



Wir waren im Urlaub, möchten am "statistics-calculation"-Branch weiterarbeiten, besitzen aber nicht mehr den aktuellen Stand ... Command-Reihenfolge?

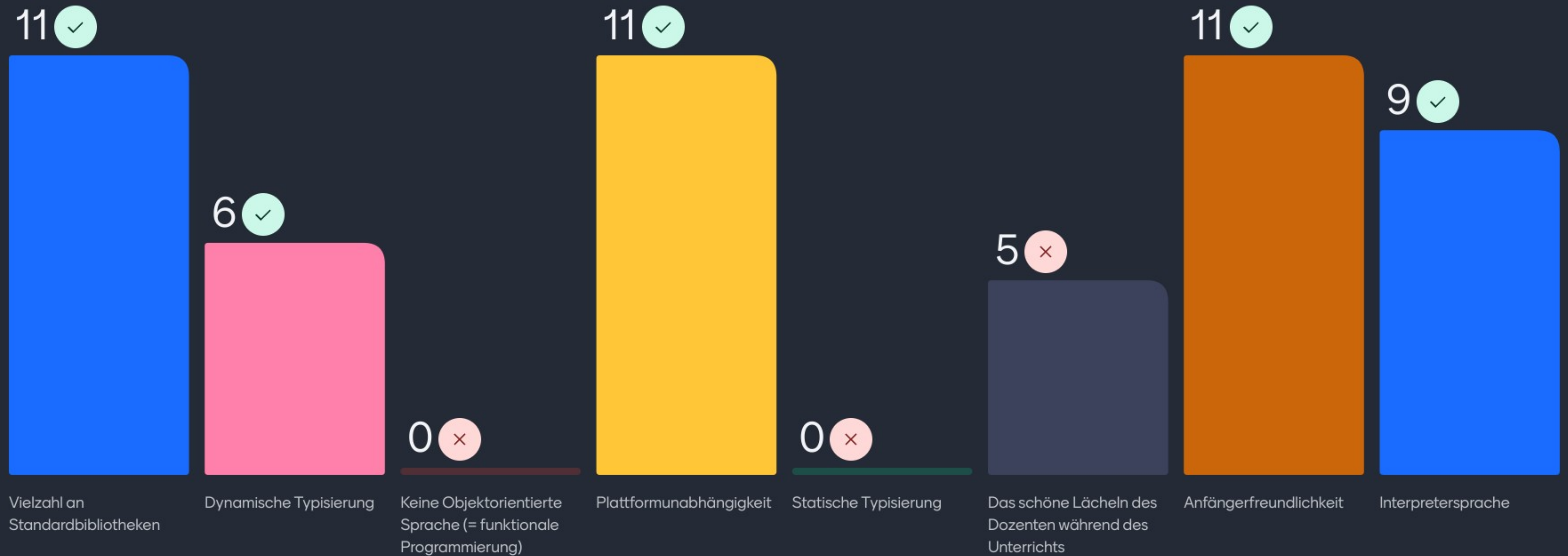
1.  git pull
2.  git checkout
statistics-calculation
3.  git push
4.  git merge main

Was fällt Ihnen alles zu Python ein?

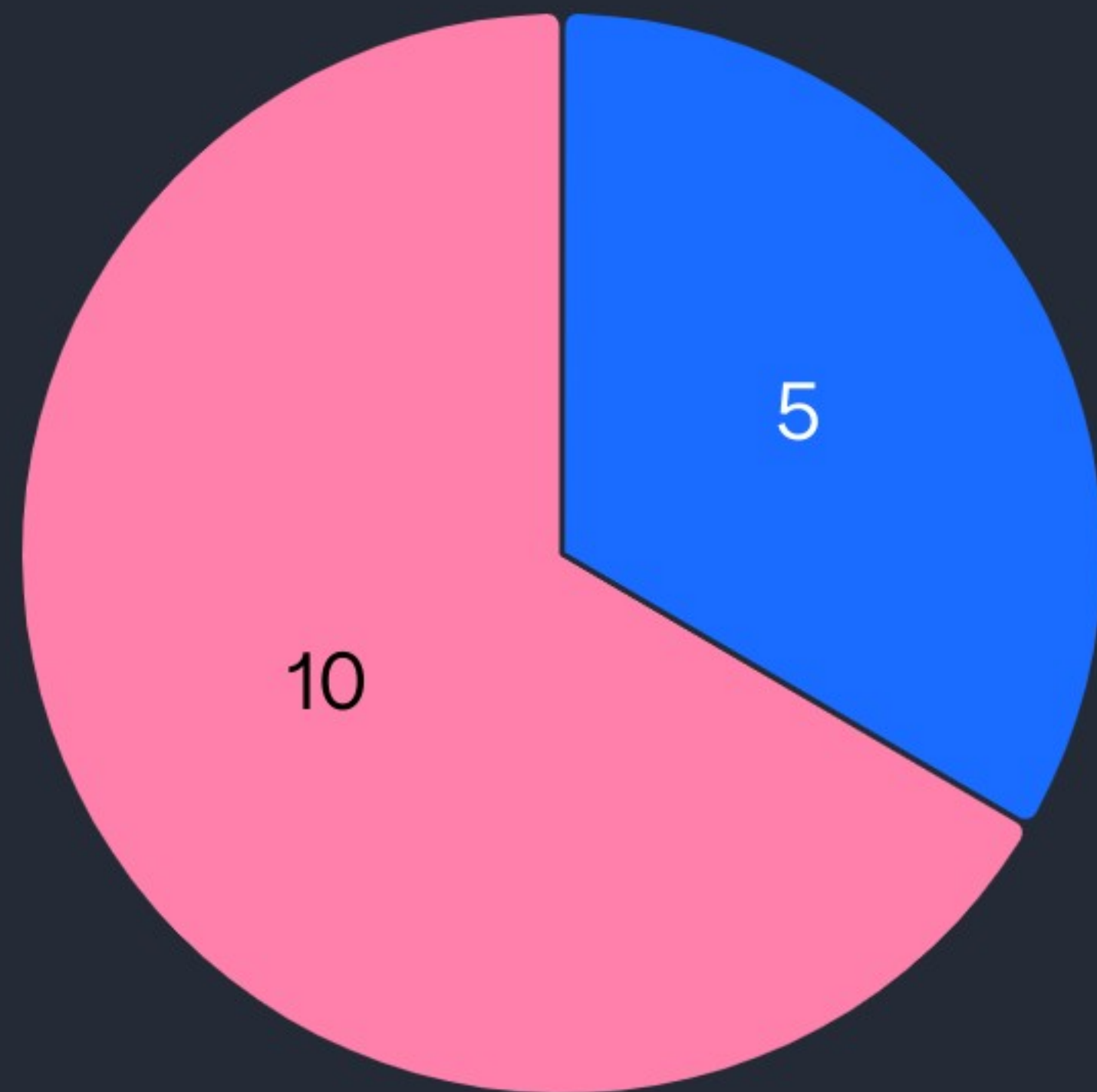
17 Antworten

vielzahl an standardbibli
programmiersprache
scripts putain
schlange
open source
progammiersprache
interpreter sprache
spielerei
ki simpel oop
1991 datascience
rasberry pi
nach monty python benannt
lesbarkeit und einfachheit

Was sind die Hauptmerkmale von Python?



Was sind die Hauptunterschiede zwischen Python und C#?



