Module 2 - Build and packaging

Consider the tree structure of **module_1**, all source is located in **src** and we have 2 modules **binary_reader** and **xlsx_filter** (the latter has been now added as we did not have time to go trough it during the training).

```
module_1
├── config
|   ├── data_spec.json
|   └── data_spec.xml
├── data
|   ├── test_data_0.dat
|   └── test_data_0.txt
├── generate_test_data.php
├── src
|   ├── binary_reader
|   |   ├── configuration
|   |   |   ├── __init__.py
|   |   |   └── __pycache__
|   |   |       └── __init__.cpython-39.pyc
|   |   ├── __init__.py
|   |   ├── input
|   |   |   ├── __init__.py
|   |   |   └── __pycache__
|   |   |       └── __init__.cpython-39.pyc
|   |   ├── __main__.py
|   |   ├── output
|   |   |   ├── __init__.py
|   |   |   └── __pycache__
|   |   |       └── __init__.cpython-39.pyc
|   |   └── util
|   |       ├── __init__.py
|   |       └── __pycache__
|   |           └── __init__.cpython-39.pyc
|   └── xlsx_filter
|       └── __main__.py
└── test.xlsx
```

There were 2 issues I could not explain at that point: 1) the package I built had no content and 2) once it had content the import statements to inner modules were failing and it could not be run inside my PC where I was trying to install it.

For the first issue `setup.cfg` has 3 important options that I did not explain correctly - hence packages that we've built were not containing all files:

```
[options]
package_dir =
    = src
packages = find:
.
.
.
[options.packages.find]
where = src
```

1. `[options] package_dir` - this is the **root** directory of where all source files are located (will be copied from) and not the path where to look for packages when using `find:` function;
2. `[options] packages` - is a list of packages in the form `binary_reader, binary_reader.inp, ...` or if detecting packages automatically just `find:` function;
3. `[options.packages.find] where` - if `find:` function is used instead of explicit naming of the packages, then this path must be also specified, pointing into the **root** / `package_dir`.

For the second issue, assuming we are in the `code/module_1/src` following commands will both run our application but first one will do so trough `runpy` module https://docs.python.org/3/library/runpy.html and will expects all imports to start with the full path of module being invoked, while the second one will directly execute specific file and import modules as relative paths. If second one is used for development (as in our case) and the package is built, distributed and deployed then it will not work (will have missing modules).

```
python -m binary_reader -c ../config/data_spec.xml --input-data ../data/*.dat"
python binary_reader/__main__.py -c config/data_spec.xml --input-data data/*.dat"
```

Last issue before end of the training

In the video part 3 - 00:33:38 (17:10) of the module **binary_reader/inp** `Reader.read_files` I have accidently renamed `read` to `seek`, thus extracted raw value was always an integer - a number of extracted bytes, which then failed in unpacking just few lines below.

The code is now corrected as per below:

```
    @Debug()
    def read_files(self, pattern, packet_structure):
        fields = [
            attribute.attrib for attribute in packet_structure.header.attribute
        ]
        fields.extend([
            variable.attrib for variable in packet_structure.payload.variable
        ])
        all_packets = []
        packet_count = 0
        for entry in glob.glob(pattern):
            packet_offset = 0
            with InputDataset(entry) as input_dataset:
                print(input_dataset)
                while packet_offset < input_dataset:
                    packet_count += 1
                    single_packet = []
                    for field in fields:
                        field_offset = packet_offset + int(field['offset'])
                        field_size = int(field['size'])
                        input_dataset.file_obj.seek(field_offset)
-                        raw = input_dataset.file_obj.seek(field_size)
+                        raw = input_dataset.file_obj.read(field_size)

                        if hasattr(self, '_unpack_{}'.format(field['type'])):
                            value = getattr(self, '_unpack_{}'.format(field['type']))
(raw)
                        else:
                            value = struct.unpack('B', raw)[0]
                        single_packet.append({
                            'name': field['name'],
                            'value': value
                        })
```

```
                    packet_offset += int(fields[-1]['offset']) + int(fields[-1]
['size'])
                    all_packets.append(single_packet)
        return all_packets
```