

# CPTD CLI

**CPTD CLI** — это не просто инструмент командной строки. Это расширяемая платформа управления, предназначенная для:

- создания пользовательских команд и расширений;
- обмена командами между пользователями;
- интеграции с внешними инструментами и API;
- автоматизации рабочих процессов, отчетности и стратегического анализа;
- работы в качестве движка для любого пользовательского или графического интерфейса (UI).

## Принципы архитектуры

### 1. CLI как расширяемая платформа

Каждая команда — это обычный Python-файл с определенным интерфейсом. Вы можете создать свою собственную команду менее чем за 5 минут.

Команды — это простые Python-модули с минимальной структурой. Каждая команда включает файл манифеста с информацией (имя, описание, автор, версия, зависимости). Разработчики могут использовать шаблон `cptd newcommand` для быстрого старта.

Команды можно тестировать и отлаживать в интерактивном режиме без перезапуска системы:

```
cptd command --add yourcommand.zip      # добавление команды (только ZIP)
cptd command --del yourcommand          # удаление команды
```

**Запустить команду:**

```
cptd yourcommand
```

**Запустить проект:**

```
cptd yourcommand
```

### 2. Безопасность и валидация

- Все команды в общем репозитории проходят строгую проверку безопасности.
- При установке выполняется автоматическая проверка на запрещённый код (например, `pip install` внутри команды).
- При публикации в общий репозиторий команда проверяется на безопасность, структуру и целостность манифестов.
- Команды, предложенные сообществом, проходят модерацию перед публикацией.

### 3. CLI как движок для UI

CLI является мостом между графическими интерфейсами, которые используют его как основной движок. CPTD CLI работает как бэкенд для всех текущих и будущих интерфейсов. Вся логика обрабатывается через CLI.

### 4. Централизованное и децентрализованное распространение

- Команды могут быть загружены и использованы из общего репозитория.
- Поддерживается стандартный формат для импорта, экспорта и обмена командами.

## 5. Автономность и надёжность

- Работает полностью офлайн — облако не требуется.
- Нет телеметрии, скрытых сборов данных или внешних подключений.
- Поддерживаются Windows, Linux и macOS.

## Почему это важно

- **Гибкость:** адаптация CLI под любые сценарии — от проверки лицензий до автоматизации.
- **Масштабируемость:** от одного разработчика до целых команд.
- **Расширяемость:** создание, обмен, модерация и интеграция команд.
- **Безопасность:** строгая проверка на всех этапах — установка, выполнение, публикация.
- **Прозрачность:** весь код открыт, модульный и проверяемый.

## Открытый исходный код и репозиторий

CPTD CLI — это бесплатный проект с открытым исходным кодом. Полный исходный код доступен в публичном репозитории:

<https://github.com/asbjornrasen/cptd-dsl>

Это обеспечивает полную прозрачность, повышает доверие и безопасность, а также даёт каждому возможность проверить, доработать или скопировать систему. Благодаря открытости CPTD гарантирует независимость и проверяемость в долгосрочной перспективе.

---

## Как добавить новую команду в CPTD CLI

### Формат отправки (только ZIP)

Все команды CPTD CLI должны отправляться в виде архива .zip.

**Пример простой команды:**

```
taskcleaner.zip
├─ main.py
├─ manifest.yaml
└─ manifest.json
```

**Пример команды проекта с подпапками:**

```
taskmanager.zip
├─ main.py
├─ manifest.yaml
├─ manifest.json
├─ util/
│   └─ parser.py
└─ service/
```

└─ api.py

## Правила:

- main.py, manifest.yaml и manifest.json должны находиться в корне архива
- Архив не должен содержать вложенной папки с именем команды
- Имя архива определяет имя команды:  
taskcleaner.zip → cptd taskcleaner
- В обоих манифестах поле entrypoint должно быть равно main.py
- Если main.py находится не в корне, команда будет отклонена
- Оба манифеста обязательны (YAML и JSON)
- Допускаются подпапки (util/, service/) для модульности
- Автоматическая установка зависимостей внутри кода запрещена

## Обязательные элементы команды:

### 1. Описание команды в SYNTAX:

```
SYNTAX = {
    "name": "yourcommand",
    "description": "Что делает команда",
    "usage": "cptd yourcommand --input <путь> [--flag]",
    "arguments": [
        {"name": "--input", "required": True, "help": "Путь к входному файлу"},
        {"name": "--flag", "required": False, "help": "Необязательный флаг"}
    ],
    "examples": [
        "cptd yourcommand --input file.cptd",
        "cptd yourcommand --input folder --flag"
    ]
}
```