5

Die Typisierung



10

Kompilierung / Interpretation



0

Keine wesentlichen Unterschiede: Dozent hat einfach mehr Bock auf Python



Ich möchte überprüfen, ob Alex immer noch eine begeisterte Leseratte ist ... Welche der folgenden Bedingungen ist korrekt?

```
1 personen = [{  
2     "name": "Alex",  
3     "alter": 25,  
4     "hobbys": {  
5         "gemeinsam": [  
6             {"bezeichnung": "Fußball", "aktiv": True},  
7             {"bezeichnung": "Schach", "aktiv": False},  
8         ],  
9         "alleine": [  
10            {"bezeichnung": "Zocken", "aktiv": False},  
11            {"bezeichnung": "Lesen", "aktiv": False},  
12            {"bezeichnung": "Traden", "aktiv": True}  
13        ]  
14    }  
15 }]
```



personen[0]["Alex"]["hobbys"]["alleine"]["lesen"]



personen["hobbys"]["alleine"][2]["aktiv"]



personen[0]["hobbys"]["alleine"][1]["aktiv"]

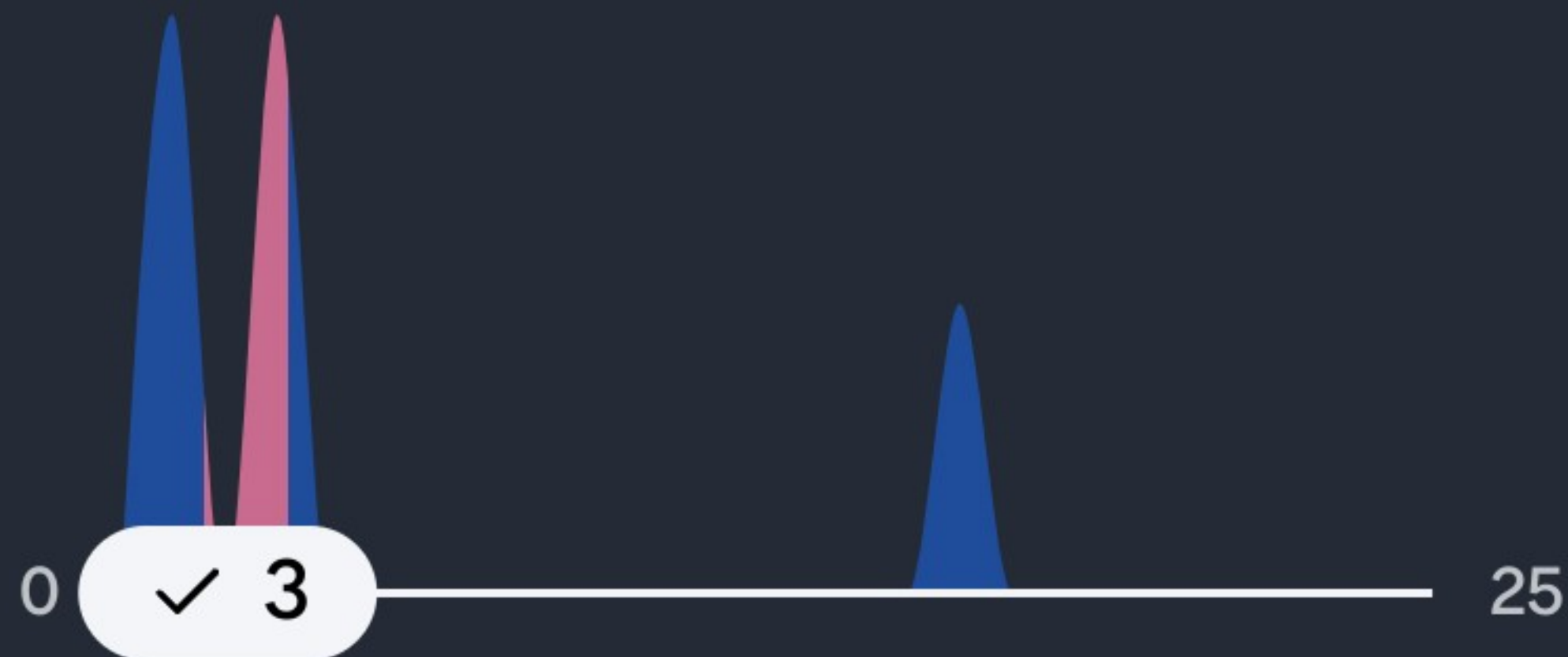


Was wird hier in der Konsole ausgegeben?

```
1  try:
2      result = 0
3      number = 8
4
5      if number % 2 == 0:
6          result = number * 2
7      else:
8          result = number + 3
9
10     print(ergebnis)
11 except Exception as error:
12     print(0)
```



Was wird hier in der Konsole ausgegeben?



```
1  try:
2      result = 0
3      number = 8
4
5      if number % 2 == 0:
6          result = number * 2
7      else:
8          result = number + 3
9
10     except Exception as error:
11         result = 6
12     else:
13         result += 2
14     finally:
15         result = result % 5
16
17     print(result)
```

Welche Bibliothek ist besser geeignet, wenn es darum geht, Daten aus verschiedenen Quellen zu laden, zu strukturieren, zu gruppieren und zu filtern?

12 ✓



Pandas

0 ✗

NumPy

Was ist ein grundlegendes Konzept der Objektorientierten Programmierung (OOP)?

10 ✓



Objekte sind Instanzen von
Klassen

0 ✗

Methoden sind die Variablen
innerhalb einer Klasse

12 ✓



Klassen sind Baupläne für
Objekte

0 ✗

Objekte enthalten nur Daten,
keine Funktionen

Welche der folgenden Aussagen über die OOP gilt als korrekt?

10 ✓



Der Code wird in kleinere, unabhängige Teile (Objekte) aufgeteilt.

0 ✗

Komplexe Programme können mit OOP schwieriger strukturiert werden.

1 ✗



Der Code kann zwar leicht erweitert werden, muss allerdings jedes mal komplett umgeschrieben werden.

10 ✓



Einmal erstellte Objekte können in verschiedenen Programmen oder Teilen des Programms wiederverwendet werden.

Was ist der Zweck des Konstruktors (`__init__` in Python) einer Klasse?

0 

Er definiert das Verhalten der Objekte.

8 

Er wird verwendet, um Objekte der Klasse zu erstellen.

