



## NotImplementedError

```
NotImplementedError: No id attribute - override get_id
```

Traceback (most recent call last)

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 1536, in **call**

```
return self.wsqi_app(environ, start_response)
```

AA

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 1514, in wsqi\_app

```
response = self.handle_exception(e)
```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 1511, in wsqi\_app

```
response = self.full_dispatch_request()
```

[illegible]

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 919, in full\_dispatch\_request

```
rv = self.handle_user_exception(e)
```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 917, in full\_dispatch\_request

```
rv = self.dispatch_request()
```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

File "C:\Users\Malin Schütz\vsc-workspace\web entwicklung\Helferbaer\venv\Lib\site-packages\flask\app.py", line 902, in dispatch\_request

```
return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args) # type: ignore[no-any-
return]
```

[illegible]

File "C:\Users\Malin Schütz\vscode\workspace\web entwicklung\Helferbaer\app.py", line 78, in  
helfer\_anmelden

```
login_user(user) #Quelle: Flask-Login, Abschnitt "Login Example"
```

**A A A A A A A A A A A A A A A**

File "C:\Users\Malin Schütz\vscode\workspace\web-entwicklung\Helferbaer\venv\Lib\site-packages\flask\_login\utils.py", line 183, in login\_user

```
user_id = getattr(user, current_app.login_manager.id_attribute())
```

[illegible]

File "C:\Users\Malin Schütz\vscode\workspace\webentwicklung\Helferbaer\venv\Lib\site-packages\flask\_login\mixins.py", line 27, in get\_id

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import date, timedelta
```

```

from flask_bcrypt import Bcrypt
from app import app

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///helferbaer.sqlite'

bcrypt = Bcrypt(app)
db = SQLAlchemy()
db.init_app(app)

class User(db.Model, UserMixin): #Quelle UserMixin: Flask-Login, Abschnitt "Your User Class"
    tablename = "user"
    userId = db.Column(db.Integer, primary_key=True, index=True)
    name = db.Column(db.String, nullable=False)
    firstName = db.Column(db.String, nullable=False)
    birthday = db.Column(db.Date, nullable=False)
    email = db.Column(db.String, nullable=False, unique=True)
    password = db.Column(db.String, nullable=False)
    phone = db.Column(db.String)
    role = db.Column(db.String, nullable=False)
    registered_date = db.Column(db.Date, default=date.today())

    jobs_created = db.relationship(
        "Job",
        foreign_keys='Job.kundeld',
        back_populates="kunde"
    )

    jobs_taken = db.relationship(
        "Job",
        foreign_keys='Job.helferId',
        back_populates="helfer"
    )

```

) meine class User sieht bisher so aus. Brauche ich also eine def get\_id?

Nein, du brauchst keine explizite def get\_id(self)-Methode, wenn du UserMixin importierst und korrekt konfigurierst – der Fehler liegt am fehlenden user\_loader und möglicherweise am Primary-Key-Namen. [\[29\]](#) [\[30\]](#)

## Problemursachen

UserMixin sucht standardmäßig nach einem Attribut namens id für get\_id(), aber dein Primary Key heißt userId. Flask-Login wirft den Fehler, weil login\_user die ID nicht findet. Zusätzlich fehlt der @login\_manager.user\_loader-Callback, der den User aus der Session lädt. [\[31\]](#) [\[29\]](#)

## Schnelle Fixes

Füge in app.py (oder wo LoginManager initialisiert wird) hinzu:

```

from flask_login import LoginManager
login_manager = LoginManager()

```

```
login_manager.init_app(app)
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id)) # Lädt per userId (PK)
```

Das passt zu deinem userId-Feld, da SQLAlchemy `query.get()` den Primary Key verwendet. [\[30\]](#) [\[31\]](#)

## User-Modell-Optimierung

- Um Konflikte zu vermeiden, benenne den Primary Key zu id um: `id = db.Column(db.Integer, primary_key=True)`. Passe Relationships an (`foreign_keys='Job.kundeId' → 'Job.kunde_id'`). [\[32\]](#) [\[30\]](#)
- Oder überschreibe explizit:

