

## Setting up the ioProc environment

To setup the ioprocv env follow the following steps closely:

- open the anaconda command prompt (on windows: powershell is preferred)
- to create the env enter into the shell:  
(base)> conda create -n ioprocv  
python=3.7 pyyaml click pip frozendict pandas tables
- *Alternative:* Use the requirements.txt from gitlab
- confirm with "y" if requested
- After completion activate the new env:  
(base)> conda activate ioprocv
- install the last libraries and ioprocv with pip into the activated environment:  
(ioprocv)> pip install arrow cerberus ioprocv

## setup ioProc workspace

step 1

### source env

initialize the environment in conda that has ioProc installed

```
(base)> conda activate ioprocv
```

step 2

### create workspace

run ioProc to setup folders and the workspace structure

```
(ioprocv)> python -m ioprocv --setupfolders
```

step 3

### execute 1st project

run ioProc to execute a project. Change first in the created projects folder

```
(ioprocv)> python -m ioprocv
```

### Help in ioProc

if you are unsure how to use ioProc or need help with its commands you can always type in a ioprocv shell:

```
(ioprocv)> python -m ioprocv --help
```



### *folder* **root**

both location and name can be chosen freely.

### *folder* **actions**

contains all shared actions useable by IOProc workflows

### *folder* **projects**

contains project folders (no naming restrictions apply). One folder contains all information and data for a project

### *file* **create\_folder\_structure.py**

driver file to be used with PyCharm. This script recreates the folderstructure of the workspace.

### *file* **general.py**

contains general actions for IOProc workflows that are also used in the example workflow of a new project

### *folder* **project1**

contains an example project that is created at workspace setup. Can be renamed and modified to start a new project.

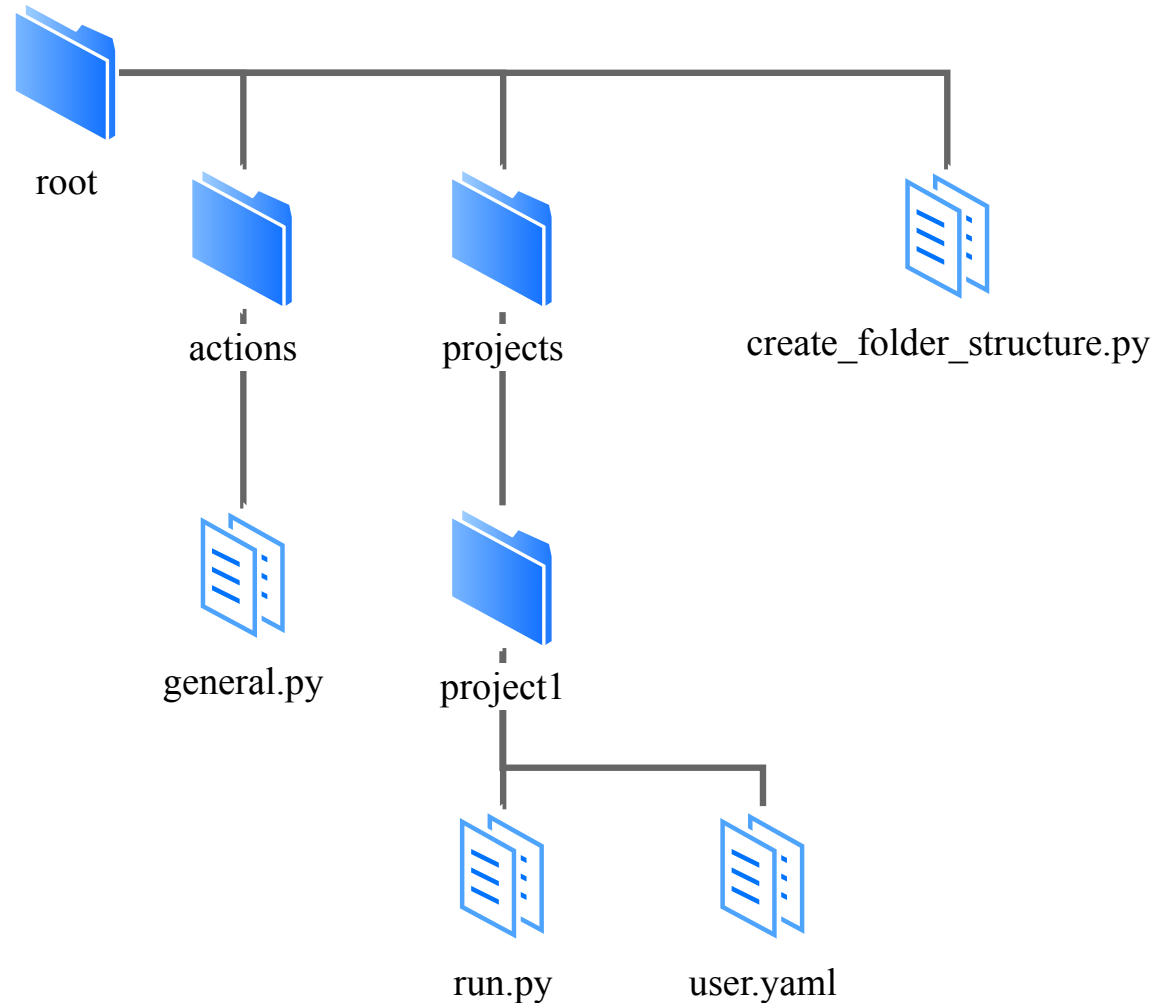
### *file* **run.py**

driver file to be used with PyCharm to execute an IOProc workflow.

### *file* **user.yaml**

contains the workflow and configuration of IOProc.

## ioProc workspace structure



**example user.yaml** as generated for the default project. Contains a basic workflow with 3 steps. Including one checkpoint "actionCheck"

### **actionFolder:**

C:/path/2/workspace/actions

### **debug:**

*timeit*: false

### **fromCheckPoint:** start

### **workflow:**

#### **- action1:**

*call*: readExcel

*fieldName*: dummy

*param1*: 7

*param2*: true

*project*: general

#### **- action3:**

*call*: printData

*project*: general

#### **- actionCheck:**

*call*: checkpoint

*project*: general

*tag*: test

### **actionFolder**

path to the folder, that contains the actions used by this workflow. By default, this folder is two folders above the project folder.

### **debug**

if debugging information should be displayed. *timeit*: True shows the time for processing each action.

### **fromCheckPoint**

the workflow is run from a predefined checkpoint. The checkpoint has to have a unique name and the workflow has to have run before up to the checkpoint. If the full workflow should be executed, set this to start.

### **workflow**

the actual workflow. This is a list of actions and their respective configuration.

### **action1**

An action in the workflow. The action config follows. The name of the action (action1 in this example) can be changed at will to an unique alphanumeric name. The config of an action contains several mandatory fields:

- *call*: the action that should be called in this step. Maps to a function in the action folder of the workspace.
- *project*: the project the action belongs to. All actions have to belong to a project

additional fields depend on the parameters required by the action and have to be specified if required.