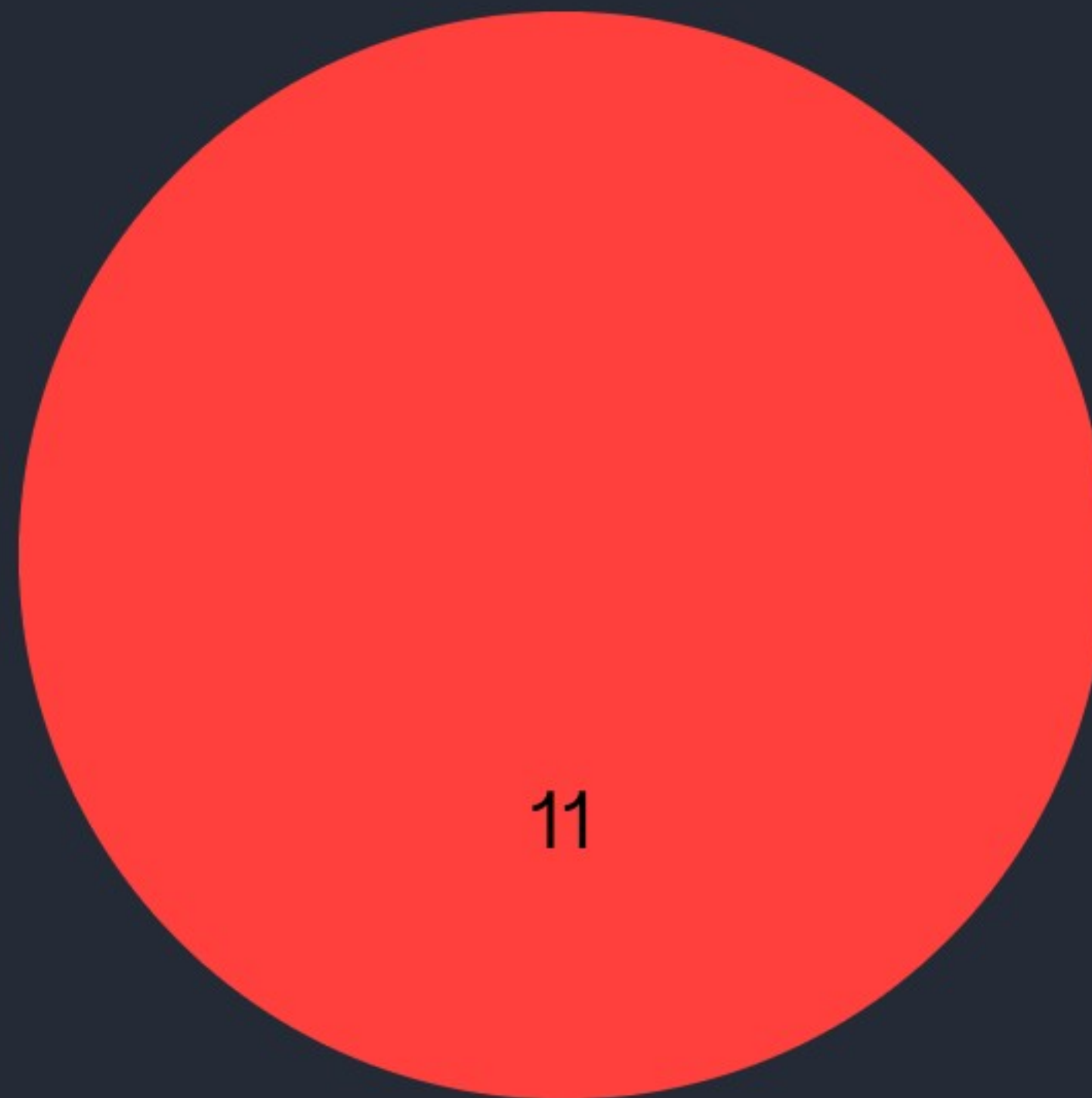
11 

Er initialisiert die Attribute eines Objekts.

0 

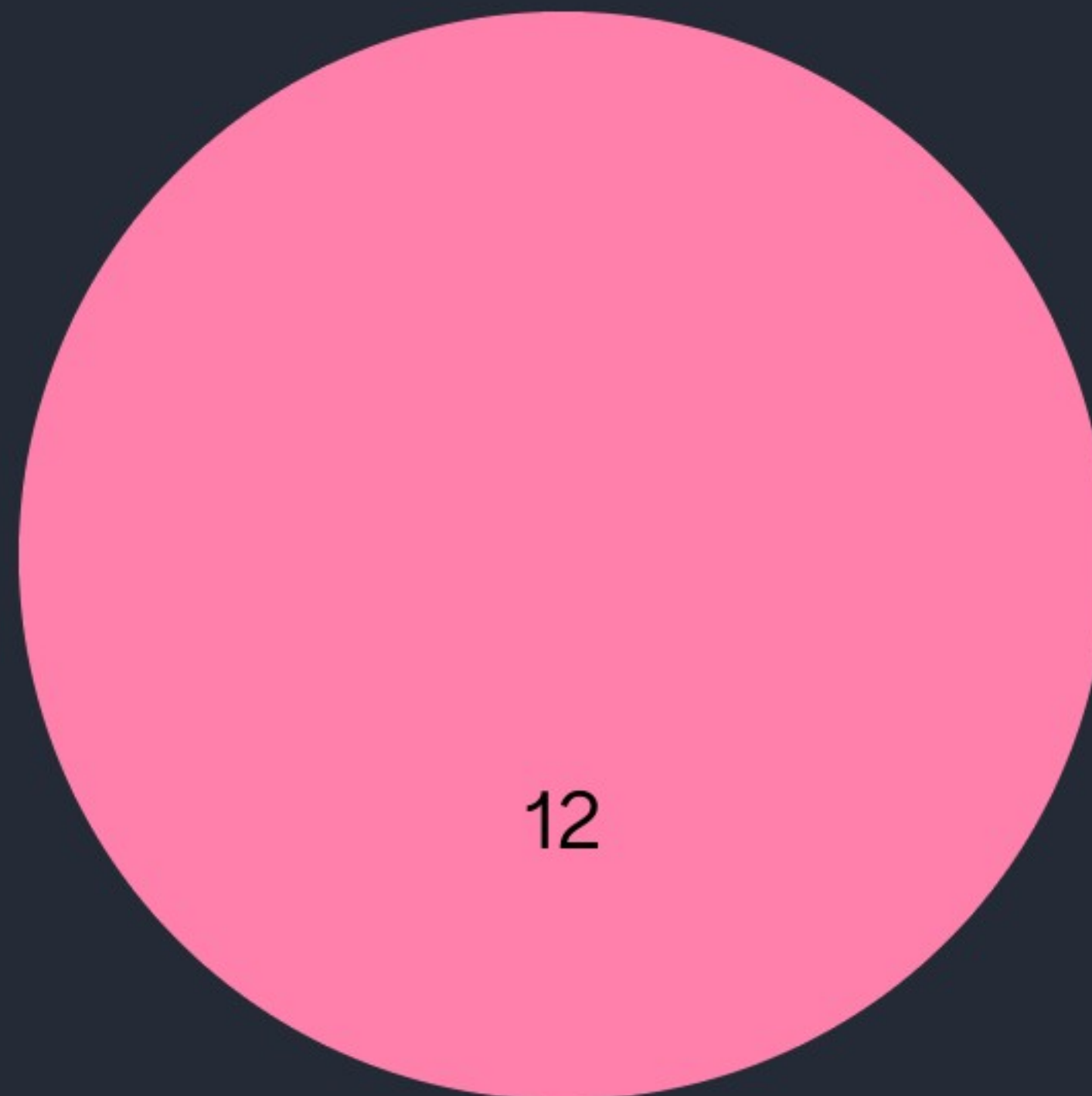
Er wird jedes Mal zuerst aufgerufen, wenn auf ein Attribut / eine Methode zugegriffen wird.

In einer Klasse Auto ... Welcher Kategorie würden Sie "geschwindigkeit" zuordnen?