```
def get_id(self):
    return str(self.userId)
```

Das erzwingt Kompatibilität mit UserMixin. [\[33\]](#) [\[34\]](#)

Nach Migration (`db.create_all()`) und Neustart löst sich der Fehler in `login_user(user)`. Teste mit `current_user.is_authenticated`. [\[29\]](#)

✱

## perfekt danke

Gerne! Viel Erfolg mit Helferbär und der Authentifizierung. [\[44\]](#)

✱

## import click

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import date, timedelta
from flask_bcrypt import Bcrypt
from app import app
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///helferbaer.sqlite'
```

```
bcrypt = Bcrypt(app)
db = SQLAlchemy()
db.init_app(app)
```

```
class User(db.Model, UserMixin): #Quelle UserMixin: Flask-Login, Abschnitt "Your User Class"
    tablename = "user"
    userId = db.Column(db.Integer, primary_key=True, index=True)
```

```

name = db.Column(db.String, nullable=False)
firstName = db.Column(db.String, nullable=False)
birthday = db.Column(db.Date, nullable=False)
email = db.Column(db.String, nullable=False, unique=True)
password = db.Column(db.String, nullable=False)
phone = db.Column(db.String)
role = db.Column(db.String, nullable=False)
registered_date = db.Column(db.Date, default=date.today())

jobs_created = db.relationship(
    "Job",
    foreign_keys='Job.kundeld',
    back_populates="kunde"
)

jobs_taken = db.relationship(
    "Job",
    foreign_keys='Job.helferId',
    back_populates="helfer"
)

def get_id(self): #Quelle: KI Prompt 2
    return str(self.userId)

#lädt Stunden des aktuellen Monats für Stundenkonto Kunde
@property
def current_month_hours_kunde(self):
    current_month = date.today().replace(day=1)
    next_month = (current_month + timedelta(days=32)).replace(day=1)
    jobs = [job for job in self.jobs_created if current_month <= job.date < next_month]

    offene = sum(job.hours for job in jobs if job.statusId == 1)
    gebuchte = sum(job.hours for job in jobs if job.statusId == 2)
    erledigte = sum(job.realHours or job.hours for job in jobs if job.statusId == 3)

    return {
        'offene': offene,
        'gebuchte': gebuchte,
        'erledigte': erledigte,
        'gesamt': offene + gebuchte + erledigte
    }

#lädt Stunden des aktuellen Monats für Stundenkonto Helfer
@property
def current_month_hours_helfer(self):
    current_month = date.today().replace(day=1)
    next_month = (current_month + timedelta(days=32)).replace(day=1)

```

```
jobs = [job for job in self.jobs_taken if current_month <= job.date < next_month]
```

```
gebuchte = sum(job.hours for job in jobs if job.statusId == 2)
```

```
erledigte = sum(job.realHours or job.hours for job in jobs if job.statusId == 3)
```

```
return {  
    'gebuchte': gebuchte,  
    'erledigte': erledigte,  
    'gesamt': gebuchte + erledigte  
}
```

```
#angefragte und erledigte Jobs für Kunden Dashboard
```

```
def get_jobs_by_status_kunde(self):
```

```
    jobs = Job.query.filter(  
        Job.kundId == self.userId  
    ).order_by(Job.date.desc()).all()
```

```
# Trennung: angefragt (offen/gebucht) vs erledigt
```

```
angefragte_jobs = [j for j in jobs if j.statusId in [1,2]]
```

```
erledigte_jobs = [j for j in jobs if j.statusId == 3]
```

```
return {  
    'angefragte_jobs': angefragte_jobs,  
    'erledigte_jobs': erledigte_jobs  
}
```

```
#angefragte und erledigte Jobs für Helfer Dashboard
```

```
def get_jobs_by_status_helfer(self):
```

```
    jobs = Job.query.filter(  
        Job.helferId == self.userId  
    ).order_by(Job.date.desc()).all()
```

```
# Trennung: gebuchte vs erledigt
```

```
gebuchte_jobs = [j for j in jobs if j.statusId == 2]
```

```
erledigte_jobs = [j for j in jobs if j.statusId == 3]
```

```
return {  
    'gebuchte_jobs': gebuchte_jobs,  
    'erledigte_jobs': erledigte_jobs  
}
```

```
#Stundenkonto für Helfer Dashboard
```

```
@property
```

```
def gesamtArbeitsStunden(self):
```

```
    if self.role != 'helfer':  
        return 0
```

```

return sum(
    job.realHours or 0
    for job in self.jobs_taken
    if job.statusId == 3
)

```

warum haben manche funktionen @property decorator und ist das zwingend notwendig?

