Bloomberg: Data retrieval & applications

Celia Fseil

M2 272 - Ingénierie Economique et Financière - Université Paris-Dauphine

2023-2024

Table of contents

Table of contents

Syllabus

- 7 sessions of 3 hours

- Provisional program:
  02/12: Formulas, BQL, VBA reminders
  02/12: Excel API I
  09/12: Excel API II
  09/12: Python API I
  17/12: Python API II
  17/12: Introduction to B-Quant + Portfolio loading and monitoring in Bloomberg

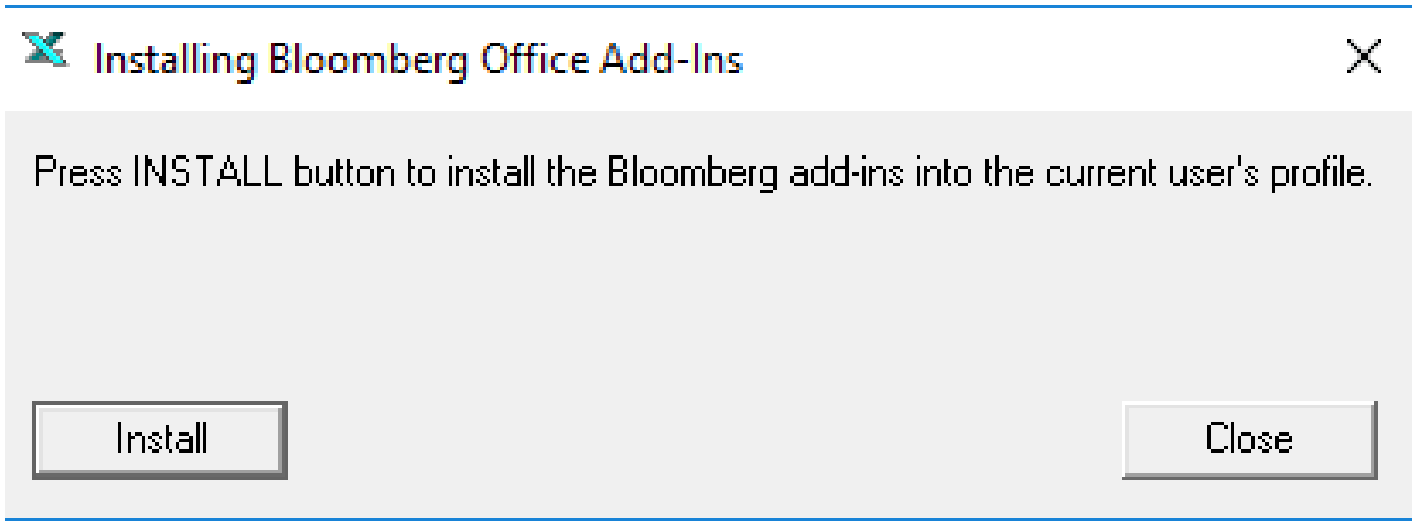
Assignments

- Project : TBC
- In groups of 2
- Grading schemes will be communicated beforehand
- To be sent to fseil_celia@hotmail.fr before (TBD) 11:59 pm.
- Defence sessions for each project will be carried out.

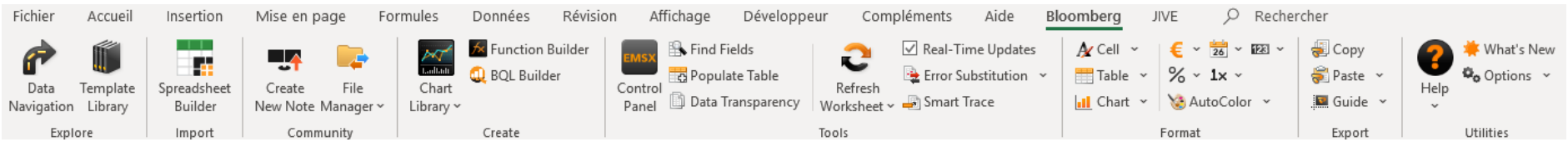
Table of contents

Bloomberg Excel add-in tab

To be able to use Bloomberg within Excel, you will need to close Excel and install the office tools add-ins. Within the search bar, type "Installing Bloomberg Office Add-Ins" and click install.



In Option menu of Excel, add Bloomberg to the ribbon.



The 3 basic formulas

Bloomberg Data Point

$= BDP(Ticker; Field)$

BDP is usually used to get the latest data available for a particular field.

Bloomberg Data History

$= BDH(Ticker; Field; Start\_Date; End\_Date; Fill; Days; Periodicity; Currency)$

BDH is used to get the historical data on a certain field for a particular security.
fill: indicates how you want missing data to be filled out
days: specifies which days of will be retrieved (e.g calendar, trading days)
per: lets you choose the periodicity of the data you're requesting
curr: allows you to set a currency different to the instrument's default

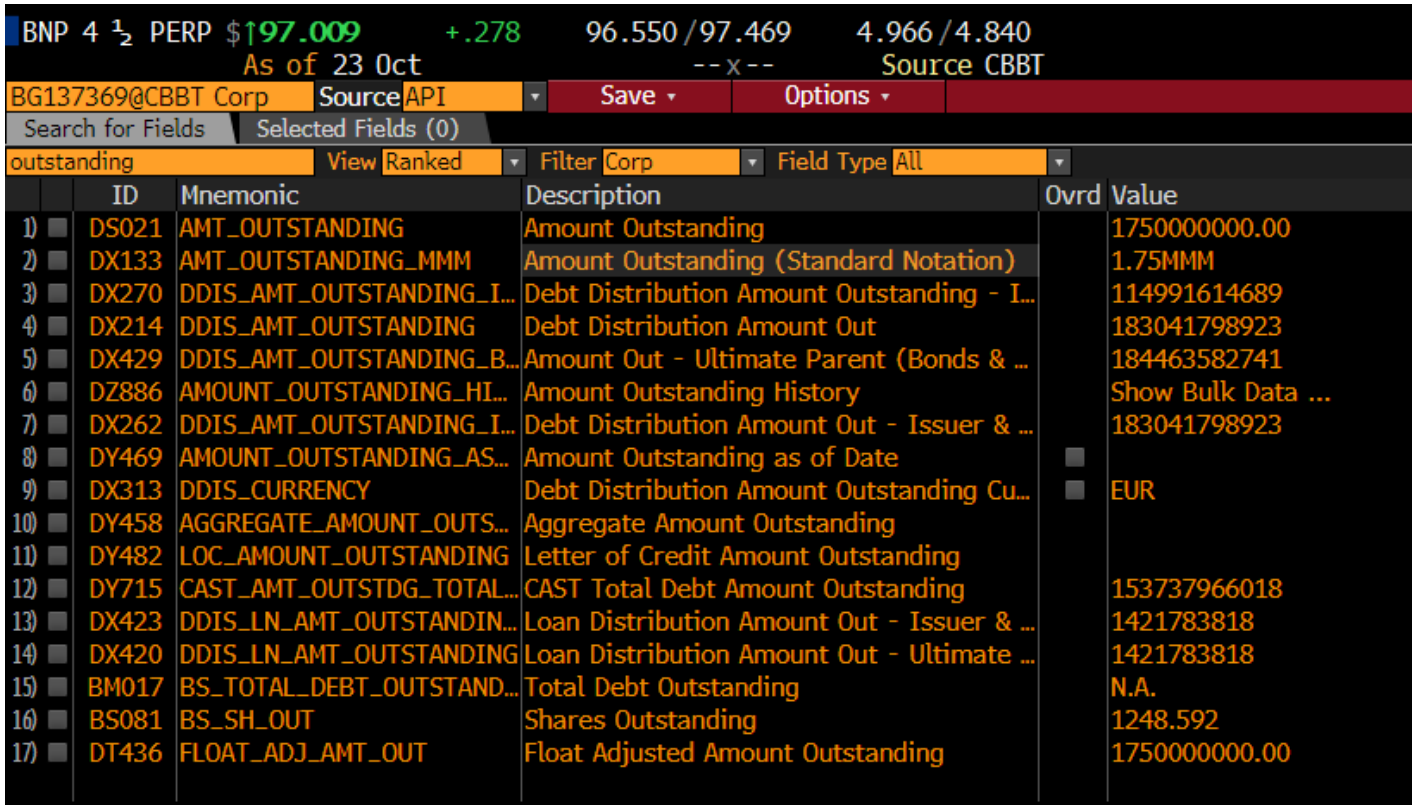
Bloomberg Data Set

$= BDS(Ticker; Field)$

BDS is similar to BDP to the exception that it returns data on several cells instead of just one.

Choosing the right functions

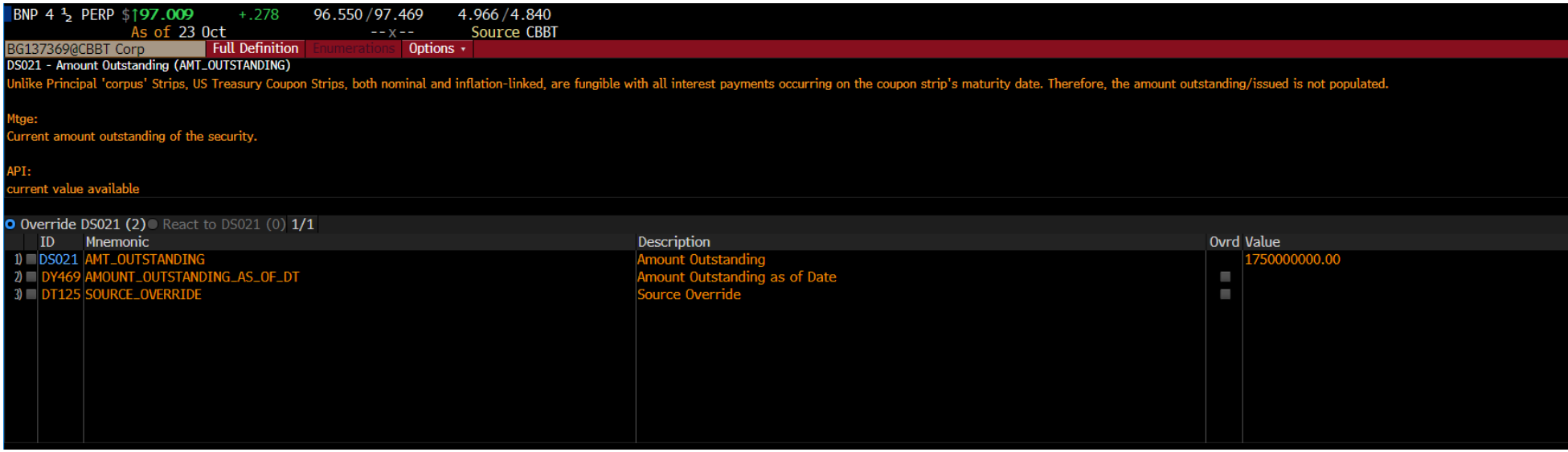
Fields are not all supported by either functions. A good way to check if your field is supported is to open the security in the terminal then type in **FLDS** and the field you want details on.



	ID	Mnemonic	Description	Ovrd	Value
1)	DS021	AMT_OUTSTANDING	Amount Outstanding		1750000000.00
2)	DX133	AMT_OUTSTANDING_MMM	Amount Outstanding (Standard Notation)		1.75MMM
3)	DX270	DDIS_AMT_OUTSTANDING_I...	Debt Distribution Amount Outstanding - I...		114991614689
4)	DX214	DDIS_AMT_OUTSTANDING	Debt Distribution Amount Out		183041798923
5)	DX429	DDIS_AMT_OUTSTANDING_B...	Amount Out - Ultimate Parent (Bonds & ...		184463582741
6)	DZ886	AMOUNT_OUTSTANDING_HI...	Amount Outstanding History		Show Bulk Data ...
7)	DX262	DDIS_AMT_OUTSTANDING_I...	Debt Distribution Amount Out - Issuer & ...		183041798923
8)	DY469	AMOUNT_OUTSTANDING_AS...	Amount Outstanding as of Date		
9)	DX313	DDIS_CURRENCY	Debt Distribution Amount Outstanding Cu...		EUR
10)	DY458	AGGREGATE_AMOUNT_OUTS...	Aggregate Amount Outstanding		
11)	DY482	LOC_AMOUNT_OUTSTANDING	Letter of Credit Amount Outstanding		
12)	DY715	CAST_AMT_OUTSTDG_TOTAL...	CAST Total Debt Amount Outstanding		153737966018
13)	DX423	DDIS_LN_AMT_OUTSTANDIN...	Loan Distribution Amount Out - Issuer & ...		1421783818
14)	DX420	DDIS_LN_AMT_OUTSTANDING	Loan Distribution Amount Out - Ultimate ...		1421783818
15)	BM017	BS_TOTAL_DEBT_OUTSTAND...	Total Debt Outstanding		N.A.
16)	BS081	BS_SH_OUT	Shares Outstanding		1248.592
17)	DT436	FLOAT_ADJ_AMT_OUT	Float Adjusted Amount Outstanding		1750000000.00

This is the example of a corporate bond. In the search bar is typed "outstanding" which allows us to see all the existing fields related to this word.

When selecting a field, you shall be able to see two things. First, at the bottom of the field description is mentioned how the field is supported in the API. Secondly, **FLDS** is also useful to easily see which overrides are available.



In this example, for the field AMT_OUTSTANDING, only current values are available, meaning it can only be retrieved via BDP. However, you can see that some overrides are available, in particular AMOUNT_OUTSTANDING_AS_OF_DT which indicates older values are nonetheless accessible.

The function builder

The add-in includes a function builder which is useful for less standard and more complex functions.