0	Instanz	✗
0	Methode	✗
11	Attribut	✓
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "beschleunigen" zuordnen?



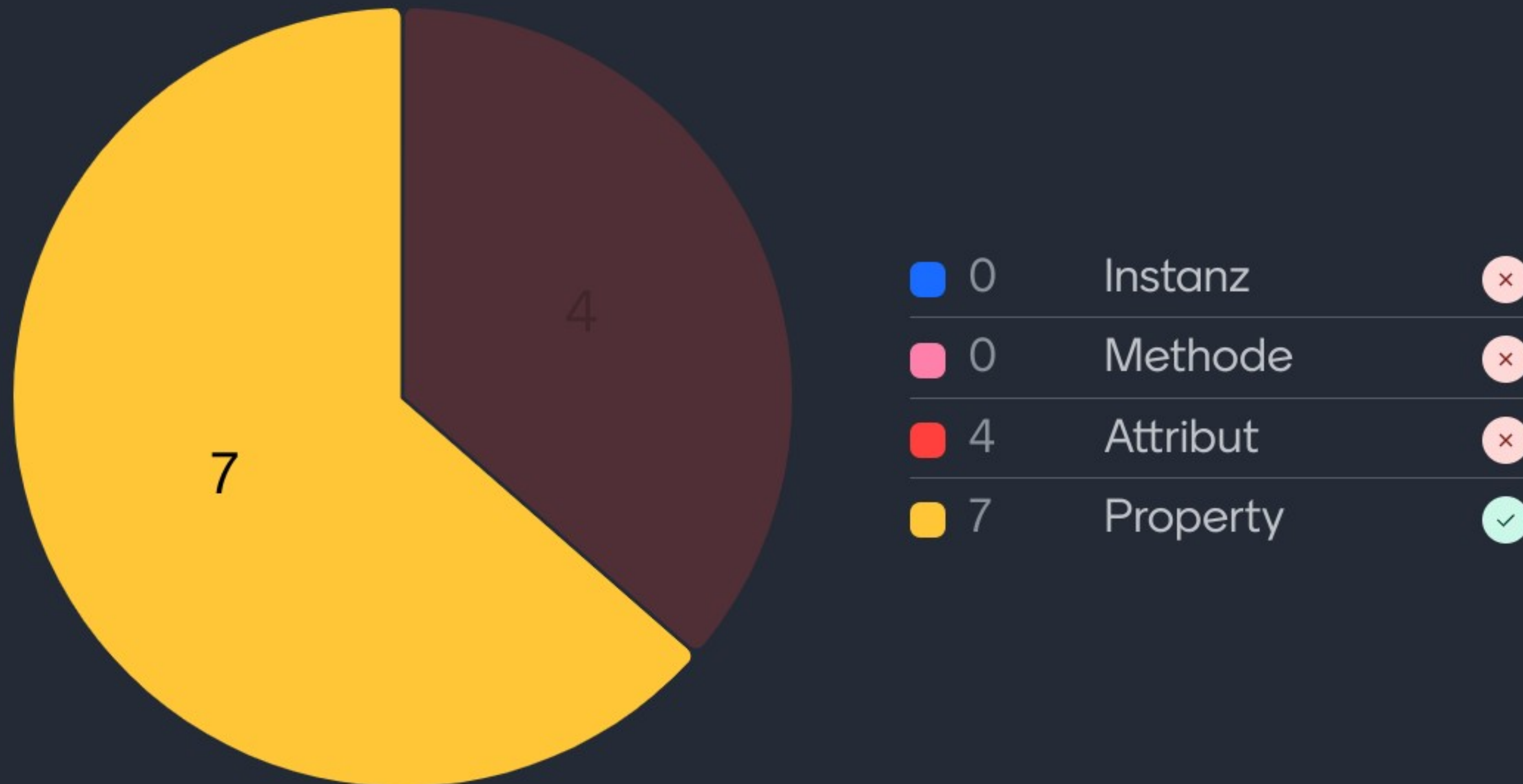
0	Instanz	✗
12	Methode	✓
0	Attribut	✗
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "autoDesDozenten" zuordnen?



11	Instanz	✓
0	Methode	✗
0	Attribut	✗
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "maximaleGeschwindigkeit" zuordnen?

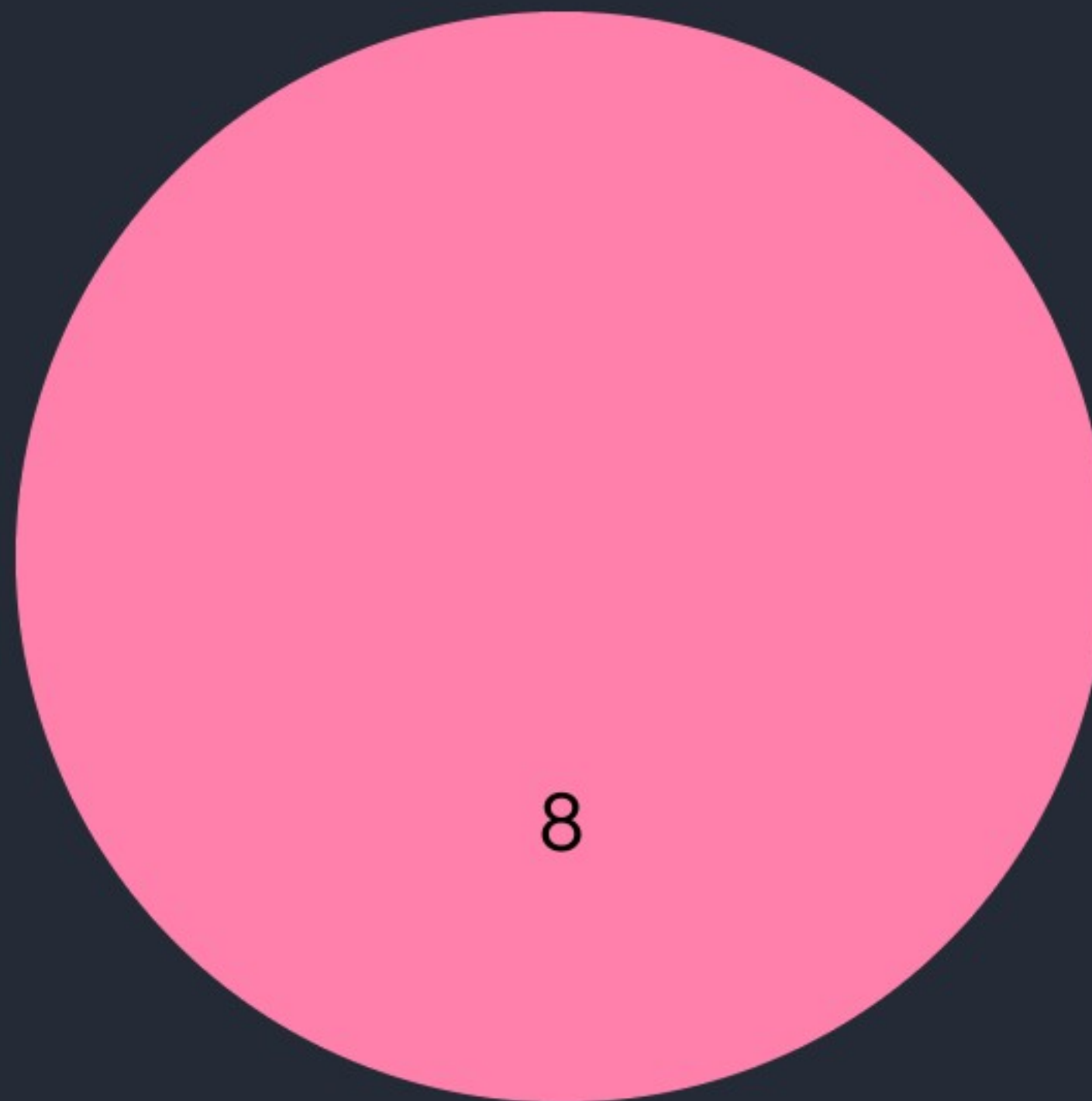


In einer Klasse Auto ... Welcher Kategorie würden Sie "marke" zuordnen?