Der @property-Decorator macht Methoden zu **read-only Attributen**, die wie normale Attribute aufgerufen werden (ohne Klammern). Er ist nicht zwingend notwendig, aber bei deinen Stundenkonto-Funktionen sehr sinnvoll.

## Wann @property verwenden

- `current_month_hours_kunde` und `current_month_hours_helfer`: @property → Zugriff als `user.current_month_hours_kunde` (Dict zurück).
- **Ohne** @property: `user.current_month_hours_kunde()` (Funktionsaufruf).
- **Templates/Jinja**: `{{ current_user.current_month_hours_kunde.offene }}` funktioniert nur mit @property. <sup>[45]</sup>

## Deine Methoden-Analyse

```

@property          # ← Wird wie Attribut behandelt
def current_month_hours_kunde(self):
    return {...}    # → user.current_month_hours_kunde['offene']

def get_jobs_by_status_kunde(self): # ← Normale Methode (kein @property)
    return {...}    # → user.get_jobs_by_status_kunde()['angefragte_jobs']

```

Funktion	@property	Aufruf	Sinnvoll für
<code>current_month_hours_*</code>	✓ Ja	<code>user.zeitkonto</code>	Templates, automatische Berechnung
<code>get_jobs_by_status_*</code>	✗ Nein	<code>user.get_jobs()</code>	Expliziter Aufruf, komplexe Logik
<code>gesamtArbeitsStunden</code>	✓ Ja	<code>user.gesamtStunden</code>	Dashboard-Anzeige

## Empfehlung

- **Behalte** @property bei Stundenkonten – perfekt für Templates.
- **Normale Methoden** für Job-Listen (expliziter, parameterisierbar).
- **Performance**: Cache @property bei schweren Berechnungen mit `functools.lru_cache`.

```

@property
@lru_cache(maxsize=1)
def current_month_hours_kunde(self):
    # Dein Code...

```

Das ist sauberes, idiomatisches Python für dein Helferbär-Dashboard! <sup>[46]</sup>



# gibt es in einer flask dokumentation was zu den property

Nein, die offizielle Flask-Dokumentation erwähnt `@property` nicht explizit, da es ein **reines Python-Feature** ist, kein Flask-spezifisches Konzept.

## Python-Dokumentation (offiziell)

`@property` wird in der Python-Dokumentation beschrieben:

```
https://docs.python.org/3/library/functions.html#property
https://docs.python.org/3/tutorial/classes.html#properties
```

```
class C:
    def __init__(self):
        self._x = None

    @property
    def x(self):
        return self._x

    @x.setter
    def x(self, value):
        self._x = value
```

## Flask-Kontext

Flask-Dokumentation behandelt **andere Decorators** wie `@app.route`, `@login_required`:

```
https://flask.palletsprojects.com/en/stable/patterns/viewdecorators/
```

```
@app.route('/')
@login_required
def index():
    return "Hallo"
```

## Dein User-Modell

```
@property # ← Python built-in, nicht Flask
def current_month_hours_kunde(self):
    return {...} # → user.current_month_hours_kunde['offene']
```

**Fazit:** `@property` kommt aus Python-Docs, nicht Flask-Docs. Perfekt für Jinja-Templates in Flask (`{{ current_user.gesamtStunden }}`).





