# Power Consumption Docs

**Project documentation with Markdown.**

*lordksix*

*Copyright © 2023 Wladimir Pasquel.*

# Table of contents

Copyright © 2023 Wladimir Pasquel.

# 1. Power Consumption Docs

This site contains the project documentation for the `Power Consumption` project which is an console application with PYTHON to measure the energy consumption of an app. It only works on LINUX

## 1.1 Table Of Contents

The documentation follows the best practice for project documentation as described by Daniele Procida in the Diátaxis documentation framework and consists of four separate parts:

Quickly find what you're looking for depending on your use case by looking at the different pages.

## 1.2 Project Overview

### 1.2.1 Backend package

Package that handles the backend of the application.

Modules exported by this package:

- `bash` : Provide the functions to change authorizations for the application to run.
- `engine` : Provide the class that handles the backend of the application.

## 1.3 Acknowledgements

I want to thank PowerAPI for the heavy lifting with their library.

Copyright © 2023 Wladimir Pasquel.

# 2. How-To Guides

This part of the project documentation focuses on a **problem-oriented** approach. You'll tackle common tasks that you might have, with the help of the code provided in this project.

## 2.1 How is it structured?

Clone the code from this GitHub repository on your preferred directory and following structure will be downloaded:

```
power-consumption-app/
├──app.py
├── docs/
│   |── explanation.md
│   |── index.md
│   ├── how-to-guides.md
|   └── reference.md
└── src/
    |
    ├── backend/
    |   ├── bash.py
    |   ├── engine.py
    |   └── __init__.py
    └── pyproject.toml
```

## 2.2 How to start

After cloning the project run the following command to install the necessary dependencies.

```
    cd power-consumption-app
    pip install power-consumption
```

## 2.3 How to configure

In the `app.py` file you will find the follow function:

```
@measure_energy(handler=csv_handler)
def application():
    """Replace code inside with the script you want to run"""
    solution = 0
    for _ in range(50):
        solution = ((1 + 2 + 3) ** 2) ** 2
    print(solution)
```

Replace the code inside the function block with your prefer code.

## 2.4 How to use it

Run

```
    python3 app.py
```

The console inside the function application() will run 100 times and the results will be saved in a file called `result.csv` or the name you have chosen.

First column: timestamp -> Timestamp of the start of the iteration Second column: tag -> Name of the function running Third column: duration -> Duration of the iteration Forth column: package_0 -> Total energy in uJ consumed by CPU during the iteration Fifth column: nvidia_gpu_0 -> Total energy in uJ consumed by GPU during the iteration

## 2.5 How to read the final result

The application will print in the console: - How much energy was consume in total - Total duration of all iterations - How much power the application consumes

Copyright © 2023 Wladimir Pasquel.

# 3. Reference

This part of the project documentation focuses on an **information-oriented** approach. Use it as a reference for the technical implementation of the `Tic Tac Toe` project code.

## 3.1 Backend packages

This package has the following modules:

1. Engine
2. Bash

Copyright © 2023 Wladimir Pasquel.

# 4. Engine module

Provide the class and functions that handles the backend of the application.

This module allows the application to run and process information

Examples:

```
>>> directory = path.dirname(path.realpath("__file__"))
>>> app = PowerConsumption(root_path=directory)
>>> csv_handler = app.create_csv()
```

The module contains the following class: - `PowerConsumption` : A class that represents the engine of the application.

The module contains the following functions: - `print_welcome_message` : Prints welcome message. - `set_file_name` : User input for the output file name. - `render_result` : Render the results.

## 4.1 `PowerConsumption` `dataclass`

A class used to represent the game engine.

**Attributes:**

| Name | Type | Description |
|---|---|---|
| `root_path` | `str` | str It is absolute path of the working directory. |
| `file_name` | `str` | str = "result" It is the name of the file where the data is going to be saved. Default to 'result'. |

**Methods:**

| Name | Description |
|---|---|
| `create_csv` | Returns the handler where the data is going to be save. |
| `get_path` | Returns the working directory path. |
| `parse_csv` | Parse the information save and return the total of energy consume. |

Copyright © 2023 Wladimir Pasquel.

**Source code in `src\backend\engine.py`**

```python
@dataclass(frozen=True)
class PowerConsumption:
    """A class used to represent the game engine.

    Attributes:
        root_path: str
            It is absolute path of the working directory.
        file_name: str = "result"
            It is the name of the file where the data is going to be saved.
            Default to 'result'.

    Methods:
        create_csv(self) -> CSVHandler:
            Returns the handler where the data is going to be save.
        get_path(self) -> str:
            Returns the working directory path.
        parse_csv(self) -> None:
            Parse the information save and return the total of energy
            consume.
    """

    root_path: str
    file_name: str = "result"

    def start(self):
        """Validation of permission for the application to work"""
        print_welcome_message()
        authorize_rapl()
        authorize_cwd(self.root_path)

    def get_path(self) -> str:
        """Returns the working directory path.

        Returns:
            str: File path with file name
        """
        name = set_file_name(self.file_name)
        file_extension = name + ".csv"
        abs_file_path = pt.join(self.root_path, file_extension)
        return abs_file_path

    def create_csv(self) -> CSVHandler:
        """Returns the handler where the data is going to be save.

        Returns:
            CSVHandler: Handler of the file where the information is going
                to be save
        """
        file_path = self.get_path()
        csv_file = CSVHandler(file_path)
        return csv_file

    def parse_csv(self) -> None:
        """Parse the information save and return the total of energy
        consume.
        """
        file_extension = self.file_name + ".csv"
        abs_file_path = pt.join(self.root_path, file_extension)
        with open(abs_file_path) as csv_file:
            next(csv_file)
            sum_duration = 0
            sum_energy = 0
            for line in csv_file:
                columns = line.split(";")
                sum_duration += float(columns[2])
                sum_energy += float(columns[3]) + float(columns[4])
            render_result(sum_energy, sum_duration)
```

### 4.1.1 `create_csv()`

Returns the handler where the data is going to be save.