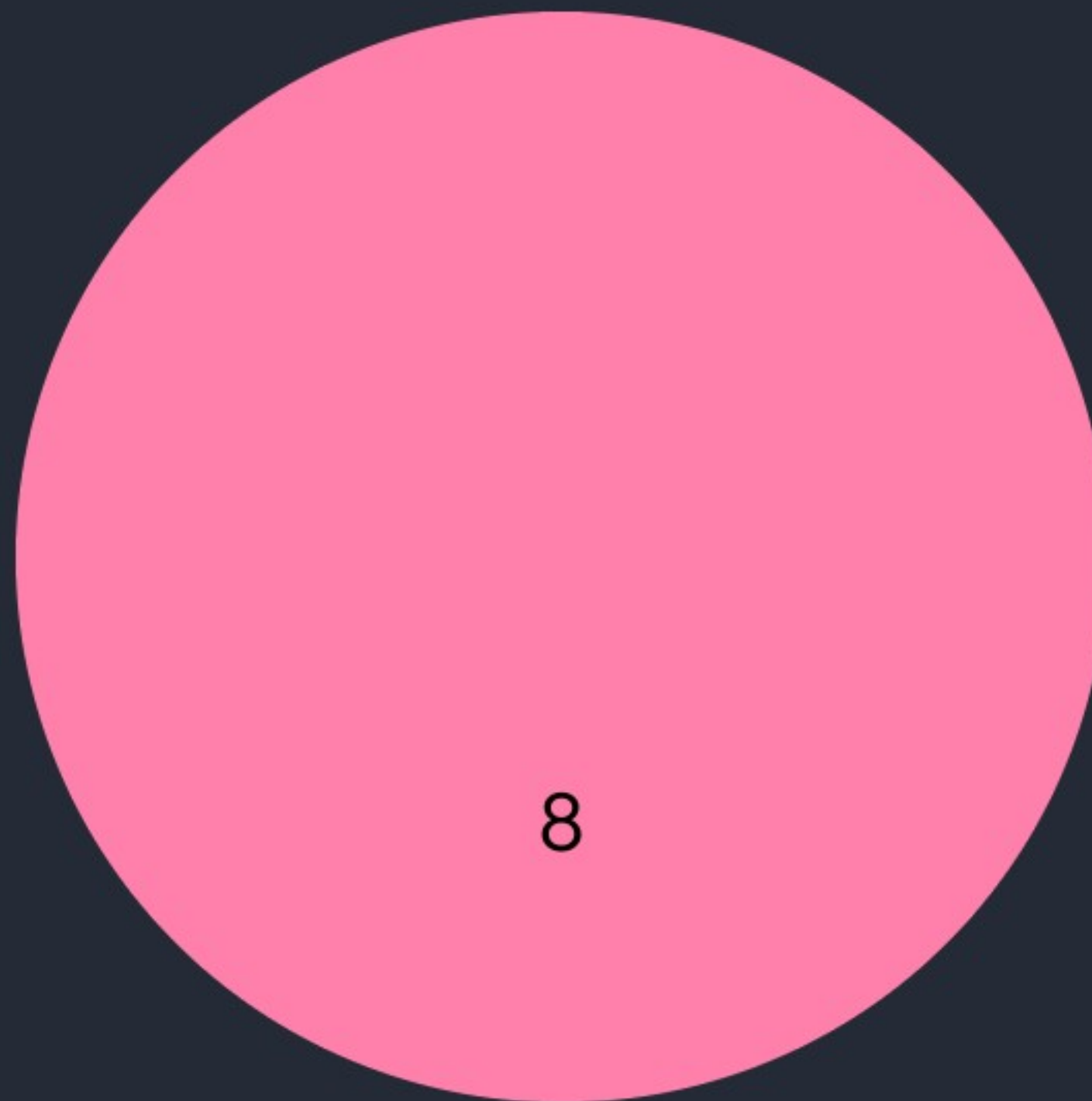
0	Instanz	✗
0	Methode	✗
9	Attribut	✓
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "bremsen" zuordnen?



0	Instanz	✗
8	Methode	✓
0	Attribut	✗
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "fahren" zuordnen?



0	Instanz	✗
8	Methode	✓
0	Attribut	✗
0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "farbe" zuordnen?



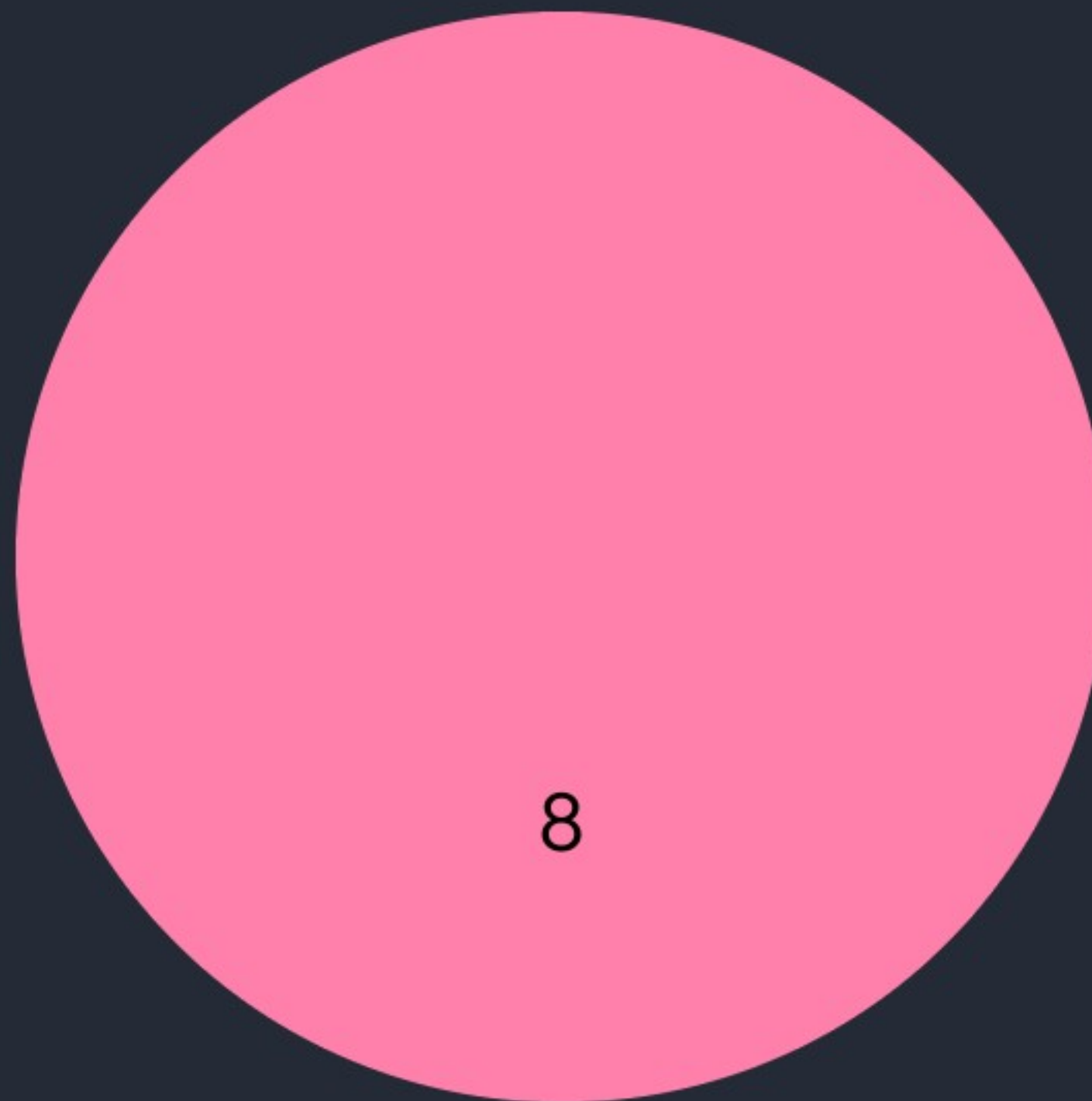
0	Instanz	×
0	Methode	×
9	Attribut	✓
0	Property	×

In einer Klasse Auto ... Welcher Kategorie würden Sie "kaputtesAuto" zuordnen?



■ 9	Instanz	✓
■ 0	Methode	✗
■ 0	Attribut	✗
■ 0	Property	✗

In einer Klasse Auto ... Welcher Kategorie würden Sie "umlackieren" zuordnen?



0	Instanz	✗
8	Methode	✓
0	Attribut	✗
0	Property	✗

Welche der folgenden Aussagen beschreibt das Konzept der Vererbung in der Objektorientierten Programmierung korrekt?

0

Eine Unterklasse kann keine eigenen Methoden oder Attribute definieren.



0

Eine Unterklasse kann nur Methoden, aber keine Attribute von ihrer übergeordneten Klasse erben.