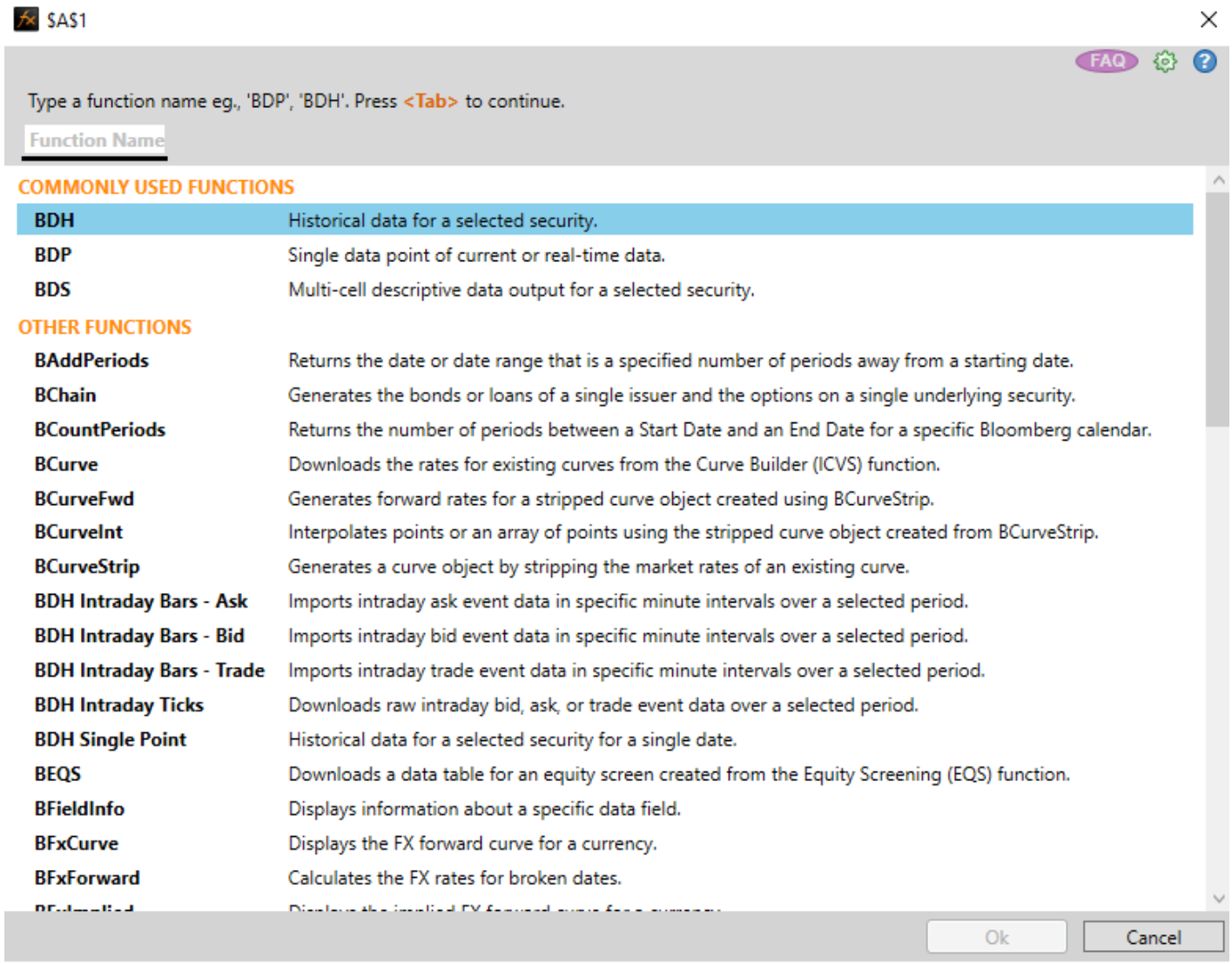


Table of contents

Bloomberg Query Language

Definition

BQL (for Bloomberg Query Language) is a language developed by Bloomberg. Similar to SQL in its construction, it transposes the concept of database queries to Bloomberg. You can therefore retrieve raw as well transformed data directly from the Bloomberg server.

Advantages

Efficiency gains
Limited use of data quotas as you only consume the aggregated data instead of all the intermediate points in-between.
Lets you use an index ticker instead of all its constituents.
Can be used in Python via B-Quant

Drawbacks

Steep learning curve
Difficult syntax
Strong dependency to Excel

Two functions are available: BQL() and BQL.Query().

BQL()

General formula

$= BQL(" Universe" ;" Expression" ;" Optional\ Parameters" ;" Optional\ Local\ Variables")$

Syntax structure:

1- Define the universe
2- Define the set of fields
3- Define the analysis / calculations you want to perform (e.g. filtering, scoring, grouping)
4- Add overrides and options to each field (e.g. currency, time period etc)

Rk/ The separation between the different parts of the formula is dependent on your computer's regional settings (either ; or ,).

The separation within the different parts is always a comma.

BQL() formulas are case-insensitive.

Examples

Different ways of defining the Universe

$= BQL("TTE\ FP\ Equity";"px\_last")$
$= BQL(("TTE\ FP\ Equity, GLE\ FP\ Equity");"px\_last")$
$= BQL("members(['CAC\ Index'])";"px\_last")$
$= BQL("members(['CAC\ Index','SPX\ Index'])";"px\_last")$

Rk/ As mentioned above, all tickers should be comma delimited

Different ways of defining the expression:

$= BQL("TTE\ FP\ Equity";"px\_last")$
$= BQL("TTE\ FP\ Equity";"px\_last, px\_low")$

Optional parameters : dates : Current value :

$= BQL("TTE\ FP\ Equity, MSFT\ US\ Equity";"px\_last")$

Optional parameters : dates : Historical data point

$= BQL("TTE\ FP\ Equity";"px\_last";"dates = -1M")$
$= BQL("TTE\ FP\ Equity";"px\_last";"dates = 2020-10-01")$

Optional parameters : dates : Historical value series : Relative dates

$= BQL("TTE\ FP\ Equity";"px\_last";"dates = range(-1M, 0D)";"currency = EUR")$
$= BQL("TTE\ FP\ Equity";"px\_last(dates = range(-1M, 0D), currency = EUR)")$

Optional parameters : dates : Historical value series : Absolute dates

$= BQL("TTE\ FP\ Equity";"px\_last(dates = range(2014-01-07, 2014-01-15))")$
$= BQL("TTE\ FP\ Equity";"px\_last";"dates = range(-1M, 0D)";"currency = EUR")$

Optional parameters : periodicity, fill

$= BQL("TTE\ FP\ Equity";"px\_last(dates = range(2014-01-01, 2016-01-15), per = M, fill = PREV)")$
$= BQL("TTE\ FP\ Equity";"px\_last";"dates = range(2014-01-01, 2016-01-15)";"per = M";"fill = PREV";)$

Aggregation: average PE ratio per GICS sector amongst the SPX Index

$= BQL("members('SPXIndex')";"avg(group(pe_ratio, classification_name(classification_scheme = GICS)))")$

BQL.Query()

General formula

$= BQL.Query(" < let() > get()for() < with() > ")$

While BQL() resembles the standard Bloomberg formulas, BQL.Query() uses a raw string of BQL code, containing get() and for() clauses.

Syntax structure:

1- let(): <optional>: lets you create one or more variable name and value pairs. A name-value pair starts with # and ends with a semi-colon ;
(e.g., $\#lastPrice = px\_last(dates = range(-2M, 0D));$)

2- get(): required : similar to the Expression with BQL formulas

3- for(): required : similar to the Universe with BQL formulas

4- with(): <optional>: to control the format or calculations of the data (e.g. dates)

Example

Series from the last month with lastPrice and lowPrice as variable names

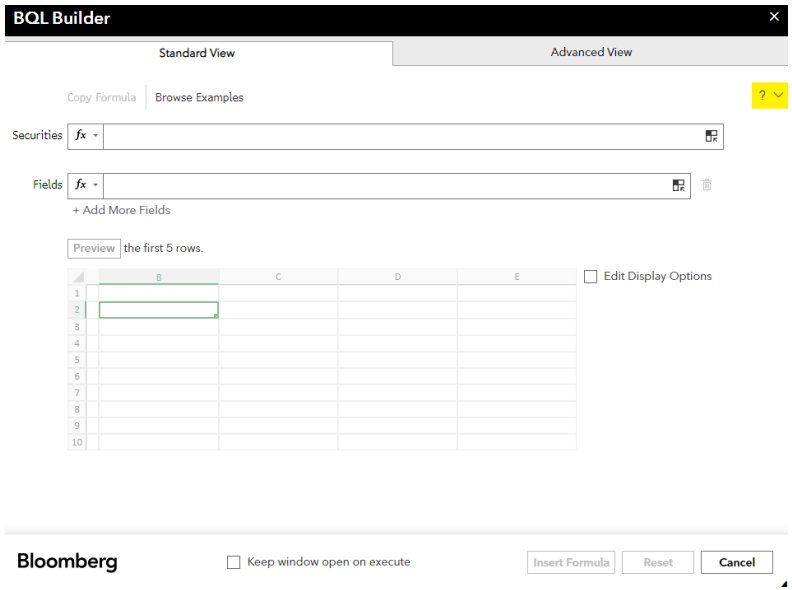
$$= BQL.Query(" let(lastPrice = px\_last; lowPrice = px\_low;)get(lastPrice, lowPrice)for(['TTE\ FP\ Equity'])with(dates = range(-\mathbf{1}M, \mathbf{0}D))")$$

Rk/ BQL.Query() formulas are case-insensitive.

Support with BQL

A useful tool to approach BQL is the **BQLX** function. Several tutorial videos and practical examples are frequently uploaded for each asset class.

A more pragmatic approach is to use the BQL Builder. Similarly to the Function Builder, this tool allows you to create your query step by step, without having to worry about syntax technicalities.



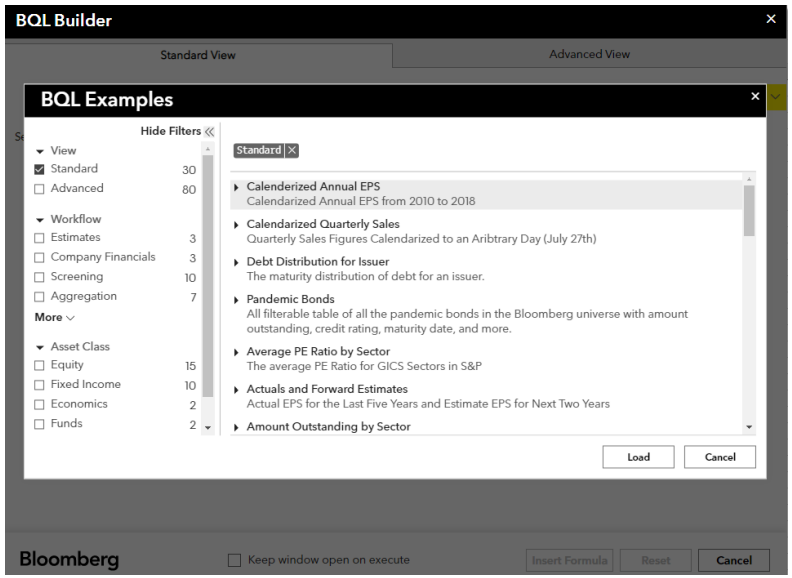


Table of contents

The 3 paradigms

- **Request/Response** :
  - A request for specific data is sent and ultimately leads to a response: REQUEST $\rightarrow$ PARTIAL_RESPONSE (0+) $\rightarrow$ RESPONSE.
  - Usually used to get raw data to exploit.
  - This is the paradigm used for reference data service (//blp/refdata) but real-time data is also available.

- Subscription
  - Continuous update of data that doesn't stop until it's explicitly cancelled.
  - Usually used for fast-changing data.
  - This is the paradigm used for the real time market data (//blp/mktdata), Market Bar (//blp/mktbar) and Custom VWAP (//blp/mktvwap) services.

- Publishing
  For publishing page-based and record-based data as well as consume it.

The different services

- Market Data — *//blp/mktdata*: subscribing to streaming real-time and delayed market data.
- Vwap — *//blp/mktvwap*: subscription-based service used when requesting streaming custom Volume-Weighted Average-Price data.
- Market Bar — *//blp/mktbar*: subscription-based service used when requesting streaming real-time intraday Market Bar data.
- **Reference Data** — ***//blp/refdata***: requesting reference data such as prices, historical/time-series and intraday bars and ticks.

The reference data service

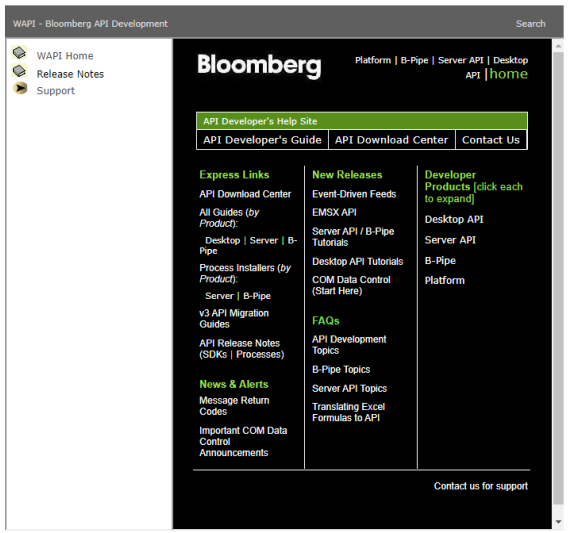
For the purpose of this class, we will only focus on the Request/Response paradigm and the associated Reference Data service.