### 2. Функция run(argv):

```
def run(argv):
    ...
```

### 3. Обработка --help:

```
if "--help" in argv or "-h" in argv:
    print_help(SYNTAX)
    return
```

### 4. Вывод справки при ошибке:

```
except Exception as e:
    print(f"[!] Ошибка аргумента: {e}")
    print_help(SYNTAX)
    return
```

## Рекомендуемый шаблон:

```
from pathlib import Path
import argparse
from cptd_tools.syntax_utils import print_help

SYNTAX = {
```

```

"name": "yourcommand",
"description": "Описание команды",
"usage": "cptd yourcommand --input <путь> [--flag]",
"arguments": [
    {"name": "--input", "required": True, "help": "Путь к файлу или папке"},
    {"name": "--flag", "required": False, "help": "Флаг"}
],
"examples": [
    "cptd yourcommand --input file.cptd",
    "cptd yourcommand --input folder --flag"
]
}

def run(argv):
    if "--help" in argv or "-h" in argv:
        print_help(SYNTAX)
        return

    parser = argparse.ArgumentParser(description=SYNTAX["description"],
add_help=False)
    parser.add_argument('--input', type=Path, required=True, help='Путь к файлу
или папке')
    parser.add_argument('--flag', action='store_true', help='Флаг')

    try:
        args = parser.parse_args(argv)
    except Exception as e:
        print(f"[!] Ошибка аргумента: {e}")
        print_help(SYNTAX)
        return

    if not args.input.exists():
        print(f"[!] Путь не существует:\n    {args.input}")
        return

    print(f"[✓] Обработка: {args.input}")
    if args.flag:
        print("[✓] Флаг установлен")

```

---

### Добавить или протестировать команду:

- Добавление: `cptd command --add yourcommand.zip`
- Просмотр всех команд: `cptd list`
- Получить справку: `cptd yourcommand --help`
- Удалить команду: `cptd command --del yourcommand`

### Стандарты:

- SYNTAX обязателен
- `run(argv)` обязателен
- `--help` нельзя реализовывать через `argparse`; только через `print_help(SYNTAX)`
- Код должен быть чистым, читаемым, без лишних зависимостей

### Манифесты:

Оба манифеста должны находиться в одной папке с `main.py`

- `manifest.yaml` — читаемый человеком
- `manifest.json` — читаемый машиной

#### Обязательные поля в манифестах:

- `name`: уникальное имя команды (совпадает с именем архива)
  - `description`: описание
  - `version`: например, `1.0.0`
  - `entrypoint`: всегда `main.py`
  - `target`: поддерживаемые ОС (`all`, `linux`, `windows`, `macos`)
  - `dependencies`: список зависимостей `pip`
  - `author`: имя автора
  - `email`: контакт
  - `github`: ссылка на GitHub
  - `website`: сайт (необязательно)
  - `license`: лицензия (например, MIT, `license.md` и т.д.)
- 

### Готовы? Отправьте свою команду в официальный репозиторий CPTD CLI

1. Сделайте fork:  
<https://github.com/asbjornrasen/cptdcli-plugin>
2. Создайте ветку:  
`feature/mycommand`
3. Добавьте ZIP-файл в:  
`cptdcli-plugin/community_plugin/yourcommand.zip`
4. Убедитесь, что:
  - структура верна
  - `main.py`, манифесты и папки в корне
  - `--help` работает
  - нет логики автоустановки зависимостей
5. Добавьте манифест в конец `community-plugins.json`:

```
{
  "name": "example",
  "description": "example",
  "version": "1.0.0",
  "target": "Windows",
  "entrypoint": "example.py",
  "dependencies": ["example"],
  "author": "example",
  "email": "example@example.com",
  "github": "https://github.com/example/example",
  "website": "https://example.com",
  "license": "example.md"
}
```

В `target` укажите: Windows, Linux, MacOS или All.

6. Откройте Pull Request с описанием.
-

**Совет: следуйте философии CPTD — ясность, модульность, практичность.**

**Нужен шаблон?**

```
cptd newcommand
```

Вы получите структуру проекта с `main.py`, `manifest.yaml`, `util/`, `service/`.

---

**Готовы создавать команды? CPTD CLI ждёт ваших идей.**

Лучшие будут включены в официальный релиз.

---

## **Резюме**

CPTD CLI — это больше чем инструмент. Это основа для создания, проверки и обмена интеллектуальными утилитами. Его гибкая архитектура, строгая безопасность и открытая модель делают его идеальным ядром управления как для персональных, так и для корпоративных систем.