0

Eine Unterklasse kann die Methoden der übergeordneten Klasse nicht überschreiben.



9

Eine Unterklasse kann Methoden und Attribute von ihrer übergeordneten Klasse erben.



Was wird hier in der Konsole ausgegeben?

```
1 class Tier:
2     def bewegen(self):
3         print("Das Tier bewegt sich.")
4     def miauen(self):
5         print("Miau!")
6     def bellen(self):
7         print("Wuff!")
8
9 class Hund(Tier):
10     def bellen(self):
11         print("Wuff!")
12
13 class Katze(Tier):
14     def miauen(self):
15         print("Miau!")
16
17 mein_hund = Hund()
18
19 try:
20     mein_hund.miauen()
21 except:
22     print("Hund kann nicht miauen!")
```

6 ✓

Miau!

4 ✗

Hund kann nicht miauen!

Was beschreibt das Konzept der Abstraktion in der Objektorientierten Programmierung?

0

Abstraktion ermöglicht es, alle Details einer Klasse und ihrer Methoden offenzulegen, damit der Code immer vollständig sichtbar ist.



Abstraktion hilft, nur die relevanten Informationen eines Objekts darzustellen und unwichtige Details zu verbergen.



3

Abstraktion bezieht sich darauf, dass Methoden nur in der Unterklasse verwendet werden können, nicht jedoch in der Oberklasse.



0

Abstraktion ist eine Technik, die es ermöglicht, mehrere Klassen gleichzeitig zu erstellen, ohne deren Unterschiede zu beachten.



Welche Zahl erscheint hier in der Konsole?



```
1  from abc import ABC, abstractmethod
2
3  class Tier(ABC):
4
5      @abstractmethod
6      def bewegen(self):
7          pass
8
9      def essen(self):
10         counter = 5
11         counter += 1
12
13  class Hund(Tier):
14      counter = 3
15      def bewegen(self):
16          self.counter += 1
17
18  class Vogel(Tier):
19      counter = 0
20      def bewegen(self):
21          self.counter += 1
22
23  mein_hund = Hund()
24  mein_vogel = Vogel()
25
26  mein_hund.bewegen()
27  mein_vogel.bewegen()
28  mein_vogel.essen()
29
30  print(mein_vogel.counter)
```

Welche Aussagen zur Kapselung in Python sind korrekt?

11 ✓



Attribute mit einem führenden Unterstrich `_` gelten als geschützt (protected) // Verwendung nur innerhalb der Klasse und ihrer Unterklassen ...

0 ✗

Der führende Unterstrich `_` verhindert komplett den Zugriff auf das Attribut von außerhalb der Klasse...

0 ✗

Kapselung in Python wird automatisch durch den Interpreter erzwungen ...

5 ✓



Mit Kapselung können Details einer Implementierung versteckt und durch öffentliche Methoden gesteuert werden...

10 ✓



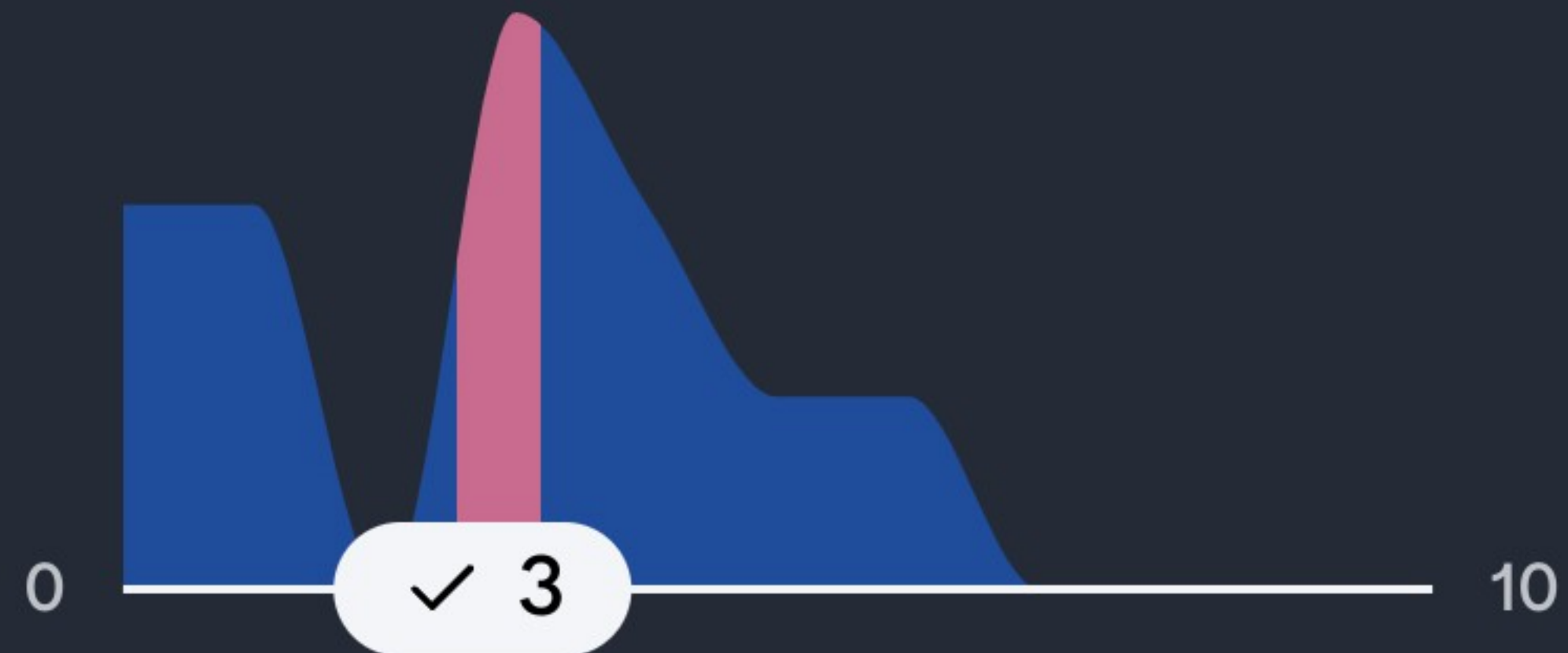
Attribute mit zwei führenden Unterstrichen `__` werden durch Name-Mangling privat gemacht und sind nur innerhalb der Klasse direkt aufrufbar...

Tippen Sie bitte auf die Stelle, an der sich ein Fehler bzgl. der Kapselung eingeschlichen hat!

```
+-----+
|           BankKonto           |
+-----+
| + blz: str                     |
| 5 str                          |
| # bankCode: str                |
+-----+
| + __init__(blz, pin):          |
|   void                        |
| + checkPin(eingabe):           |
|   bool                        |
| + zeigeBankcode():             |
|   str                         |
+-----+
```

Welche Zahl erscheint hier in der Konsole?

```
1 class Smartphone():
2     @property
3     def nummer(self):
4         self.counter += 1
5         return self._nummer
6
7     @nummer.setter
8     def nummer(self, text):
9         self.counter += 1
10        self._nummer = text
11
12    def __init__(self, name, nummer):
13        self.counter = 0
14        self._nummer = nummer
15        if self.isIphone(name):
16            self.counter += 1
17
18    def isIphone(self, name):
19        self.counter += 1
20        return "iphone" in name.lower()
21
22    mein_smartphone = Smartphone("iPhone", "0123456789")
23
24    try:
25        neue_nummer = "987654321"
26        alte_nummer = mein_smartphone._nummer
27        if (neue_nummer != alte_nummer):
28            mein_smartphone._nummer = neue_nummer
29    except:
30        mein_smartphone.counter -= 1
31    else:
32        mein_smartphone.counter += 1
33    finally:
34        print(mein_smartphone.counter)
```