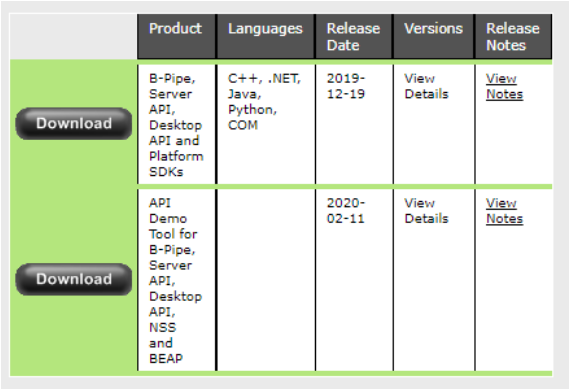
Data type	Request	Service	Comment
Per request data	Reference data Bulk data	ReferenceDataRequest	~BDP ~BDS
Historical data		HistoricalDataRequest	~BDH
Intraday data	Tick data Bar data	IntradayTickRequest IntradayBarRequest	~BDH ~BDH

The SDK package

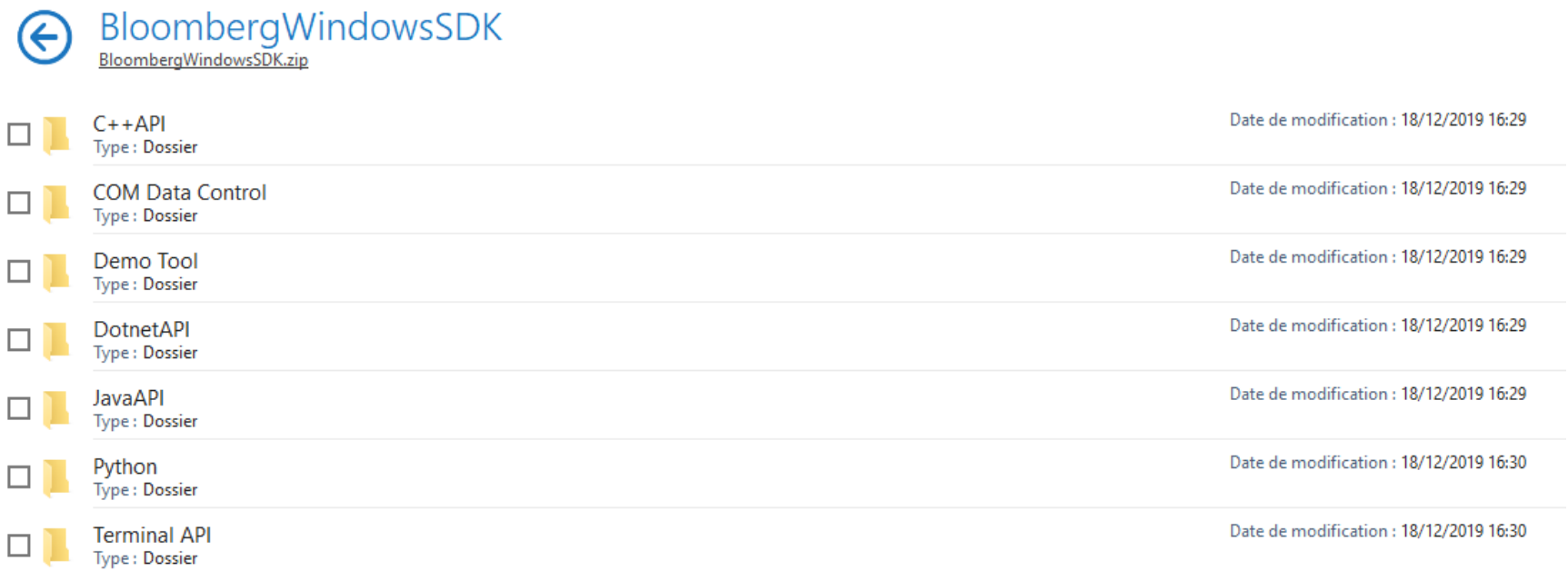
A practical way of diving into the API is to build upon examples made available by Bloomberg.

To do so, you may go to **WAPI** and click on the API Download Center.





In the Windows tab, you may download the B-Pipe, Server, API, Desktop API and Platform SDK package.
For the purpose of this class, we will exclusively work on the Windows version but a Linux version is also available.



The package contains the Bloomberg SDK (software development kit). It consists of several folders essentially classified by programming language. In the next two sections, we will focus on the COM and Python APIs.

Table of contents

VBA programming - best practices

- Code summary

```
Public Function fn_Return(dbl_Price() As Double, Optional str_Rdt_Type As String) As Double()
'---------------------------------------------------------------------------------------
'Summary:
    'This function computes the returns of a matrix of prices.
    'The user may chose the type of returns she wants to compute. If the user specifies
    '"LOG" for logarithmic, the function computes the logarithmic returns, else the function
    'returns arithmetic ones

'Inputs:
    'dbl_Price() : Matrix of prices
    'str_Rdt_Type : Type of returns (optional)

'Output:
    'A matrix of returns

'Author:
    'CF, Université Paris-Dauphine - 2016
'---------------------------------------------------------------------------------------
```

- Comment your code

```
'Declaration of the variables
    Dim dbl_Return() As Double
    Dim lng_NbRdt As Long
    Dim lng_Nbactif As Long
    Dim lng_Cols As Long
    Dim lng_rows As Long

'Variables values assignement
    lng_NbRdt = UBound(dbl_Price(), 1) - 1
    lng_Nbactif = UBound(dbl_Price(), 2)
    ReDim dbl_Return(lng_NbRdt, lng_Nbactif)

'Computation of the returns
   Select Case str_Rdt_Type
        Case "LOG"
            For lng_Cols = 1 To lng_Nbactif
                For lng_rows = 1 To lng_NbRdt
                    dbl_Return(lng_rows, lng_Cols) = Application.WorksheetFunction.Ln((dbl_Price(lng_rows + 1, lng_Cols) / dbl_Price(lng_rows, lng_Cols))
                Next lng_rows
            Next lng_Cols

        Case Else
            For lng_Cols = 1 To lng_Nbactif
                For lng_rows = 1 To lng_NbRdt
                    dbl_Return(lng_rows, lng_Cols) = dbl_Price(lng_rows + 1, lng_Cols) - dbl_Price(lng_rows, lng_Cols)
                Next lng_rows
            Next lng_Cols
    End Select

'Returns the results
    fn_Return = dbl_Return

End Function
```

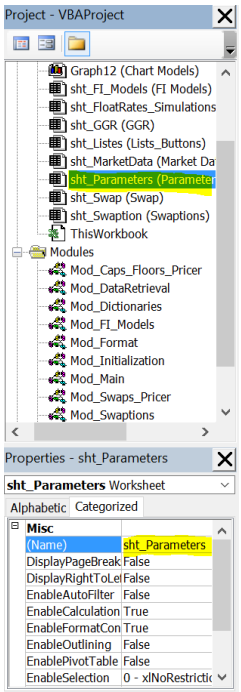
- Name your variables properly and explicitly

Example	Type	Example	Type
int_xxx	Integer	dat_xxx	Date
lng_xxx	Long	fn_xxx	Function
dbl_xxx	Double	sub_xxx	Sub procedure
bool_xxx	Boolean	str_xxx	String
var_xxx	Variant	sht_xxx	Sheet
arr_xxx()	Array	rng_xxx	Range
wb_xxx	Workbook	mod_xxx	Module

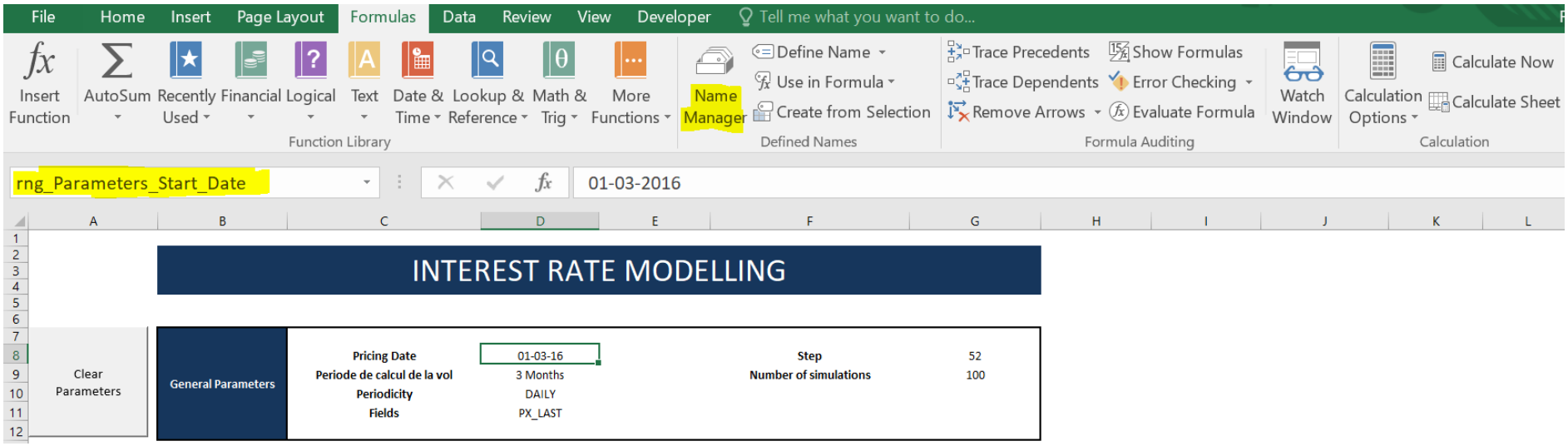
/!\Always use Option Explicit

- Use modules to write your programs
  Do not code in a sheet. If you or another user deletes the sheet, you will lose all your work.
  Use several modules so as to make your code as readable as possible (e.g: mod_Main, mod_Data, mod_Toolbox etc).

- Proper sheet and range manipulation
  Avoid using thisworkbook.sheets("Sheet1"). Your program will encounter an error if a user changes the name of "Sheet1" to something else.
  A safer option consists in naming your sheets in Developer.



You can also name your ranges instead of using Cells(1,1) or Range("A1").



The names you give your worksheet and range can then be used as is in your program:

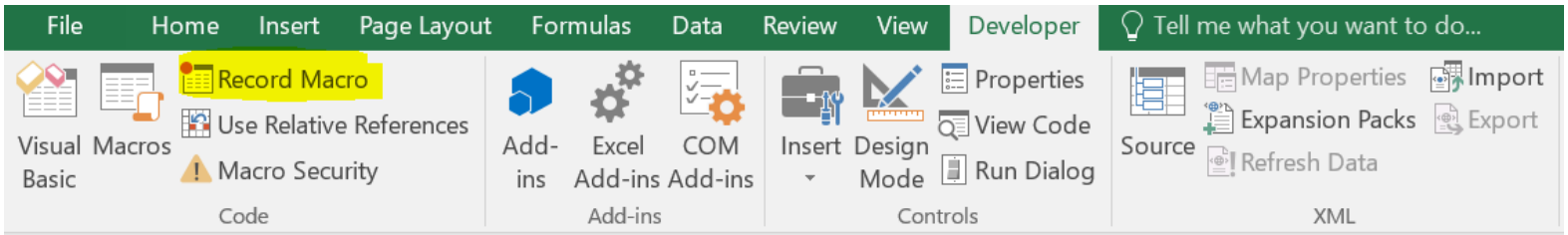
```
'General Parameters
    sht_Parameters.[rng_Parameters_Start_Date].Cells.ClearContents
```

Declaring a variable as a Variant should only be used when you are unaware of what your variable will contain.
If you know the type of your variable, declare it properly.

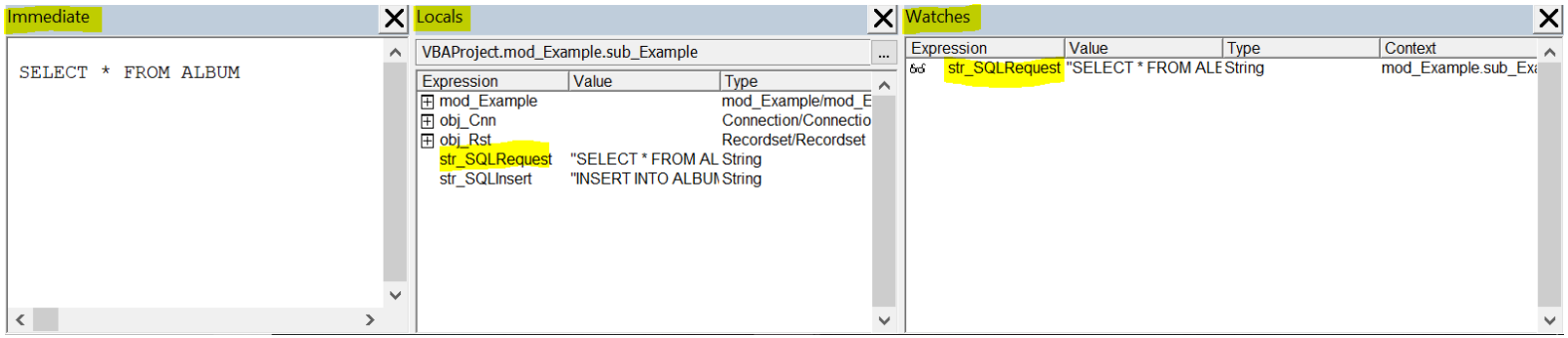
VBA programming - tips and tricks

- Helpful shortcuts
  Alt + F11: opens developer window
  F5: runs the whole macro
  F8: step into (line by line)

- The macro recorder can be helpful
  When needed, the macro recorder can be very helpful. For example, it can give you the code to automatize a particular range format or the creation of a graph or diagram.
  /!\Do not leave the code as is. Chances are many lines of codes are non-essential and for the sake of clarity you should make your code as concise as possible.



- Debugging using Immediate, Locals and Watches windows



Immediate: use in a loop for example via the line debug.print(*name of the variable*). The sequence will be stored even if the macro is done running.

Locals: gives you the value and type of all the variables in the environment; can get a little overwhelming with long macros.

Watches: drag and drop the variable in the window. Use to monitor a few variables only.