**Returns:**

| Name | Type | Description |
|---|---|---|
| `CSVHandler` | `CSVHandler` | Handler of the file where the information is going to be save |

Copyright © 2023 Wladimir Pasquel.

**Source code in `src\backend\engine.py`**

```python
def create_csv(self) -> CSVHandler:
    """Returns the handler where the data is going to be save.

    Returns:
        CSVHandler: Handler of the file where the information is going
            to be save
    """
    file_path = self.get_path()
    csv_file = CSVHandler(file_path)
    return csv_file
```

## 4.1.2 `get_path()`

Returns the working directory path.

**Returns:**

| Name | Type | Description |
|---|---|---|
| `str` | `str` | File path with file name |

**Source code in `src\backend\engine.py`**

```python
def get_path(self) -> str:
    """Returns the working directory path.

    Returns:
        str: File path with file name
    """
    name = set_file_name(self.file_name)
    file_extension = name + ".csv"
    abs_file_path = pt.join(self.root_path, file_extension)
    return abs_file_path
```

## 4.1.3 `parse_csv()`

Parse the information save and return the total of energy consume.

**Source code in `src\backend\engine.py`**

```python
def parse_csv(self) -> None:
    """Parse the information save and return the total of energy
    consume.
    """
    file_extension = self.file_name + ".csv"
    abs_file_path = pt.join(self.root_path, file_extension)
    with open(abs_file_path) as csv_file:
        next(csv_file)
        sum_duration = 0
        sum_energy = 0
        for line in csv_file:
            columns = line.split(";")
            sum_duration += float(columns[2])
            sum_energy += float(columns[3]) + float(columns[4])
        render_result(sum_energy, sum_duration)
```

## 4.1.4 `start()`

Validation of permission for the application to work

Copyright © 2023 Wladimir Pasquel.

**Source code in `src\backend\engine.py`**

```python
def start(self):
    """Validation of permission for the application to work"""
    print_welcome_message()
    authorize_rapl()
    authorize_cwd(self.root_path)
```

## 4.2 `print_welcome_message()`

Prints welcome message

**Source code in `src\backend\engine.py`**

```python
def print_welcome_message():
    """Prints welcome message"""
    print("Welcome to the Power App")
    print("For now it just runs on Linux")
    print("Get ready to now how much energy you script consumes")
```

## 4.3 `render_result(sum_energy, sum_duration)`

Render the results

**Source code in `src\backend\engine.py`**

```python
def render_result(sum_energy: float, sum_duration: float) -> None:
    """Render the results"""
    joules = sum_energy / 1000000
    print(f"Total energy {joules} J")
    print(f"Total energy {sum_duration} s")
    watt = joules / sum_duration
    print(f"Total power {watt} W")
```

## 4.4 `set_file_name(default)`

User input for the output file name

**Returns:**

| Name | Type | Description |
|---|---|---|
| `str` | `str` | Returns file name selected or 'result' |

**Source code in `src\backend\engine.py`**

```python
def set_file_name(default: str) -> str:
    """User input for the output file name

    Returns:
        str: Returns file name selected or 'result'
    """
    print("What name of the output file")
    file_name: str = input("Enter for default('result'): ").strip()
    if file_name == "":
        return default
    return file_name
```

Copyright © 2023 Wladimir Pasquel.

# 5. Bash module

Provide the functions to change authorizations for the application to run.

This module changes permissions intel RAPL and local directories.

Examples:

```
>>> authorize_rapl()
>>> authorize_cwd(path_chosen)
```

The module contains the following functions: - `authorize_rapl` : A function tht changes the permissions of the Intel RAPL directory. - `authorize_cwd` : A function tht changes the permissions of the chosen directory.

## 5.1 `authorize_cwd(path)`

Changes the permissions of the chosen directory

**Parameters:**

| Name | Type | Description | Default |
|---|---|---|---|
| `path` | `str` | Chosen directory | *required* |

**Source code in `src\backend\bash.py`**

```python
def authorize_cwd(path: str):
    """Changes the permissions of the chosen directory

    Args:
        path (str): Chosen directory
    """
    print("Change permissions of current directory")
    command = f"sudo chmod -R 777 {path}"
    subprocess.run(command, shell=True, stdout=subprocess.PIPE, check=True)
```

## 5.2 `authorize_rapl()`

Changes the permissions of the Intel RAPL directory

**Source code in `src\backend\bash.py`**

```python
def authorize_rapl():
    """Changes the permissions of the Intel RAPL directory"""
    print("Change permissions to run Intel RAPLs")
    subprocess.run(
        "sudo chmod -R a+r /sys/class/powercap/intel-rapl",
        shell=True,
        stdout=subprocess.PIPE,
        check=True,
    )
```

Copyright © 2023 Wladimir Pasquel.