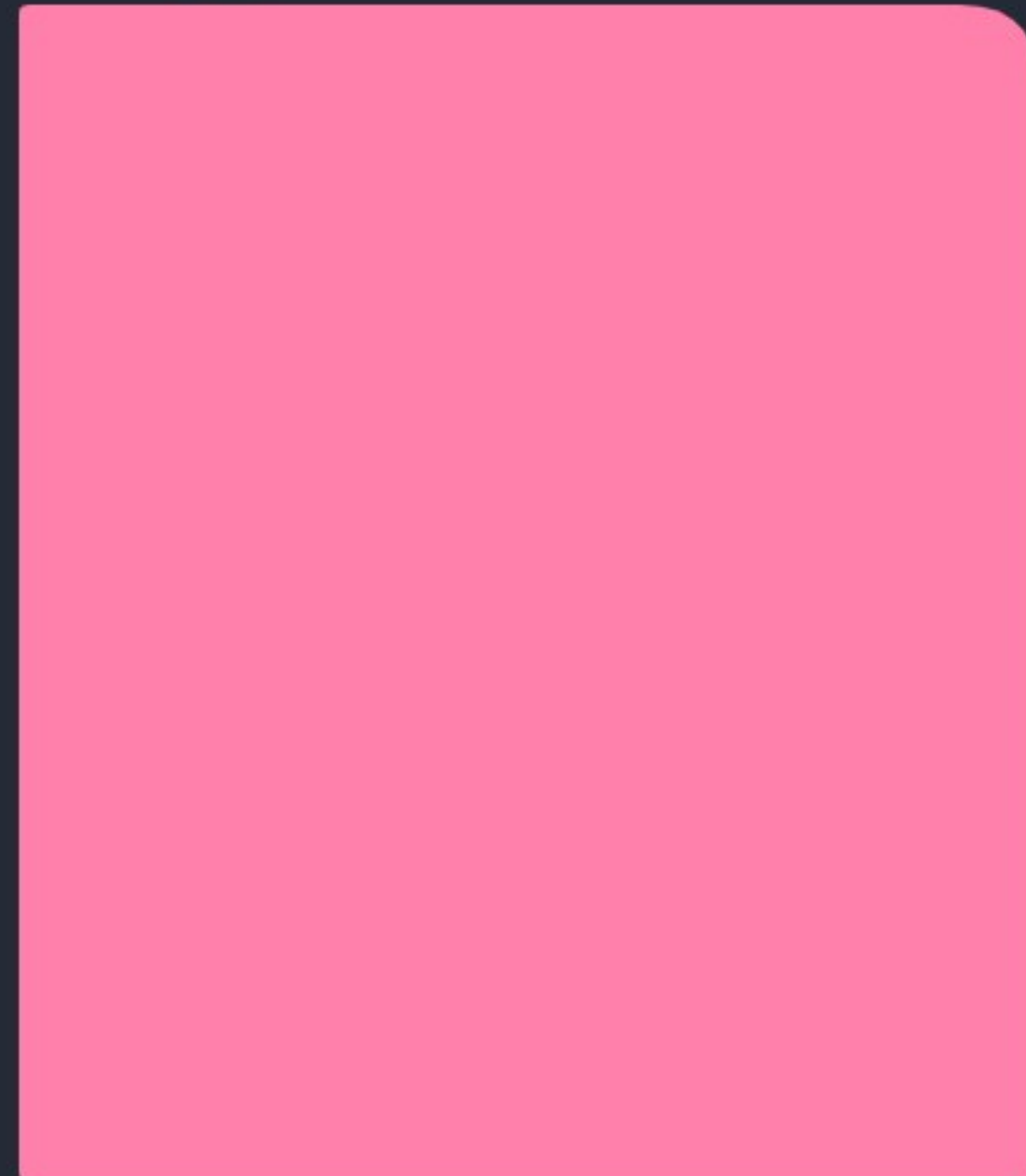
Was beschreibt der Begriff "Polymorphismus" in der Objektorientierten Programmierung?

2 ✗



Polymorphismus bedeutet, dass eine Klasse mehrere Vererbungshierarchien gleichzeitig haben kann.

5 ✓



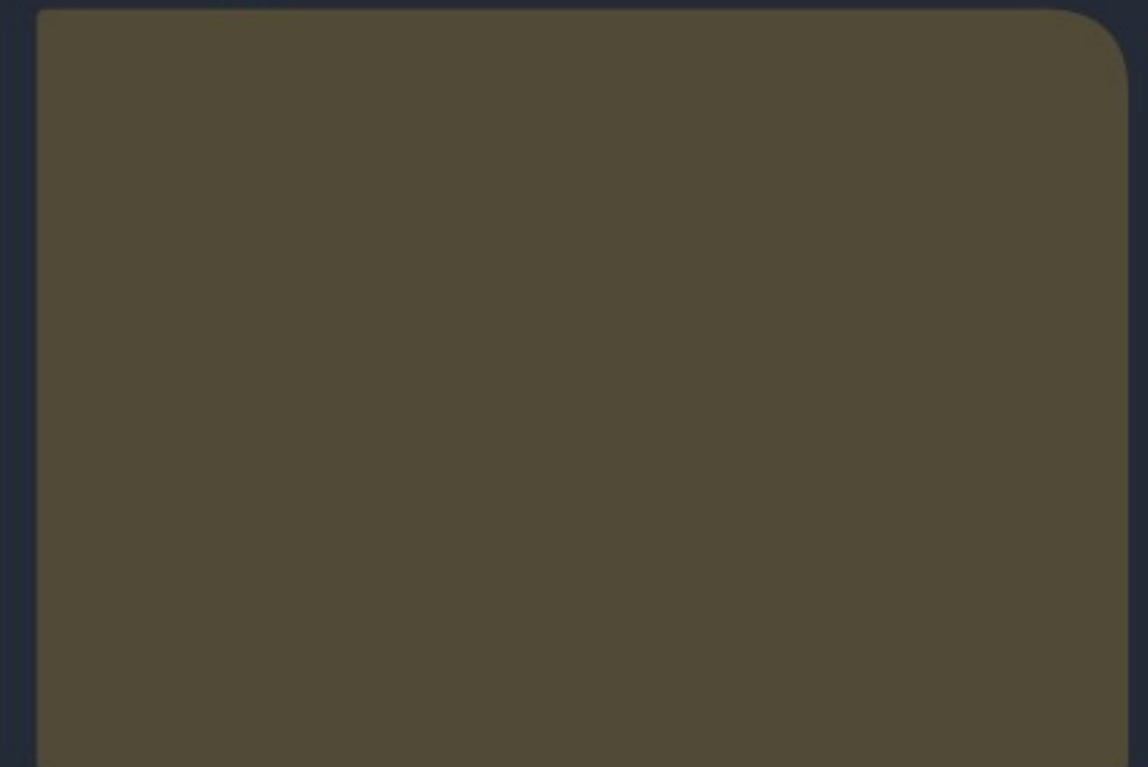
Polymorphismus bedeutet, dass eine Methode in einer abgeleiteten Klasse eine Methode der Basisklasse überschreiben kann.