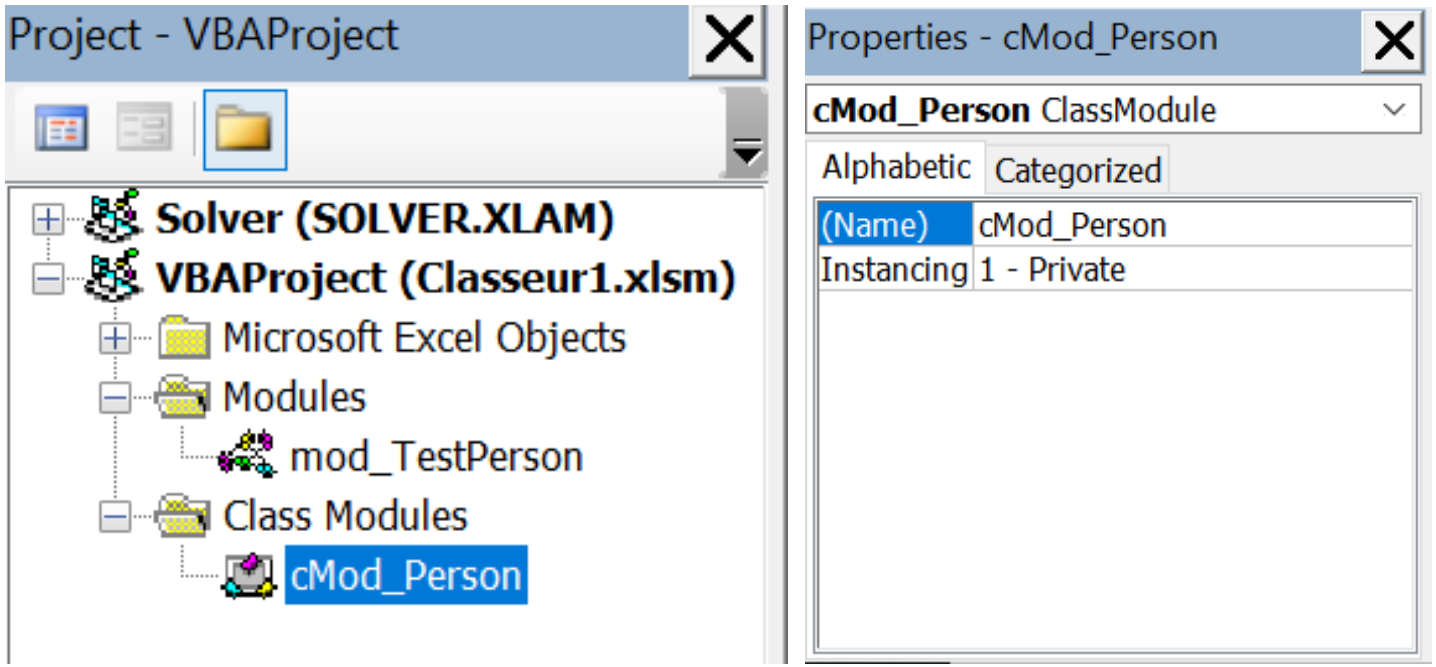
VBA programming - the basics of OOP

OOP is a programming paradigm which revolves around the notion of object. An object can be thought of a pre-constructed and yet customizable brick.

Everyday example :
An everyday example of this paradigm would be the blueprint for a car. The engineer will think of his car with certain attributes: color (red, blue, black etc), door number (3,5 etc), transmission type (stick or automatic) and methods: start engine, speed up, brake, roll the windows down etc. Then when going into production, the cars coming out of the plant will be instances of that blueprint. Some will be 3-door automatic blue cars, whilst others will be red, with 5 doors and have manual transmission. The original concept however is unique.

Excel example :
When coding in VBA, you use objects all the time. Cells, ranges, worksheets and workbooks are all objects as you can see on the Object Explorer window (shortcut F4).

Objects in VBA are created using a class within class modules (= the blueprint) using right-click, insert, class module. The class module can be renamed as seen on the right window (here cMod_TestPerson).



Once the class is created in a class module, you can instantiate a version of the object in a regular module (here mod_TestPerson).

The brick of OOP in VBA:

Attributes:
Attributes in a class are equivalent are equivalent to variables that describe the objects. For example the color of a car is one of its attributes.

Properties:
Either read or write only access to the attributes.

Methods:
Procedures and functions associated with an object.

Events:
Procedures that occur when an object is instantiated or closed.

Getting started

To start using Bloomberg with Excel/VBA, we'll take a look at the examples available in the SDKWindowsPackage we downloaded earlier.

These are part of the COM/ActiveX API.

For this section of the class, we'll solely focus on building a BDH-like function. In order to do, we'll start from the file HistoryExample.xlsm. This example contains the main bricks and will allow us to build upon and revamp the code to make it more flexible.

The only pre-requisite (other than having Bloomberg up and running) to using the API is to select the "Bloomberg API COM 3.5 Type Library" reference in Developer.

Retrieving historical data: DataControl class module

The code for historical data retrieval is contained in a class module named DataControl and is to be instantiated in a "regular" module.

The class module structure:
1/ **Class_Initialize()** & **Class_Terminate()**: class events
These will run whenever the object is instantiated or closed.

2/ **MakeRequest()**: method
This method is used to build the data request, i.e. specify the tickers, the fields, the dates and all the other parameters we need to retrieve and send it.

3/ **session_ProcessEvent()**: method
This method is used to process and extract the data contained in the response message after the data request has been sent.

Retrieving historical data: code session

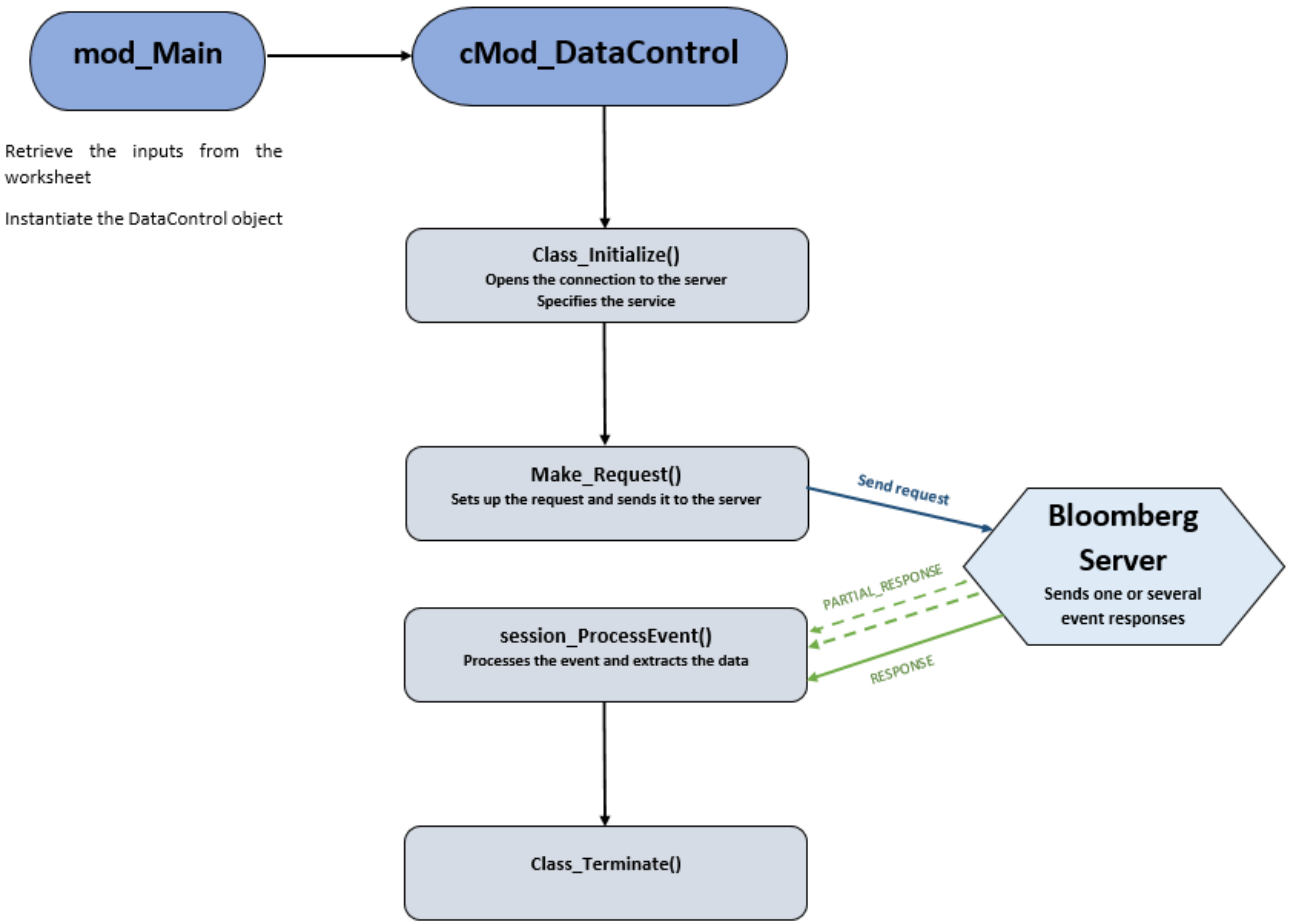
Session goals:

Refurbish the MakeRequest() method to include the desired arguments (tickers, fields, start date, end date + options).

Rework the session_ProcessEvent() to store the data in a more suitable object.

More generally, cleaning up the basis class module code from any reference to the worksheet i.e. making it as OOP-like as possible.

The code is structured as follows:



Message

Partial_Response and Response messages are structured as follows, the goal being to to extract the dates and the values for the fields :

```
HistoricalDataResponse = {
    securityData = {
        security = "GLE FP Equity"
        eidData[] = {
        }
        sequenceNumber = 2
        fieldExceptions[] = {
        }
        fieldData[] = {
            fieldData = {
                date = 2020-10-12
                PX_LAST = 12.712000
                PX_OPEN = 12.51000
            }

            fieldData = {
                date = 2020-10-13
                PX_LAST = 12.210000
                PX_OPEN = 12.711000
            }
        }
    }
}
```

Table of contents

Python crash course

See the Jupyter notebook Python Crash Course for a primer on functions used in this course.

Getting started with the BLP API

- Automatic package installation
  You directly run the following command *pip install blpapi* in a command prompt.

- Manual package installation
  In **WAPI**, go to the **API Download Center** and download the most recent package. You can then extract the blpapi subfolder and copy it in your Python package directory.

- Help can be found:
  - With the live assistance through the terminal's **HELP** functionality
  - Within the different examples included in the Bloomberg Windows SDK zip file.
  - In the Core Developer Guide available **here**

Pro-tip

Using the package requires a C++ compiler that comes with the initial installation. Sometimes, this C++ compiler will become obsolete and you will run into the following error:

```
Mismatch between C++ and Python SDK libraries.

Python SDK version    3.12.1
Found C++ SDK version 3.11.6.1

Download and install the latest C++ SDK from:

    http://www.bloomberg.com/professional/api-library

If a recent version of the C++ SDK is already installed, please ensure that the
path to the library is added to PATH before entering the interpreter.
```

To solve this problem, you need to download the most recent BloombergWindowsSDK zip package from WAPI. Within the *BloombergWindowsSDK\C++API\v.x.x.x.\lib* folder you will find the **blpapi3_32.dll**, **blpapi3_32.lib** and **blpapi3_64.dll**, **blpapi3_64.lib** files.

In a shell window, run *where blpapi3_32.dll* and copy the files to that location. You might need to restart your computer before doing so to make sure the previous versions of the dlls are not still being used.

Retrieving historical data: code session 1

See the first notebook: Python & Bloomberg session 1 BDH.ipynb.

Retrieving reference data : code session 2

See the second notebook: Python & Bloomberg session 2 BDP.ipynb.

