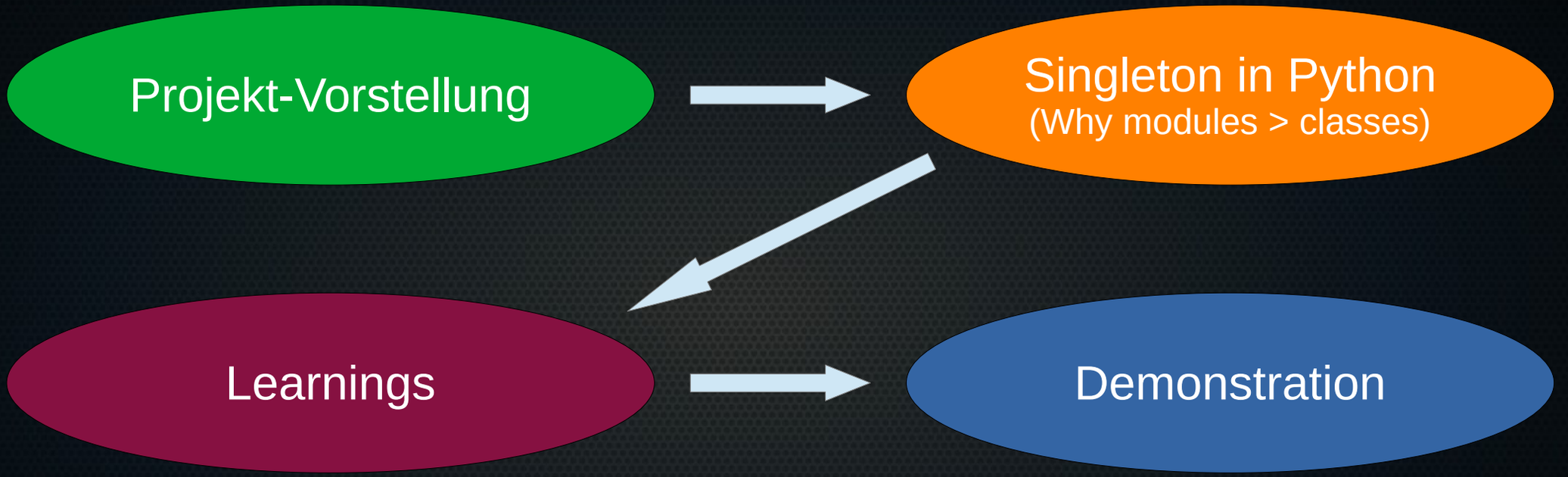
<-- RspCam -->

Raspberry Pi as a

Timelapse Camera

Übersicht


Projekt-Vorstellung
Singleton in Python
(Why modules > classes)
Learnings
Demonstration

