3
3 z = x + y
4 result = (x * y) + z
5
6 print(f"The result is {result}")
```

Übung: Hello There

Schreiben Sie ein Skript hello_there.py, das den Benutzer auffordert, **`seinen Namen einzugeben`** und die folgende Zeile unter Verwendung des eingegebenen Namens ausgibt:

```
Hello, <NAME>
```

<NAME> sollte durch den eingegebenen Namen ersetzt werden.

Lösung: Hello There

```
1 name = input("Please enter your name: ")
2
3 print(f"Hello there, {name}")
```

Übung: Calculator

Schreiben Sie ein Skript simple_calculator.py, das den Benutzer auffordert, zwei Zahlen einzugeben. Das Skript sollte die folgende Zeichenfolge ausgeben, wobei <SUM> die Summe der beiden eingegebenen Zahlen ist:

```
The result is <SUM>
```

Lösung: Calculator

```
1 nr_1 = float(input("Please enter a number (step 1 of 2): "))
2 nr_2 = float(input("Please enter a number (step 2 of 2): "))
3 result_sum = nr_1 + nr_2
4
5 print(f"The result of {nr_1} + {nr_2} is {result_sum}")
```

Kurzer Rückblick

Was haben wir bereits angeschaut?

- Den Python-Interpreter und unserer Einrichtung
- Lernen von Objekten, Werten und Datentypen
- Verstehen von Anweisungen und Ausdrücken
- Text ausgeben und Benutzereingaben lesen

Ende

Das war alles für dieses Kapitel