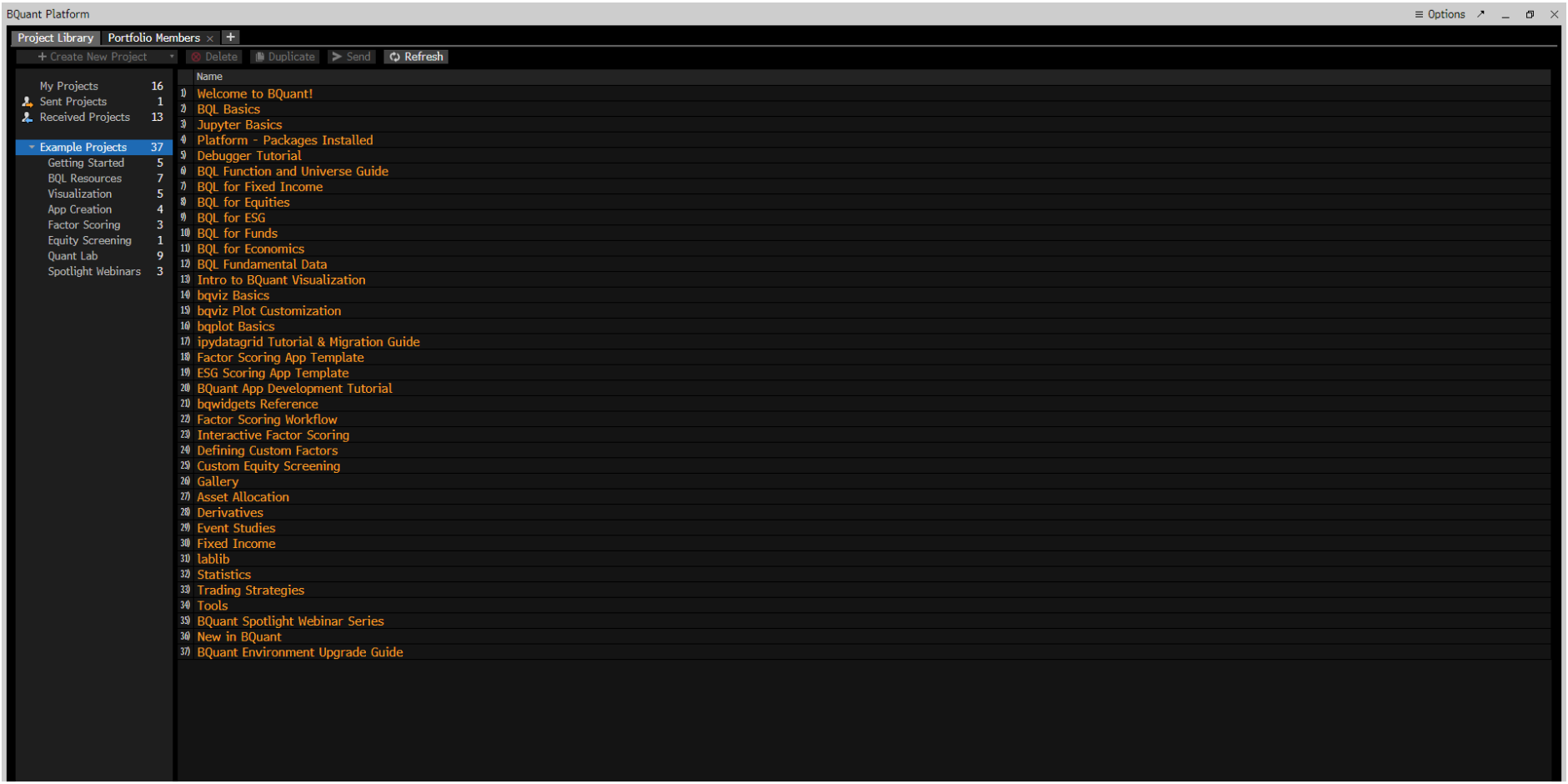
Table of contents

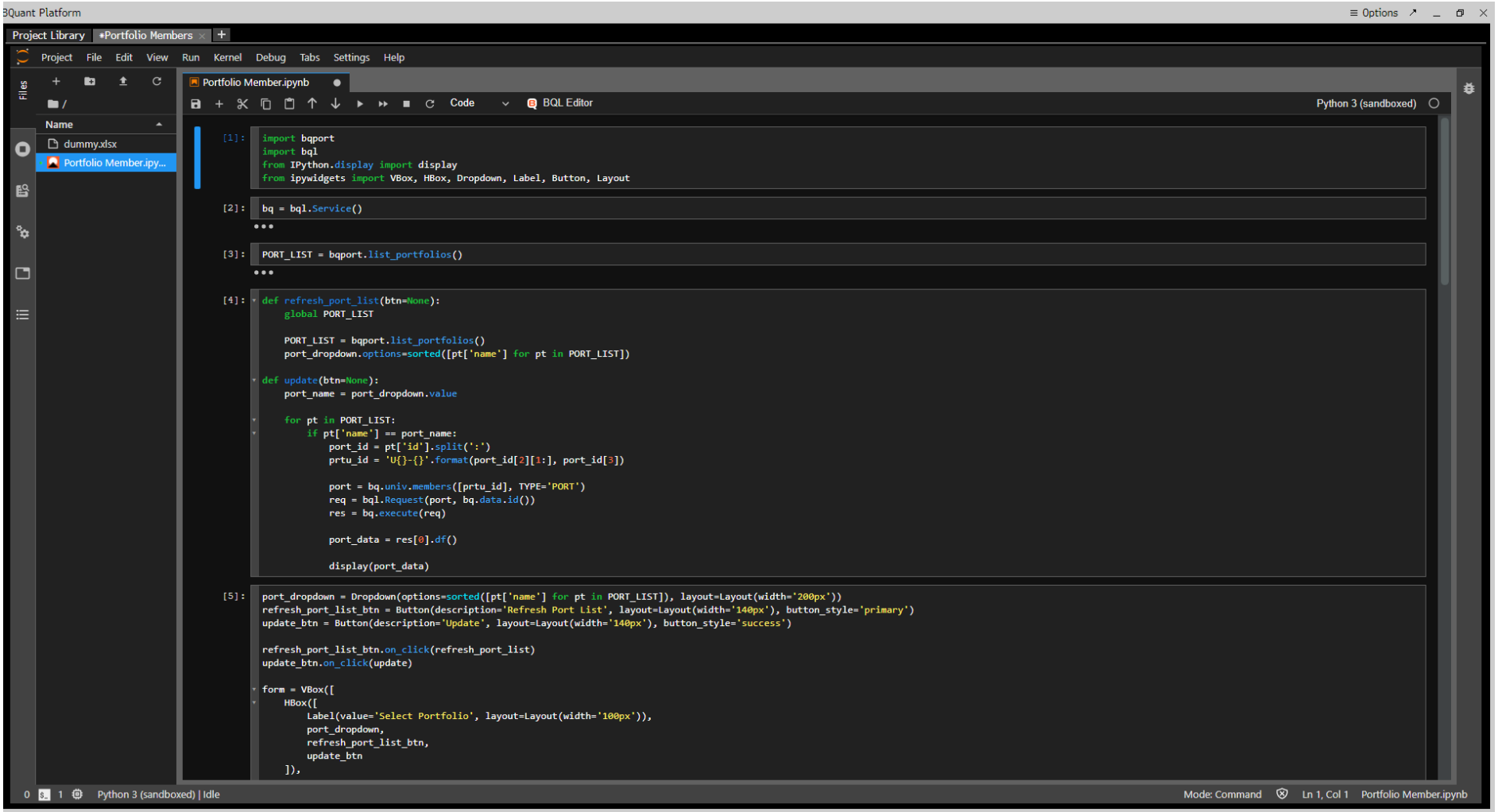
BQuant

BQuant (**BQNT <GO>**) is an interactive development tool for quant-oriented solutions launched by Bloomberg a few years ago and made available to buy-side Bloomberg Anywhere users upon request. It relies on 4 major pillars:

- Python
- Jupyter notebooks
- BQL
- Viz + User Interface

There are several examples of apps and analyses available as starter projects. There are also updated and upgraded regularly by the BQuant team as new features are added.


BQuant Platform
Options
Project Library
Portfolio Members
Create New Project
Delete
Duplicate
Send
Refresh
My Projects 16
Sent Projects 1
Received Projects 13
Example Projects 37
Getting Started 5
BQL Resources 7
Visualization 5
App Creation 4
Factor Scoring 3
Equity Screening 1
Quant Lab 9
Spotlight Webinars 3
Name
1 Welcome to BQuant!
2 BQL Basics
3 Jupyter Basics
4 Platform - Packages Installed
5 Debugger Tutorial
6 BQL Function and Universe Guide
7 BQL for Fixed Income
8 BQL for Equities
9 BQL for ESG
10 BQL for Funds
11 BQL for Economics
12 BQL Fundamental Data
13 Intro to BQuant Visualization
14 bqviz Basics
15 bqviz Plot Customization
16 bqplot Basics
17 ipydatagrid Tutorial & Migration Guide
18 Factor Scoring App Template
19 ESG Scoring App Template
20 BQuant App Development Tutorial
21 bqwidgets Reference
22 Factor Scoring Workflow
23 Interactive Factor Scoring
24 Defining Custom Factors
25 Custom Equity Screening
26 Gallery
27 Asset Allocation
28 Derivatives
29 Event Studies
30 Fixed Income
31 lablib
32 Statistics
33 Trading Strategies
34 Tools
35 BQuant Spotlight Webinar Series
36 New in BQuant
37 BQuant Environment Upgrade Guide


BQuant Platform
Options
Project Library
*Portfolio Members
Project File Edit View Run Kernel Debug Tabs Settings Help
Files
Name
dummy.xlsx
Portfolio Member.ipy...
Portfolio Member.ipynb
Code
BQL Editor
Python 3 (sandboxed)
[1]:
import bqport
import bql
from IPython.display import display
from ipywidgets import VBox, HBox, Dropdown, Label, Button, Layout
[2]:
bq = bql.Service()
[3]:
PORT_LIST = bqport.list_portfolios()
[4]:
def refresh_port_list(btn=None):
global PORT_LIST
PORT_LIST = bqport.list_portfolios()
port_dropdown.options=sorted([pt['name'] for pt in PORT_LIST])
def update(btn=None):
port_name = port_dropdown.value
for pt in PORT_LIST:
if pt['name'] == port_name:
port_id = pt['id'].split(':')
prtu_id = 'U{}-{}'.format(port_id[2][1:], port_id[3])
port = bq.univ.members([prtu_id], TYPE='PORT')
req = bql.Request(port, bq.data.id())
res = bq.execute(req)
port_data = res[0].df()
display(port_data)
[5]:
port_dropdown = Dropdown(options=sorted([pt['name'] for pt in PORT_LIST]), layout=Layout(width='200px'))
refresh_port_list_btn = Button(description='Refresh Port List', layout=Layout(width='140px'), button_style='primary')
update_btn = Button(description='Update', layout=Layout(width='140px'), button_style='success')
refresh_port_list_btn.on_click(refresh_port_list)
update_btn.on_click(update)
form = VBox([
HBox([
Label(value='Select Portfolio', layout=Layout(width='100px')),
port_dropdown,
refresh_port_list_btn,
update_btn
]),
0 1 Python 3 (sandboxed) | Idle
Mode: Command
Ln 1, Col 1
Portfolio Member.ipynb

Demos

In **BQIQ <GO>**, you can find several videos and past webinars demonstrating the use of BQuant with practical examples.

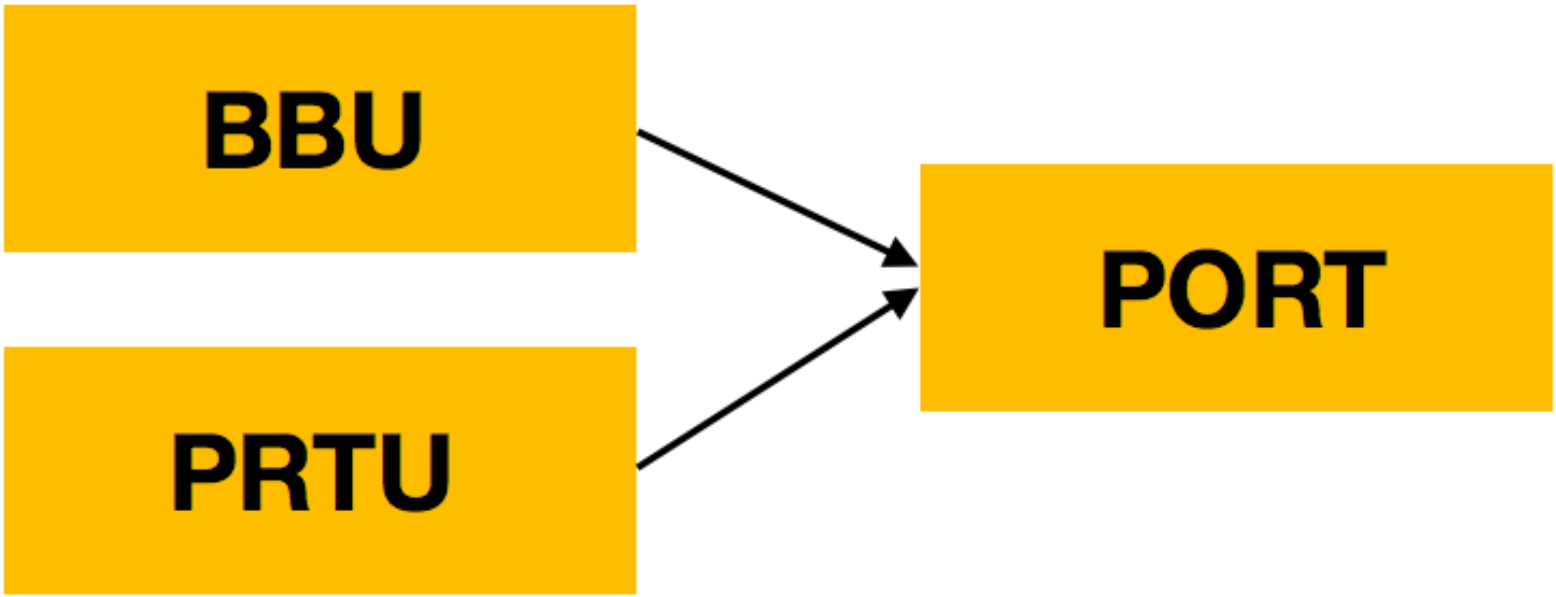
Here are a couple of interesting videos:

- BQuant Promo Video (in overview)
- BQuant Spotlight Consistent Dividend Growth: screening for Quality Companies (in case studies)

Table of contents

Loading portfolios into Bloomberg PORT

Loading portfolios into Bloomberg is achieved through the PRTU-BBU-PORT trinity:



1. **PRTU** for portfolio administration and manual creation
2. **BBU** for portfolio upload via file
3. **PORT** for portfolio monitoring

PRTU: Portfolio administration window

PRTU <GO> is the function that allows you to:
- Create + manage portfolios and benchmarks
- Share portfolios with other users

There are three sections to the portfolio administration window:

- Side bar:
  - Where you can manage and create **Portfolios**, **Portfolio Groups** and **Benchmarks** (see below).
  - You can also set up **Proxies** for certain security exceptions (for example a non-tradable index with an ETF).
  - You can also set up **Profiles** to specify the calculation methods for several fields (e.g. pricing source, risk models to use, VaR settings etc) as well as view Bloomberg default ones.
  - Finally, in **Port Views**, you get to customize a view (columns and fields to be displayed) you'll ultimately see in **PORT <GO>**.

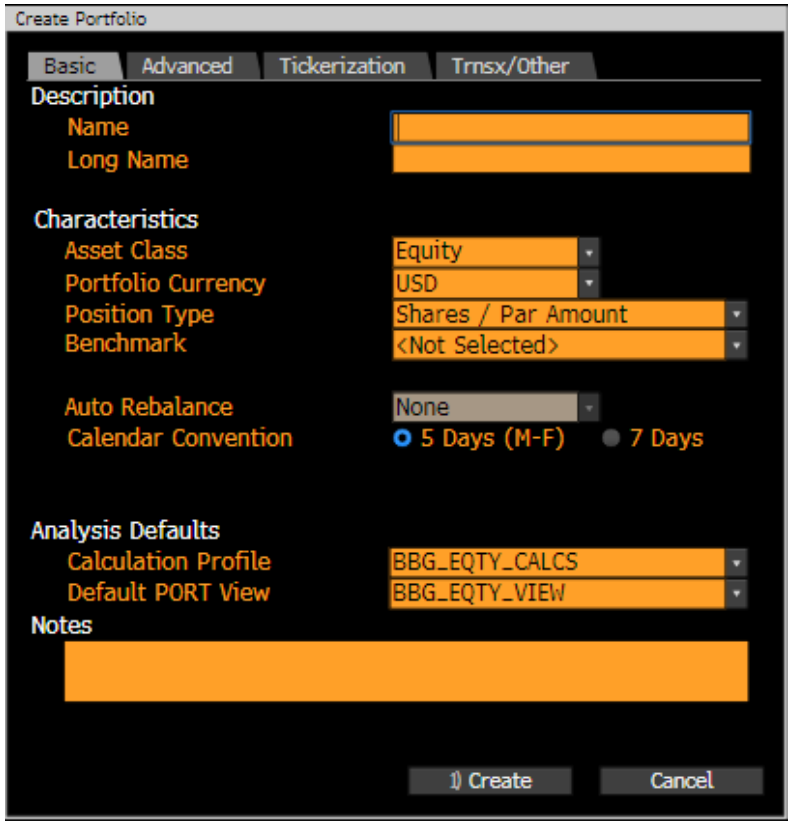
- Tool bar: To create, remove, share or export Portfolios, Benchmarks, Profiles and Views.

- Administration options: gives you an overview of the different Portfolios, Benchmarks, Profiles and Views.