4 ✓



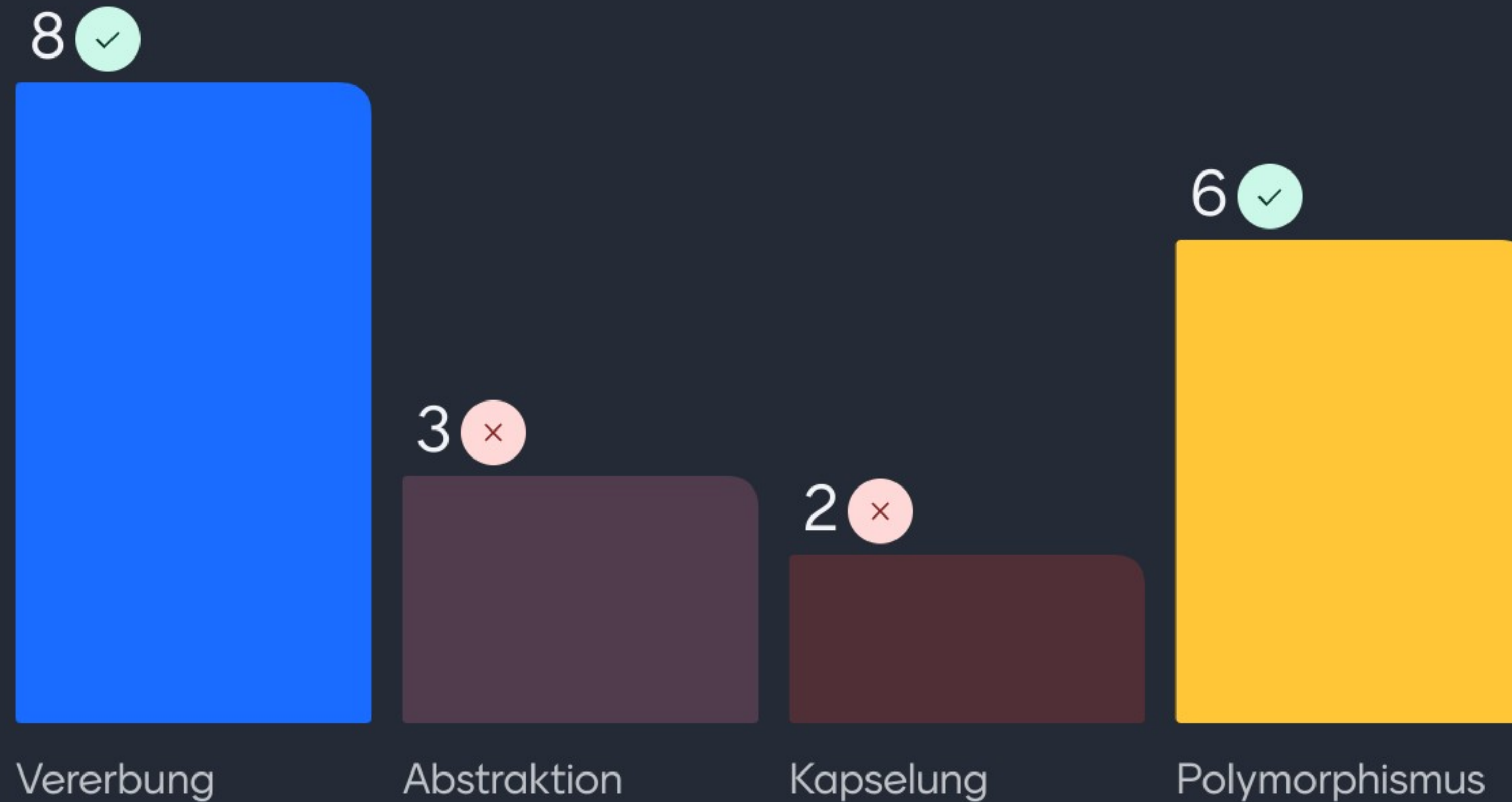
Polymorphismus bedeutet, dass eine Methode denselben Namen in verschiedenen Klassen hat, aber unterschiedliche Parameter akzeptiert.

3 ✗



Polymorphismus bedeutet, dass Objekte einer Klasse zur Laufzeit dynamisch Typen wechseln können.

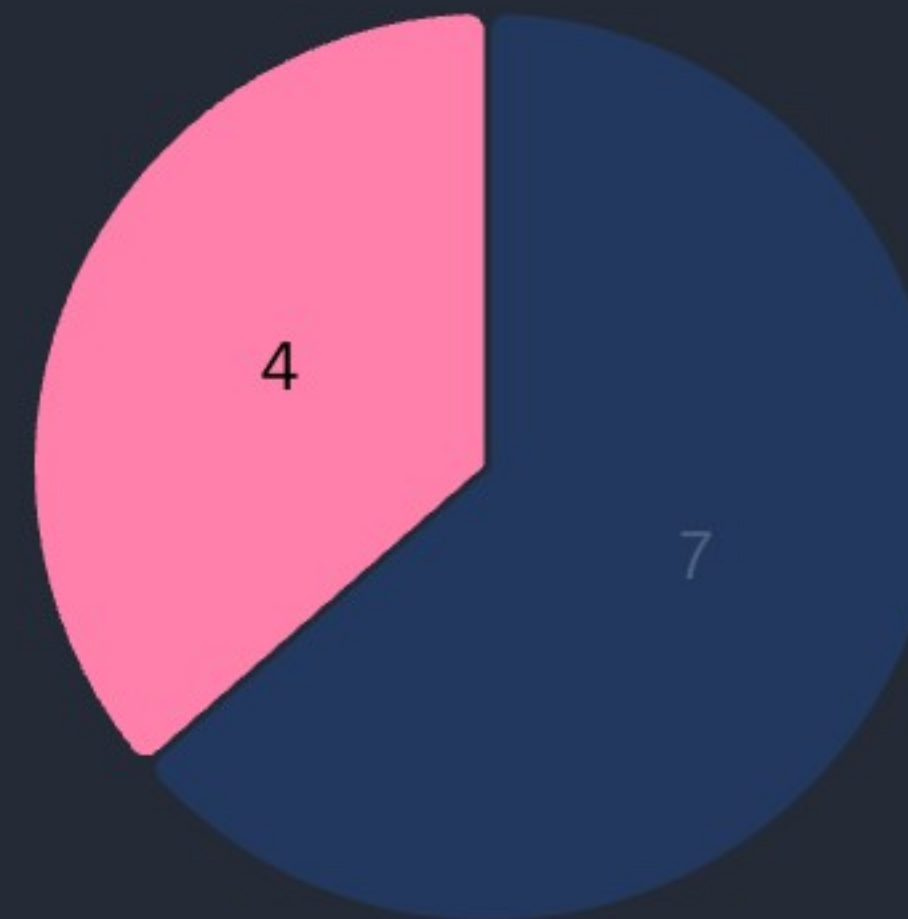
Welche Prinzipien der OOP können der nachfolgenden Grafik entnommen werden?



```
1 from abc import ABC, abstractmethod
2
3 class Market:
4     def trade(self):
5         print("I trade the market!")
6
7 class StockMarket(Market):
8     def trade(self):
9         print("I trade the stock market!")
```

Handelt es sich bei der Beziehung zwischen Mensch und Herz um eine Komposition oder eine Aggregation?

```
1 class Herz:
2     def __init__(self, schlagkraft):
3         self.schlagkraft = schlagkraft
4
5     def schlagen(self):
6         print("Das Herz schlägt mit einer Kraft von", self.schlagkraft, "N.")
7
8 class Mensch:
9     def __init__(self, name, herz):
10        self.name = name
11        self.herz = herz
12
13    def leben(self):
14        print(f"{self.name} lebt.")
15        self.herz.schlagen()
```



7	Aggregation	✗
4	Komposition	✓

Handelt es sich bei der Beziehung zwischen Bibliothek und Buch um eine Komposition oder eine Aggregation?

```
1 class Bibliothek:  
2     def __init__(self, name):  
3         self.name = name  
4         self.buecher = []  
5  
6     def buch_hinzufuegen(self, buch):  
7         self.buecher.append(buch)  
8  
9 class Buch:  
10     def __init__(self, titel, autor):  
11         self.titel = titel  
12         self.autor = autor  
13  
14     def anzeigen(self):  
15         print(f"Das Buch '{self.titel}' wurde von {self.autor} geschrieben.")
```



10

Aggregation



0

Komposition