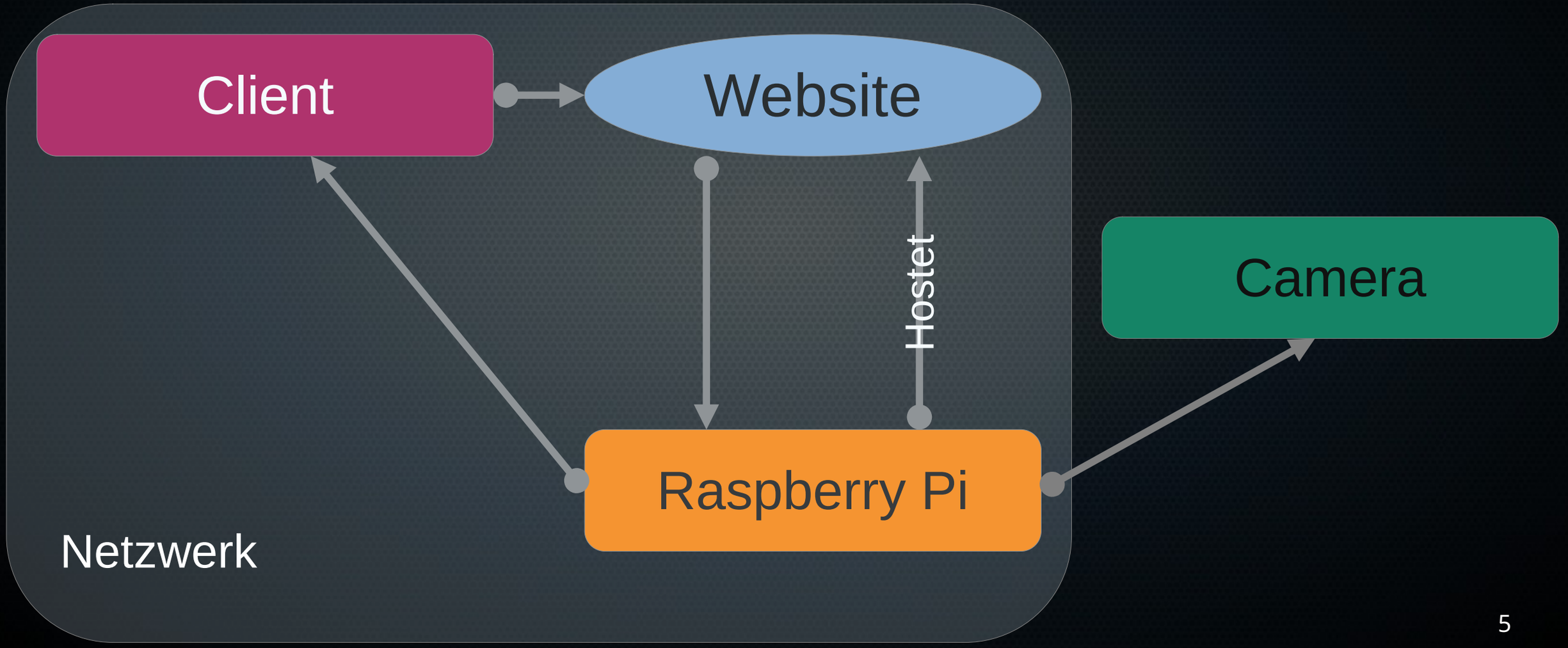
Projekt RspCam

Projekt RspCam

- Timelapse (Zeitraffer)
- Für 3d Druck

- In regelmässigen Abständen Photos machen
- UI (Benutzeroberfläche)
  - Start
  - Name
  - Einstellungen
- Zu einem Video machen und senden (SSH, Telegram)

Kommunikation zwischen User und Raspberry Pi

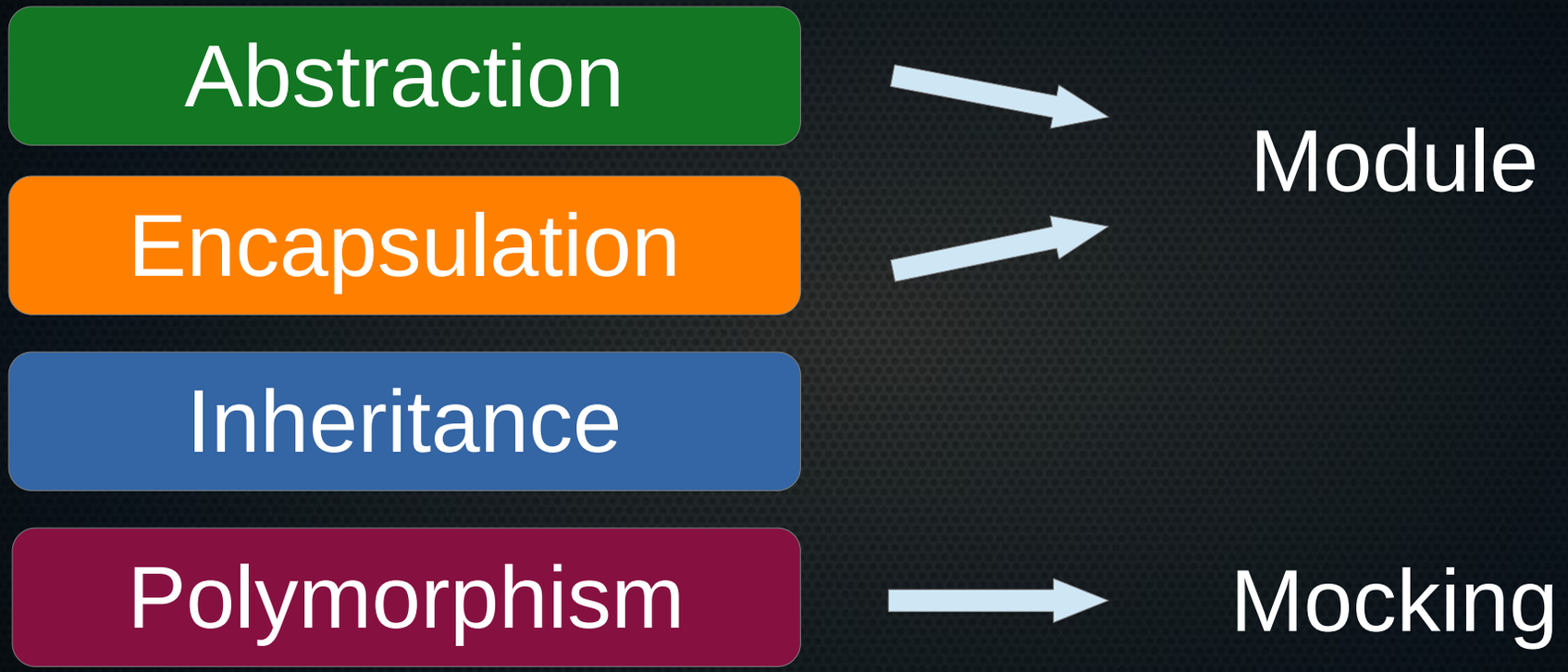


OOP Integration im Code

OOP Integration

- Klassen wurden benutzt im Code
  - Flask benötigt Funktionen
- OOP Principles wurden beachtet
- Singleton Pattern wurde verwendet (abgewandelte Form)