PRTU: Portfolio administration window

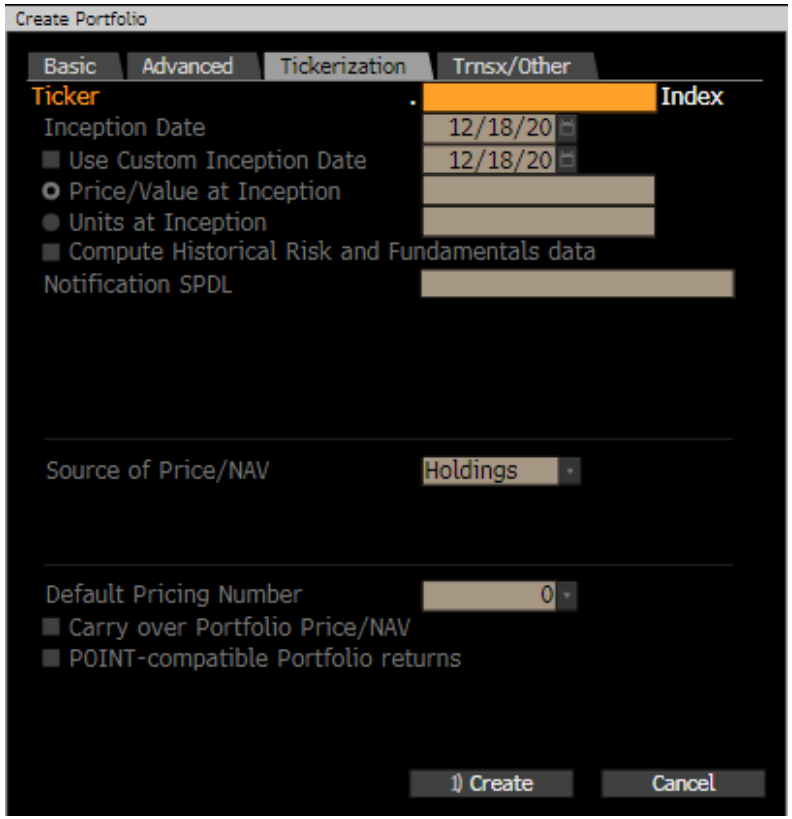
Creating a portfolio

To create a portfolio, you need to click on Create.



- Name (unique, mandatory field) & Long Name
- Asset Class: Equity, Balanced, FI (for Fixed Income) or Fund of Funds
- Portfolio Currency: designates the currency in which the portfolio and main cash position will be denominated in.
- Position Type:
  - Shares / Par amount: Positions are entered as number of shares or nominal amount
  - Fixed Weight: Positions are entered as portfolio weights. The same weights from the last rebalancing date and the same are used everyday until the next rebalancing occurs
  - Drifting Weight: Positions are entered as portfolio weights at each rebalancing date. Between two rebalancing dates, the weights drift and reflect the market fluctuations of the assets.
- Benchmark: allows you to select a custom benchmark. You can create a benchmark before creating your portfolio or afterwards and edit the portfolio.
- Calendar Convention: whether you want to display your portfolio on 5 or 7 days
- Calculation Profile: allows you to select standard or custom Calculation Profiles
- Default PORT View: allows you to select standard or custom PORT views

Tickerization:

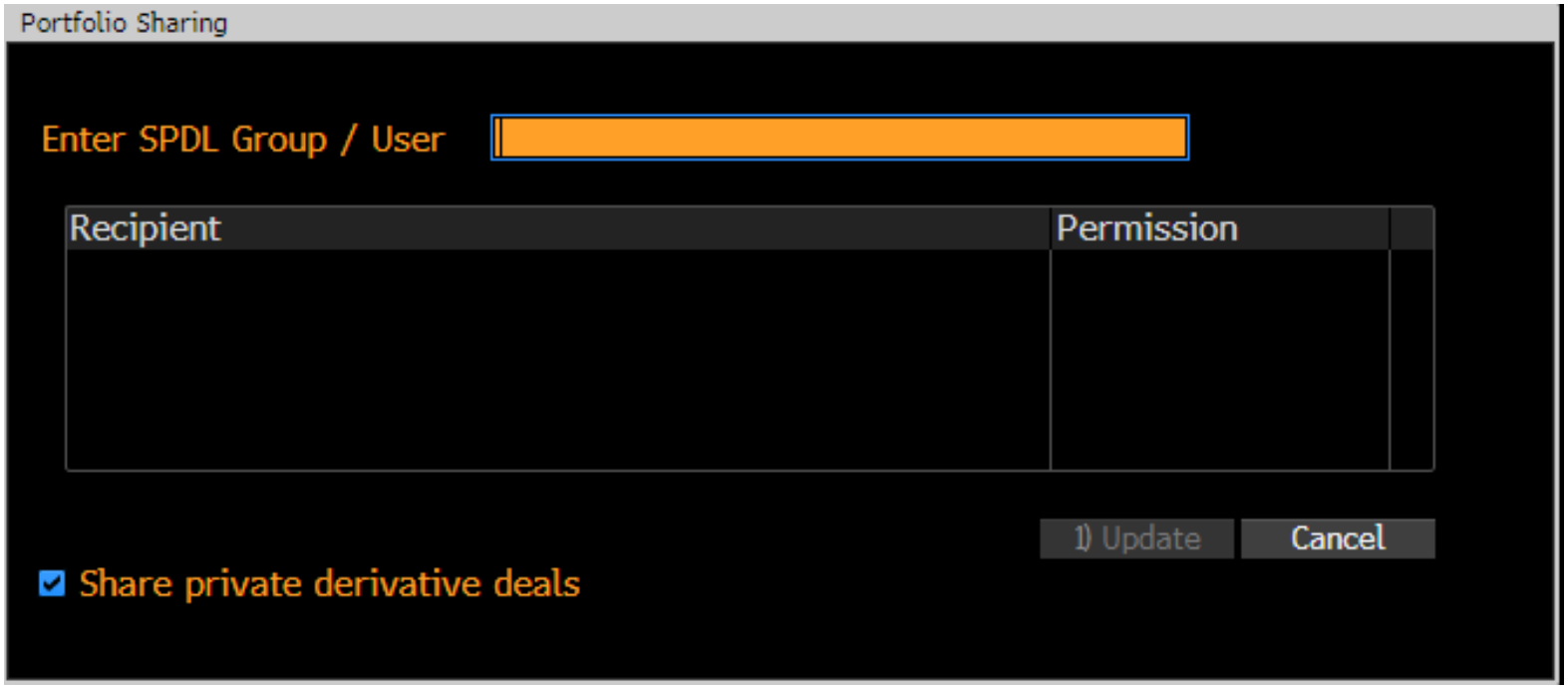


You can tickerize a portfolio i.e. assign a ticker to it. This allows you to add to open the portfolio like any other security as well as access it within the API or in your launch pad.
Tickerization is not an instant process, it requires a few minutes to be completed. You should receive a Bloomberg Mail once the ticker is created.

PRTU: Portfolio administration window

Sharing a portfolio

In the toolbar, you will find a button that allows you to share a portfolio

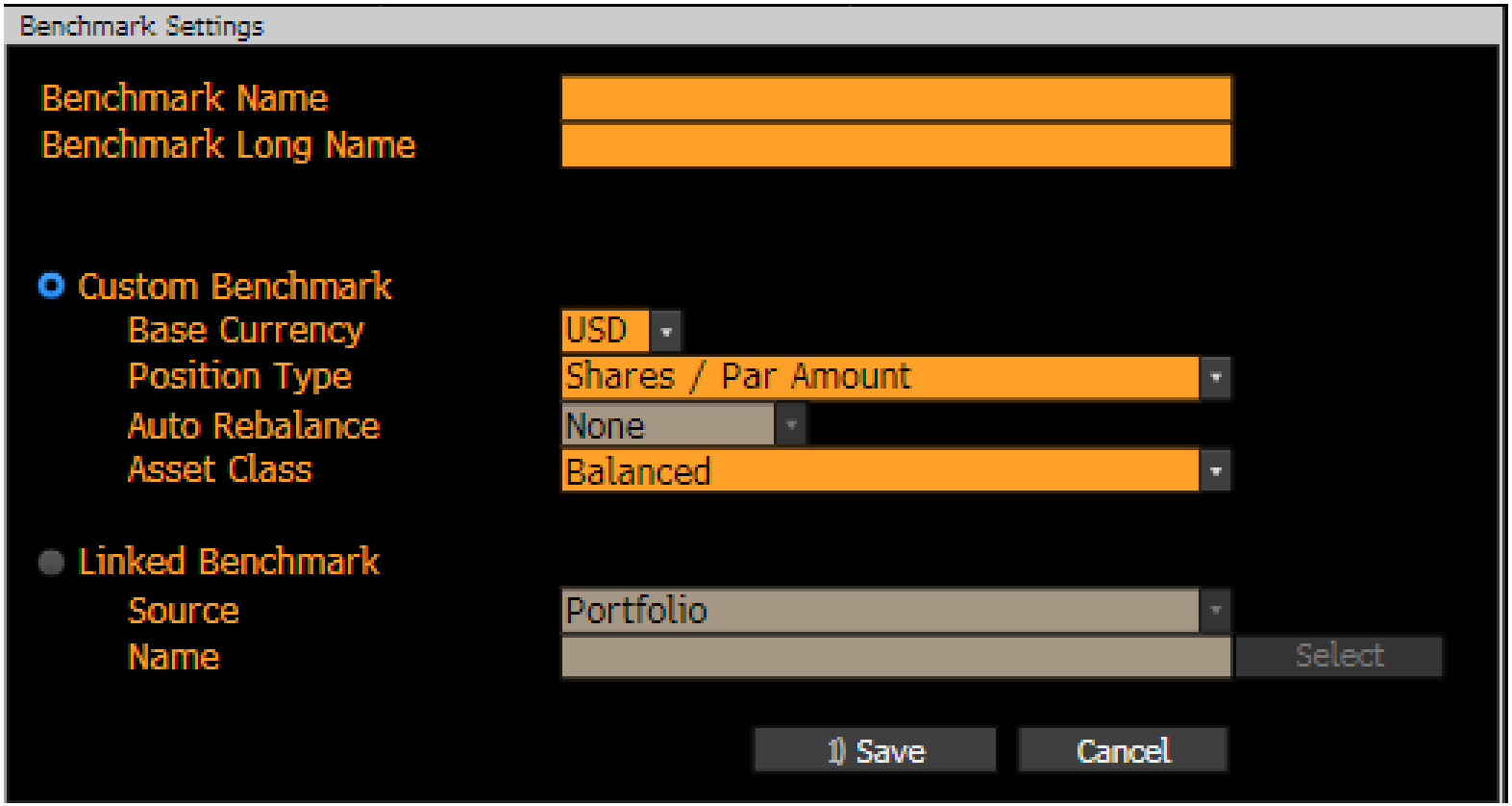


A helpful function to know your colleagues' user id is to run **IAM <GO>** and retrieve their uuid.

PRTU: Portfolio administration window

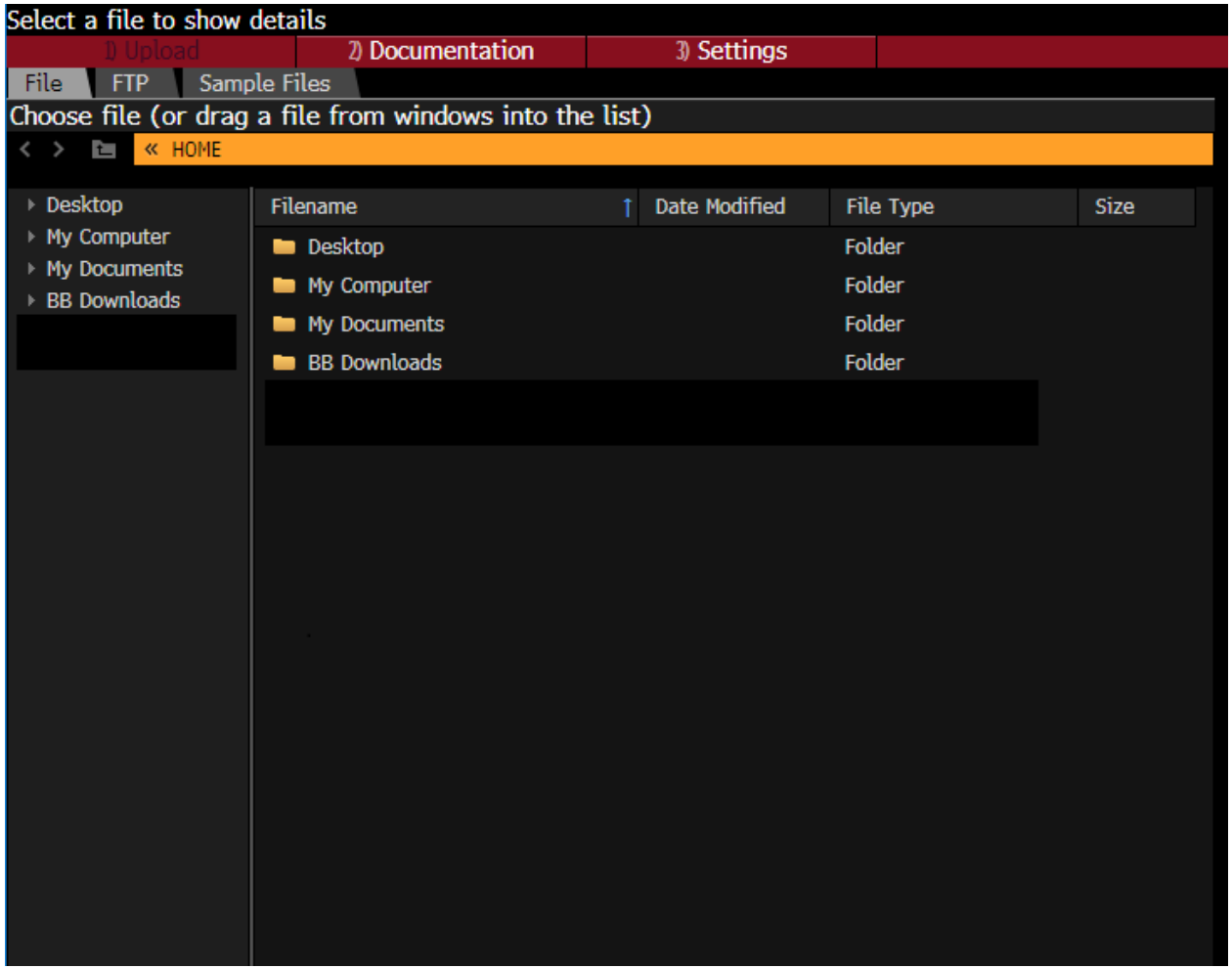
Creating a benchmark

In the toolbar, you will find a button that allows you to share a portfolio



BBU: Bloomberg Uploader

Bloomberg Uploader (**BBU <GO>**) is the handy tool used for uploading portfolios into Bloomberg automatically.
Max upload: 50 portfolios daily



You can upload a portfolio via two canals:

1. **File**
   You directly produce the file that shall be uploaded in BBU. The file is directly picked up from a dedicated folder on your computer or in the Personal File Manager.
   This is ideal for developing portfolio projects, non-live portfolios and experimenting.