# gibt es ein tutorial für property?

Ja, hier sind die besten deutschsprachigen Tutorials für `@property`:

## Top-Empfehlungen

### 1. Python-Basics Tutorial (schriftlich)

<https://python-basics-tutorial.readthedocs.io/de/24.3.0/oop/property.html>

- Perfekte Erklärung mit Getter/Setter-Beispielen
- Beispiele: `s1.length = 2` → `@length.setter`
- **Ideal für Helferbär**: Zeigt Klassen-Attribute wie deine Stundenkonten

### 2. YouTube-Videos (praktisch)

- "Property Decorator: GETTER, SETTER und DELETER" (11 Min)  
<https://www.youtube.com/watch?v=PgekwucmID8>
- "Getters & Settings (@Property Decorator)" (OOP-Serie)  
<https://www.youtube.com/watch?v=CWd2USTgc8Q>

## Praktisches Beispiel (für dein User-Modell)

```
class User(db.Model, UserMixin):
    # ... deine Felder ...

    @property
    def full_name(self): # ← Getter
        return f"{self.firstName} {self.name}"

    @full_name.setter    # ← Setter
    def full_name(self, value):
        names = value.split(" ", 1)
        self.firstName, self.name = names[0], names[1]

# Nutzung:
user.full_name # → "Max Mustermann" (ruft Getter)
user.full_name = "Anna Beispiel" # ruft Setter
```

**Start mit dem Python-Basics-Link** – passt perfekt zu deinem Flask/SQLAlchemy-Code-Stil!



# könnte ich auc hohne property ein dictionary zurückgeben?

Ja, **absolut!** Du kannst problemlos ein Dictionary ohne `@property` zurückgeben.

## Ohne `@property` (normale Methode)

```
def current_month_hours_kunde(self): # ← Kein @property
    # Dein Code identisch...
    return {'offene': 5, 'gebuchte': 3, 'erledigte': 8, 'gesamt': 16}

# Aufruf:
user.current_month_hours_kunde() # ← Mit Klammern!
result = user.current_month_hours_kunde()
print(result['offene']) # 5
```

## Mit `@property` (Attribut)

```
@property
def current_month_hours_kunde(self):
    # Gleicher Code...
    return {'offene': 5, 'gebuchte': 3, 'erledigte': 8, 'gesamt': 16}

# Aufruf:
user.current_month_hours_kunde # ← Ohne Klammern!
print(user.current_month_hours_kunde['offene']) # 5
```

## Wann welches verwenden?

Kontext	@property	Normale Methode
Templates	✓ {{ current_user.stunden['offene'] }}	✗ {{ current_user.stunden()['offene'] }}
Python-Code	✓ user.stunden['gesamt']	✓ user.stunden()['gesamt']
Parameter	✗	✓ user.stunden(2024, 1)
Performance	△ Neu berechnet bei jedem Zugriff	✓ Kann gecacht werden

## Deine beste Wahl für Helferbär

```
# Für Templates (Dashboard) → @property behalten
@property
def current_month_hours_kunde(self):
    return {...}

# Für komplexe Abfragen → normale Methode
```

```
def get_hours_for_month(self, year, month):  
    return {...} # Mit Parametern!
```