OOP Principles



Singleton Pattern

- Eine Klasse mit immer genau einer Instanz
- Konstruktor ist privat
- Stattdessen Klasse.get_instance() oder ähnlich
  - Beim ersten Aufruf wird das Objekt erstellt und gespeichert
  - Danach immer wiederverwendet

```java
public class Database {

    // Singleton Instanz wird gespeichert
    private static Database instance;

    public Connection connection;

    // Privater Konstruktor
    private Database() {
        connection = new Connection();
    }

    // Wird verwendet anstelle des Konstruktors
    public static Database getInstance() {

        // Wenn es noch kein Objekt existiert, wird ein neues erstellt und gespeichert
        if (instance == null) {
            instance = new Database();
        }

        // Falls es schon eines gibt, gib dieses zurueck
        return instance;
    }
}
```

```python
import os

on_raspberry_pi: bool = False
images_folder: str = 'output/images/'
timelapses_folder: str = 'output/timelapses/'
camera_port: int = 0

interval_minutes: float = 0.3
recording_duration_minutes: float = 10
```

```python
def load_settings():
    """
    Load settings from environment variables
    """
    global images_folder, camera_port, timelapses_folder, on_raspberry_pi,
interval_minutes, recording_duration_minutes

    # Not optional: ON_RASPBERRY_PI
    on_pi = os.getenv('ON_RASPBERRY_PI')
    assert on_pi is not None, 'ON_RASPBERRY_PI not set'
    on_raspberry_pi = bool(on_pi)

    # Optional: IMAGES_FOLDER
    path = os.getenv('IMAGES_FOLDER')
    if path is not None:
        images_folder = path

    # Optional: TIMELAPSES_FOLDER
    timelapses_path = os.getenv('TIMELAPSES_FOLDER')
    if timelapses_path is not None:
        timelapses_folder = timelapses_path

    # Not optional: CAMERA_PORT
    port = os.getenv('CAMERA_PORT')
    if port is not None:
        camera_port = int(port)
    else:
        # No default, needs to be set explicitly
        raise Exception('CAMERA_PORT not set')

    int_min = os.getenv('INTERVAL_MINUTES')
    assert int_min is not None, 'INTERVAL_MINUTES not set'
    interval_minutes = float(int_min)

    rec_dur = os.getenv('RECORDING_DURATION_MINUTES')
    assert rec_dur is not None, 'RECORDING_DURATION_MINUTES not set'
    recording_duration_minutes = float(rec_dur)
```

```python
load_settings()
```

```python
import os

class Settings:
    on_raspberry_pi: bool = False
    images_folder: str = 'output/images/'
    timelapses_folder: str = 'output/timelapses/'
    camera_port: int = 0

    interval_minutes: float = 0.3
    recording_duration_minutes: float = 10

    def __init__(self):
        self.load_settings()

    def load_settings(self):
        """
        Load settings from environment variables
        """

        # Not optional: ON_RASPBERRY_PI
        on_pi = os.getenv('ON_RASPBERRY_PI')
        assert on_pi is not None, 'ON_RASPBERRY_PI not set'
        self.on_raspberry_pi = bool(on_pi)

        # Optional: IMAGES_FOLDER
        path = os.getenv('IMAGES_FOLDER')
        if path is not None:
            self.images_folder = path

        # Optional: TIMELAPSES_FOLDER
        timelapses_path = os.getenv('TIMELAPSES_FOLDER')
        if timelapses_path is not None:
            self.timelapses_folder = timelapses_path

        # Not optional: CAMERA_PORT
        port = os.getenv('CAMERA_PORT')
        if port is not None:
            self.camera_port = int(port)
        else:
            # No default, needs to be set explicitly
            raise Exception('CAMERA_PORT not set')

        int_min = os.getenv('INTERVAL_MINUTES')
        assert int_min is not None, 'INTERVAL_MINUTES not set'
        self.interval_minutes = float(int_min)

        rec_dur = os.getenv('RECORDING_DURATION_MINUTES')
        assert rec_dur is not None, 'RECORDING_DURATION_MINUTES not set'
        self.recording_duration_minutes = float(rec_dur)
```

Singleton Pattern in Python

Module

```python
# Importieren + Instanz kriegen
import settings



# Auf Attribute (modul-globale Variablen) zugreifen
settings.interval_minutes

# Auf Methoden (Funktionen) zugreifen
settings.load_settings()
```

Klasse

```python
# Importieren
from settings import Settings

# Instanz kriegen
settings = Settings.get_instance()



# Auf Attribute (member variables) zugreifen
settings.interval_minutes

# Auf Methoden (member functions) zugreifen
settings.load_settings()
```

Learnings

- Web Interfaces sind nützlich für die Kommunikation mit dem Raspberry Pi
- Flask ist geeignet dafür
- Module können als Singleton dienen in Python
- Wie Mocking funktioniert

- Die Raspberry Pi-Kamera ist schlecht
- Immer commiten & pushen

Timelapse Video Demonstration