2. **FTP: file transfer protocol**
   FTP is useful for long-established, live portfolios. Setting it up requires the creation of a route between the data source (usually the fund valuation entity) and Bloomberg. This will naturally not be studied in this class.

BBU: Bloomberg Uploader

File format

Several examples of files are available in 'Samples Files' section. We'll see how to build these files in class.

BBU: Bloomberg Uploader

Upload monitor

After you have created your file and saved it on you computer, here are the steps to follow:

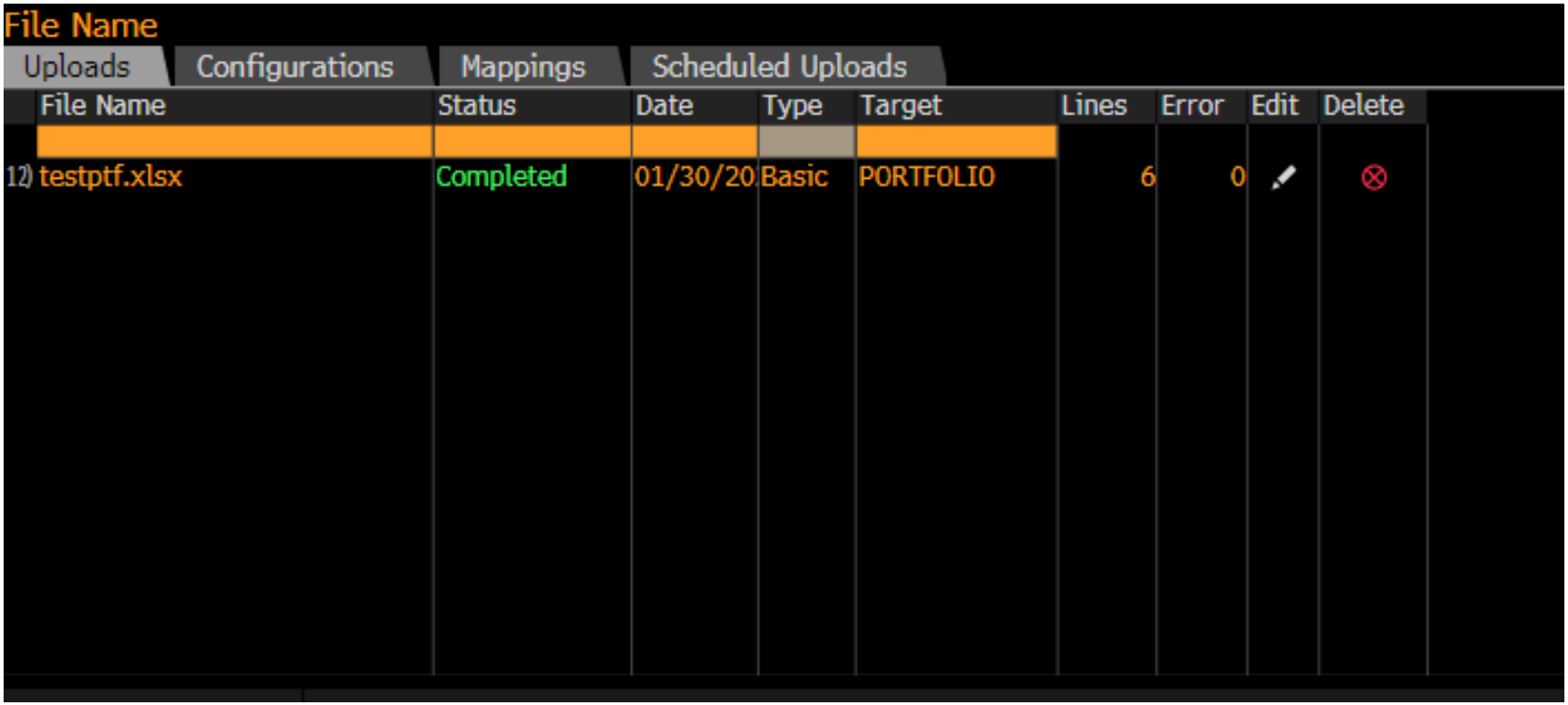
1- Upload the file (top left-hand button after selecting it):
The uploads tab then displays the Status of a portfolio uploaded to BBU, as well as the Date, Type, and the number of Errors that may have occurred during the upload.

2- Check the number of errors:
If you have encountered errors, you'll need to check your file format.

3- File mapping:
If the file is ready to map, you can access the mapping editor. Mapping consists in specifying which column from your file corresponds to Bloomberg fields.
If you always use the same file formats for BBU, the Mapping tab allows you to select predefined mappings.


<Back> to Return
10) Actions
Edit Mapping
View Results
View Errors
View Alerts
Input File
Result File
File
File Setup
Select Worksheet
'Feuil1$'
Use existing mapping
testptf.xlsx
Delimiters
Comma
Header line in file
1
Start data upload at line
2
Ignore lines beginning with
* - Mandatory Fields
Portfolio Name
ISIN
Security Name
Position
Mkt Px
Date
2. IEFTESTPTF2 US0028241000 ABBOTT LABORATORIES 100 44.15 12/18/2020
3. IEFTESTPTF2 US00817Y1082 AETNA INC 100 88.28 12/18/2020
4. IEFTESTPTF2 US0184901025 ALLERGAN INC 100 211.19 12/18/2020
5. IEFTESTPTF2 US03027X1000 AMERICAN TOWER CORP 100 98.17 12/18/2020
6. IEFTESTPTF2 US0325111070 ANADARKO PETROLEUM CORP 100 75.32 12/18/2020
7. IEFTESTPTF2 US1156372096 BROWN-FORMAN CORP-CLASS 100 87.39 12/18/2020
8.

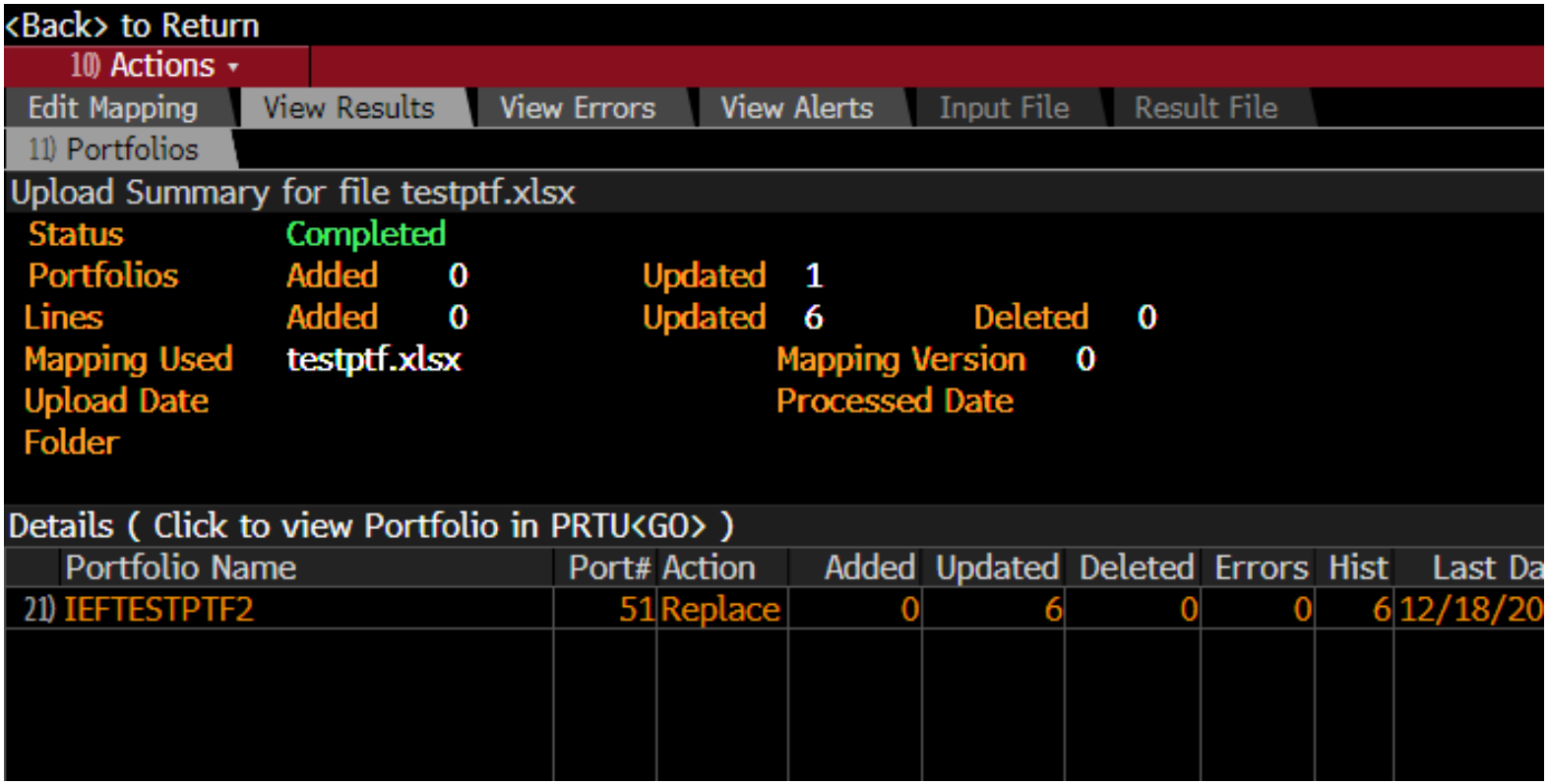
Once you have solved the errors and completed the mapping, the upload is completed. You can click on the line to access information the file.



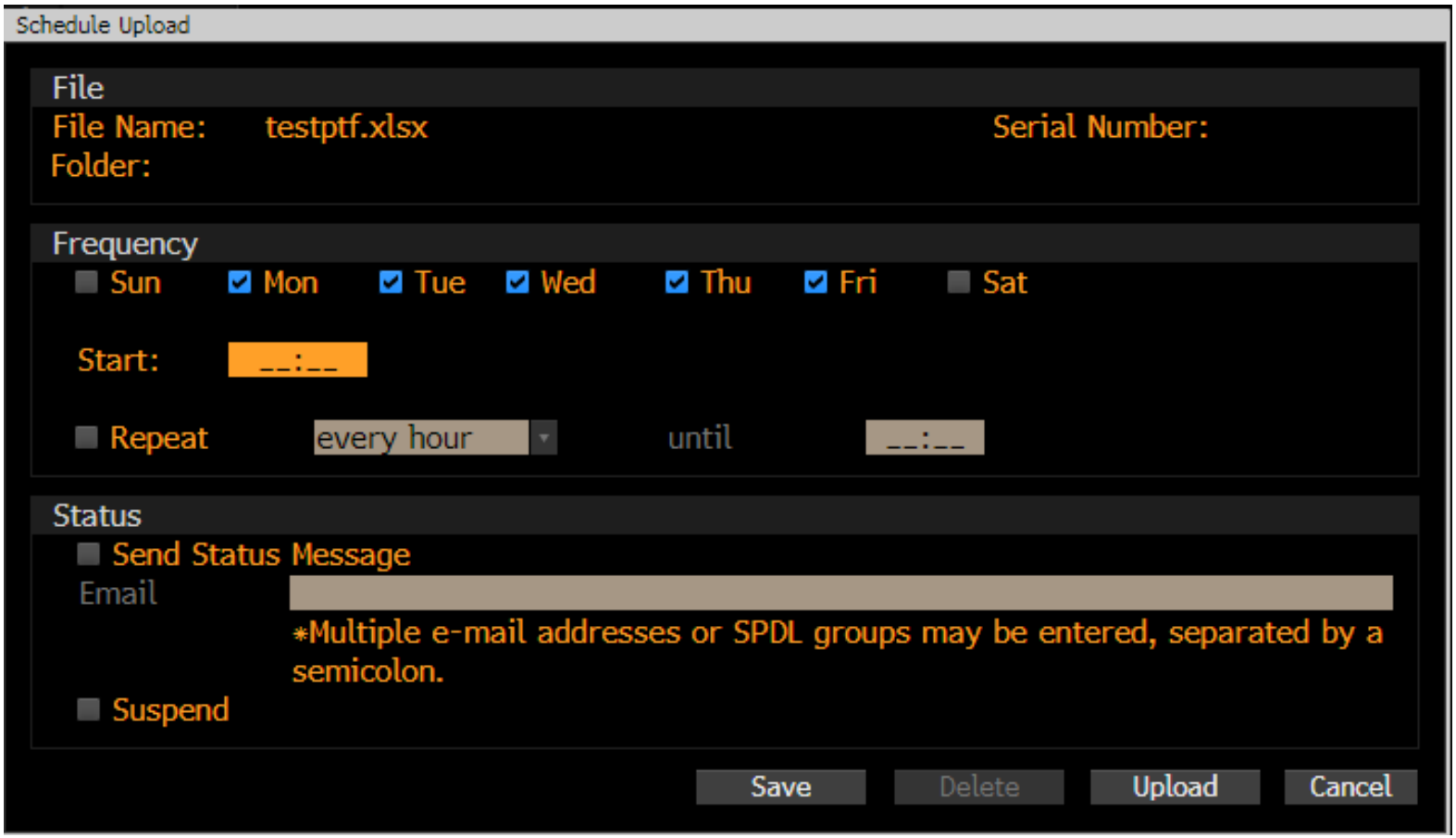
BBU: Bloomberg Uploader

Upload schedule

The Bloomberg upload schedule allows you to automatize the upload of your files into PORT via BBU. To plan uploads, you'll need to go on the Views Results tab, then click on the Actions button and select Schedule Uploads.