**Fazit:** Dictionary-Rückgabe klappt immer. @property nur für **häufige Template-Zugriffe** sinnvoll.

✱✱

1. <https://stackoverflow.com/questions/73977948/flask-login-using-usermixin-no-id-attribute-error-despite-being-implemented>
2. <https://flask-login.readthedocs.io>
3. <https://www.freecodecamp.org/news/how-to-authenticate-users-in-flask/>
4. <https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login>
5. [https://flask-login.readthedocs.io/en/0.4.0/\\_modules/flask\\_login/utils.html](https://flask-login.readthedocs.io/en/0.4.0/_modules/flask_login/utils.html)
6. [https://flask-login.readthedocs.io/en/0.6.2/\\_modules/flask\\_login/mixins/](https://flask-login.readthedocs.io/en/0.6.2/_modules/flask_login/mixins/)
7. <https://faun.pub/python-flask-tutorial-part-5-d3f0be695dac>
8. <https://github.com/techwithtim/Flask-Web-App-Tutorial/issues/84>
9. [https://www.reddit.com/r/flask/comments/as1jiz/af\\_user\\_object\\_has\\_no\\_attribute\\_is\\_active/](https://www.reddit.com/r/flask/comments/as1jiz/af_user_object_has_no_attribute_is_active/)
10. <https://gist.github.com/shrayasr/10004888>
11. <https://www.freekb.net/Article?id=4677>
12. <https://github.com/pallets-eco/flask-security-3.0/issues/472>
13. <https://flask123.sinaapp.com/article/13/>
14. <https://github.com/mattupstate/flask-security/issues/472>
15. <https://www.mitchellbusby.com/2016/08/21/tutorial-roll-your-own-user-authorisation-management-with-flask-login/>
16. <https://stackoverflow.com/questions/10695093/how-to-implement-user-loader-callback-in-flask-login>
17. [https://flask-user.readthedocs.io/en/latest/data\\_models.html](https://flask-user.readthedocs.io/en/latest/data_models.html)
18. [https://www.reddit.com/r/flask/comments/1bifyui/dont\\_understand\\_login\\_manager\\_and\\_user\\_loader/](https://www.reddit.com/r/flask/comments/1bifyui/dont_understand_login_manager_and_user_loader/)
19. <https://www.freecodecamp.org/news/how-to-setup-user-authentication-in-flask/>
20. [https://www.reddit.com/r/flask/comments/1adr569/use\\_flasklogin\\_without\\_sql\\_sqlalchemy/](https://www.reddit.com/r/flask/comments/1adr569/use_flasklogin_without_sql_sqlalchemy/)
21. <https://stackoverflow.com/questions/12075535/flask-login-cant-understand-how-it-works>
22. [https://github.com/maxcountryman/flask-login/blob/main/src/flask\\_login/utils.py](https://github.com/maxcountryman/flask-login/blob/main/src/flask_login/utils.py)
23. <https://hackersandslackers.com/flask-login-user-authentication/>
24. [https://www.reddit.com/r/flask/comments/137x94/can\\_someone\\_help\\_me\\_understand\\_user\\_loader/](https://www.reddit.com/r/flask/comments/137x94/can_someone_help_me_understand_user_loader/)
25. <https://stackoverflow.com/questions/29199624/flask-login-errors-with-specific-attributes>
26. <https://stackoverflow.org.cn/questions/57893830>
27. <https://www.appsloveworld.com/coding/flask/72/flask-login-get-id-override-prevents-other-methods-from-working>
28. <https://stackoverflow.com/questions/67964061/flask-login-get-id-override-prevents-other-methods-from-working>
29. <https://flask-login.readthedocs.io>

30. <https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login>
31. <https://stackoverflow.com/questions/73977948/flask-login-using-usermixin-no-id-attribute-error-despite-being-implemented>
32. <https://www.freecodecamp.org/news/how-to-setup-user-authentication-in-flask/>
33. <https://github.com/mattupstate/flask-security/issues/472>
34. <https://stackoverflow.com/questions/34951134/flask-login-only-works-if-get-id-returns-self-email>
35. <https://www.mitchellbusby.com/2016/08/21/tutorial-roll-your-own-user-authorisation-management-with-flask-login/>
36. <https://www.freekb.net/Article?id=4545>
37. <https://stackoverflow.com/questions/50138426/how-to-get-name-of-logged-user-flask-sqlalchemy>
38. <https://groups.google.com/g/google-web-toolkit/c/JVaYlul7v6Q>
39. <https://hackersandslackers.com/flask-login-user-authentication/>
40. [https://flask-user.readthedocs.io/en/latest/data\\_models.html](https://flask-user.readthedocs.io/en/latest/data_models.html)
41. <https://overiq.com/flask-101/authentication-in-flask/index.html>
42. [https://flask-login.readthedocs.io/\\_/downloads/en/0.4.0/pdf/](https://flask-login.readthedocs.io/_/downloads/en/0.4.0/pdf/)
43. <https://www.youtube.com/watch?v=2dEM-s3mRLE>
44. <https://flask-login.readthedocs.io>
45. <https://flask-login.readthedocs.io>
46. <https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login>
47. <https://stackoverflow.com/questions/10695093/how-to-implement-user-loader-callback-in-flask-login>
48. [https://www.reddit.com/r/flask/comments/tu060b/implementing\\_different\\_classes\\_for\\_flask\\_login/](https://www.reddit.com/r/flask/comments/tu060b/implementing_different_classes_for_flask_login/)
49. <https://flask-login.readthedocs.io/en/latest/>
50. [https://www.reddit.com/r/flask/comments/1bifyui/dont\\_understand\\_login\\_manager\\_and\\_user\\_loader/](https://www.reddit.com/r/flask/comments/1bifyui/dont_understand_login_manager_and_user_loader/)
51. [https://www.reddit.com/r/flask/comments/l37x94/can\\_someone\\_help\\_me\\_understand\\_user\\_loader/](https://www.reddit.com/r/flask/comments/l37x94/can_someone_help_me_understand_user_loader/)
52. [https://www.reddit.com/r/flask/comments/sg2y2n/2\\_type\\_user\\_login\\_using\\_flasklogin/](https://www.reddit.com/r/flask/comments/sg2y2n/2_type_user_login_using_flasklogin/)
53. [https://www.reddit.com/r/learnpython/comments/h9m9ey/flasklogin\\_missing\\_user\\_loader/](https://www.reddit.com/r/learnpython/comments/h9m9ey/flasklogin_missing_user_loader/)
54. <https://gist.github.com/leongjinqwen/a205cbe8185d8c83f9d300cc6c8634f1>
55. <https://flask.palletsprojects.com/en/stable/patterns/viewdecorators/>
56. <https://flask-restplus.readthedocs.io/en/0.7.1/documenting.html>
57. <https://flask.palletsprojects.com/en/stable/api/>
58. <https://flask-restplus.readthedocs.io/en/stable/swagger.html>
59. <https://book.pythontips.com/en/latest/decorators.html>
60. <https://flaskforstartups.com/tutorials/function-decorators-and-routes/>
61. <https://stackoverflow.com/questions/17330160/how-does-the-property-decorator-work-in-python>
62. <https://flask-plugin.pages.dev/api>
63. <https://dev.to/the1kimk/function-decorators-in-python-understanding-property-getter-and-setter-methods-3a8e>
64. <https://flask.palletsprojects.com/en/stable/blueprints/>

65. <https://www.youtube.com/watch?v=Akaoc7BMrfQ>
66. <https://python-basics-tutorial.readthedocs.io/de/24.3.0/oop/property.html>
67. <https://www.youtube.com/watch?v=PgekwucmID8>
68. <https://www.ionos.de/digitalguide/websites/web-entwicklung/python-property/>
69. <https://de.w3docs.com/snippets/python/wie-funktioniert-der-property-decorator-in-python.html>
70. <https://www.youtube.com/watch?v=64VsNQjlx2c>
71. [https://www.python-kurs.eu/python3\\_dekorateure.php](https://www.python-kurs.eu/python3_dekorateure.php)
72. <https://www.ionos.de/digitalguide/websites/web-entwicklung/python-decorators/>
73. <https://www.youtube.com/watch?v=7wKBS831vkM>
74. <https://www.python-forum.de/viewtopic.php?t=11848>
75. [https://www.reddit.com/r/learnpython/comments/16msif3/returning\\_keys\\_with\\_matching\\_values\\_when\\_the\\_keys/](https://www.reddit.com/r/learnpython/comments/16msif3/returning_keys_with_matching_values_when_the_keys/)
76. <https://marketmix.com/de/python-dictionary-cheat-sheet/>
77. [https://www.python-kurs.eu/python3\\_dictionaries.php](https://www.python-kurs.eu/python3_dictionaries.php)
78. [https://www.reddit.com/r/Python/comments/d62du1/help\\_understanding\\_the\\_strangeness\\_of\\_dict/](https://www.reddit.com/r/Python/comments/d62du1/help_understanding_the_strangeness_of_dict/)
79. <https://stackoverflow.com/questions/3137685/using-property-decorator-on-dicts>
80. <https://statistikguru.de/python/python-schluessel-dictionarys-entfernen.html>
81. <https://wiki.somebot.de/python/dicts/methoden/>
82. <http://py-tutorial-de.readthedocs.io/de/python-3.3/datastructures.html>