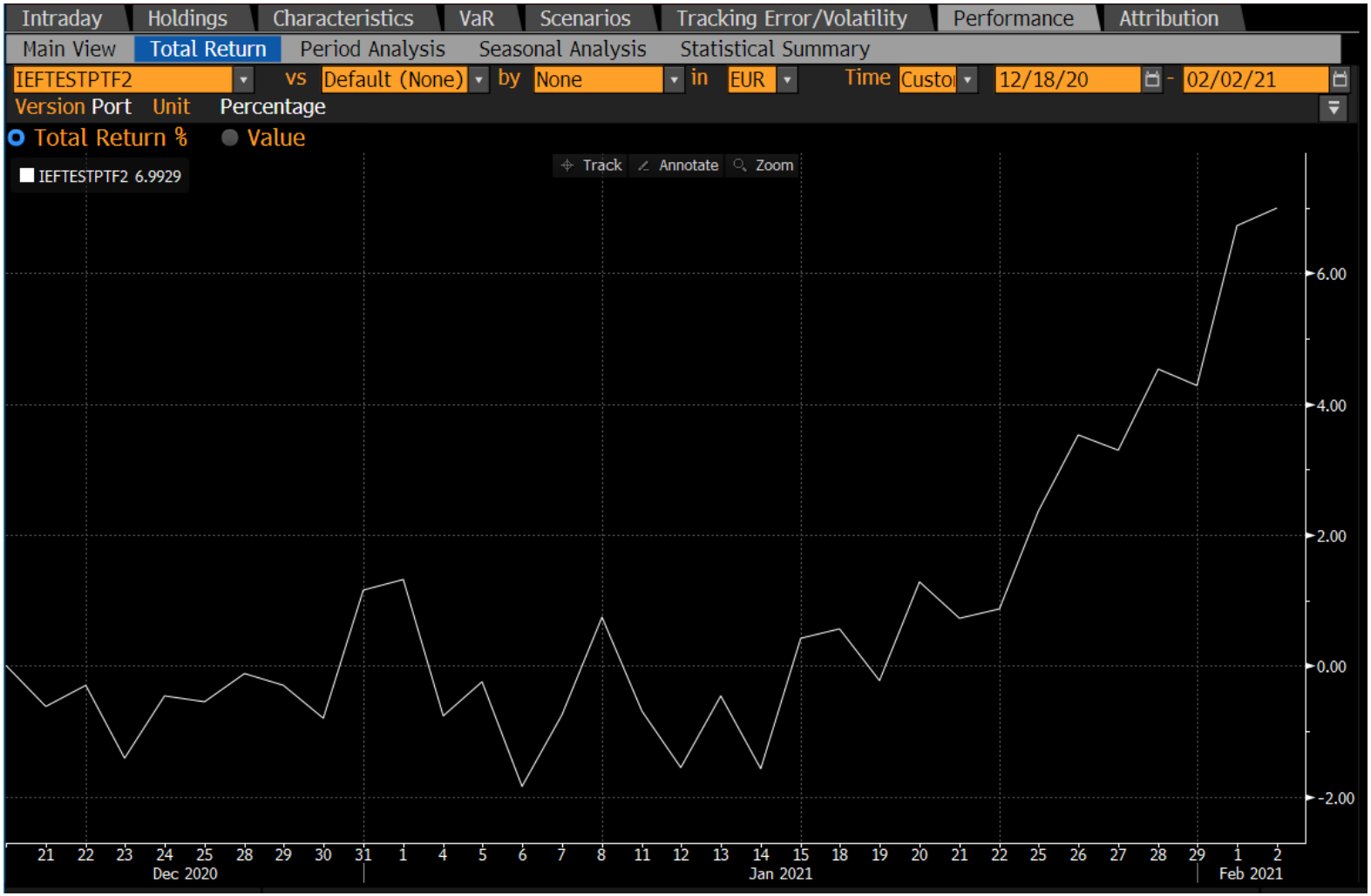


You can specify the day, time and frequency at which files should be uploaded. The uploader will retrieve the files from the same location where the file was first uploaded. You can also program email alerts to inform you of the upload status.



PORT


Intraday
Holdings
Characteristics
VaR
Scenarios
Tracking Error/Volatility
Performance
Attribution
Main View
Allocation
IEFTESTPTF2
vs
Default (None)
by
None
in
EUR
As of
12/18/20
Date
Trend
Name
#
% Wgt
Mkt Val
Pos
Px Close
Crncy
ISIN
Ticker
IEFTESTPTF2 3 100.00 33,502
ABBOTT LABORATORIES 26.58 8,906 100.00 108.97 USD US0028241000 ABT US
AMERICAN TOWER CORP 54.04 18,105 100.00 221.52 USD US03027X1000 AMT US
BROWN-FORMAN CORP-CLASS B 19.37 6,491 100.00 79.42 USD US1156372096 BF/B US
Holdings as of: 12/18/2020
(!) 3 Notices
Submitted at: 15:00:17
Zoom
90%


Intraday
Holdings
Characteristics
VaR
Scenarios
Tracking Error/Volatility
Performance
Attribution
Main View
Total Return
Period Analysis
Seasonal Analysis
Statistical Summary
IEFTESTPTF2
vs
Default (None)
by
None
in
EUR
Time
Custo
12/18/20
-
02/02/21
Version Port
Unit
Percentage
Total Return %
Value
IEFTESTPTF2 6.9929
Track
Annotate
Zoom
6.00
4.00
2.00
0.00
-2.00
21 22 23 24 25 28 29 30 31 1 4 5 6 7 8 11 12 13 14 15 18 19 20 21 22 25 26 27 28 29 1 2
Dec 2020
Jan 2021
Feb 2021

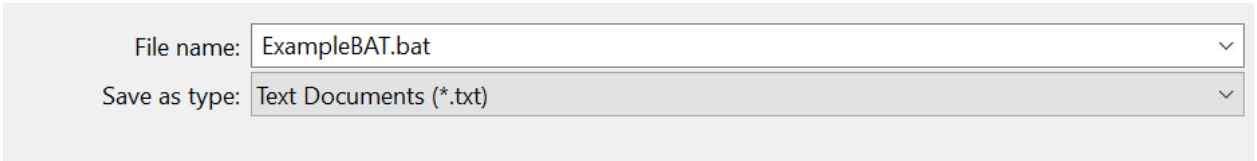

Intraday
Holdings
Characteristics
VaR
Scenarios
Tracking Error/Volatility
Performance
Attribution
Main View
VaR Comparison
Distribution
VaR Simulations
Factor Breakdown
IEFTESTPTF2
vs
Default (None)
by
None
in
EUR
As of
02/02/21
Model
MAC2
Bloomberg Ris
Unit
P&L
CLvl
95%
Scaling
1 day
Name
VaR (MC)
CVaR (MC)
Mkt Val
Pos
% Wgt
Partial VaR (MC)
Marginal VaR (MC)
IEFTESTPTF2 970.30 1,279.28 35,697 100.00
ABBOTT LABORATORIES 272.45 362.95 9,997 100.00 28.01 -188.86 2.40
AMERICAN TOWER CORP 675.57 896.04 19,634 100.00 55.00 -586.79 3.18
BROWN-FORMAN CORP-CLASS B 177.35 236.81 6,066 100.00 16.99 -109.73 1.75
Holdings as of: 12/18/2020
(!) 4 Notices
Submitted at: 15:01:12
Zoom
90%

Automatizing Uploads to PORT

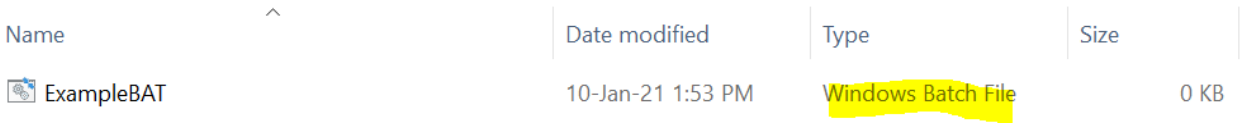
Creating BAT files

BAT files (for Batch) are text files with a **.bat** extension that are used to run command line in the Command Prompt.
In our case, the can be useful to run Python scripts automatically via the Task Scheduler.

To create a BAT file:
1- Open a text editor (e.g. Notepad on Windows)
2- Save it in its dedicated folder with the extension .bat



You now have an empty Windows Batch File:



Doubling clicking on the icon launches the execution of the file instructions (at this stage this will cause an error). To edit a bat file, right click and press Edit.

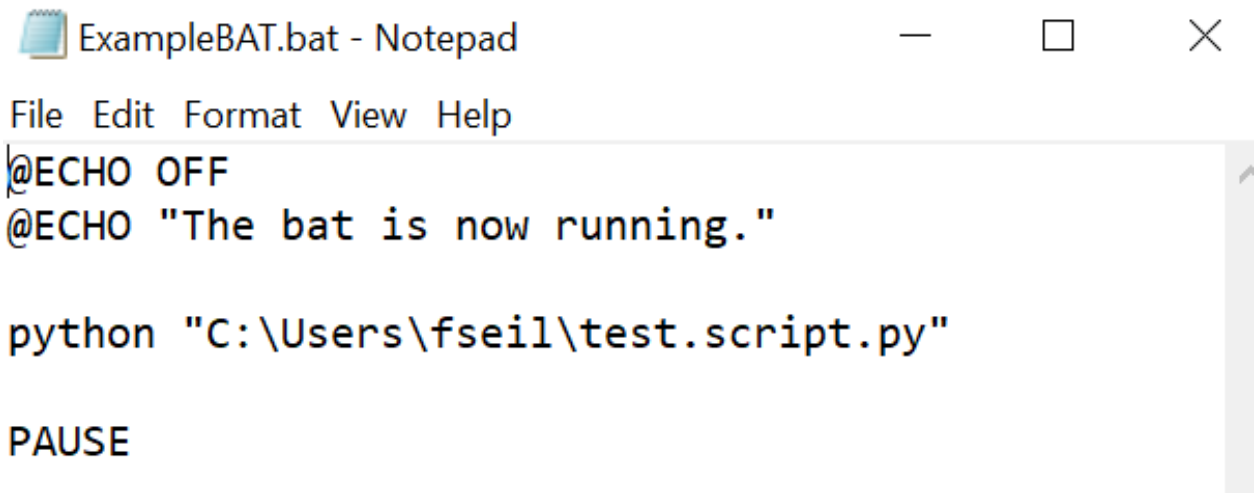
Automatizing Uploads to PORT

Anatomy of a BAT file

Some key words:

1. **@ECHO OFF** clears the console prompt
2. **ECHO** allows you to type some text in the console prompt
3. **Instructions**
4. **PAUSE**, when added at the end of your BAT file, will keep the console open after

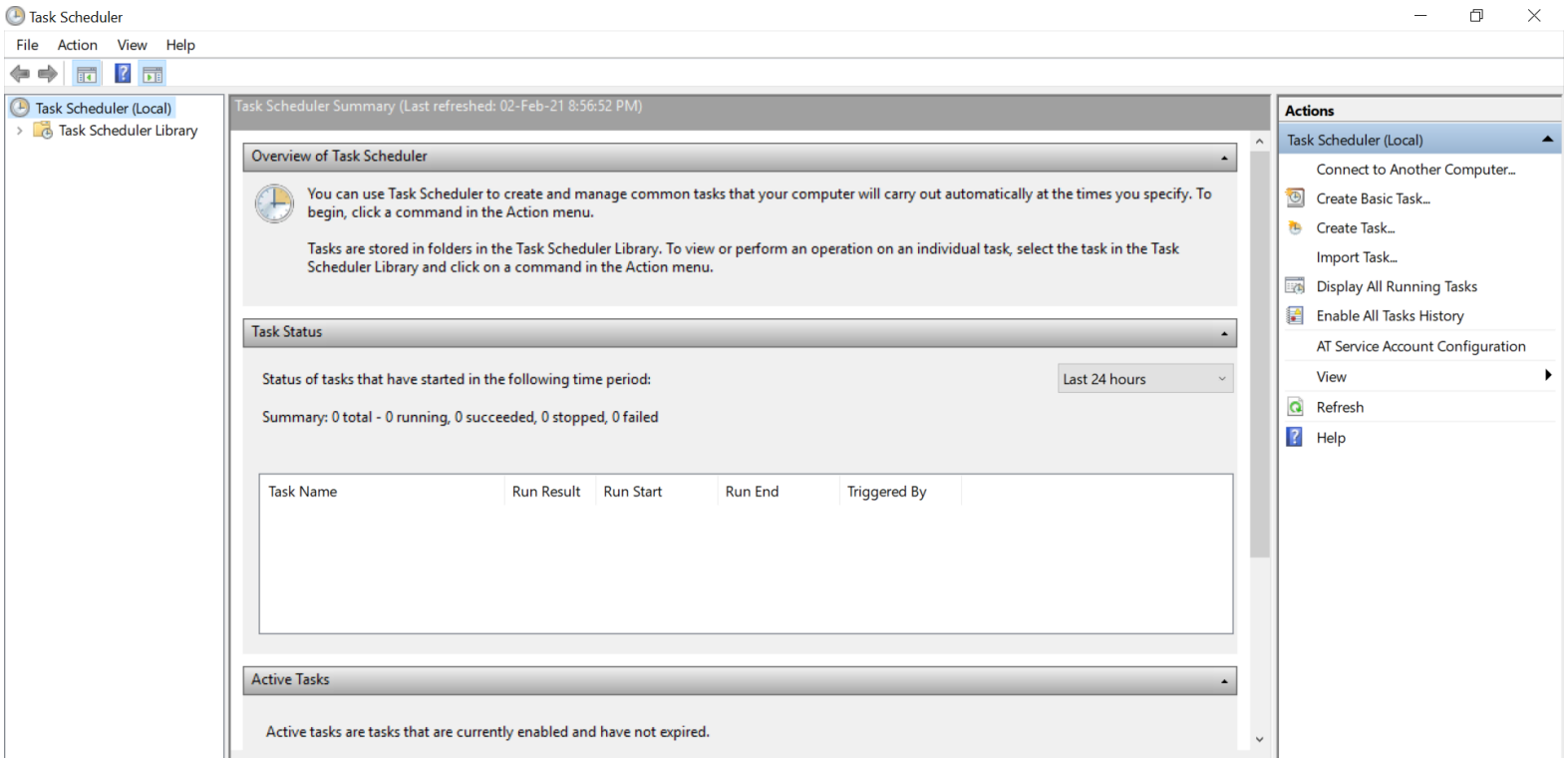
Assuming you are running a Python script, the bat file could look like this:



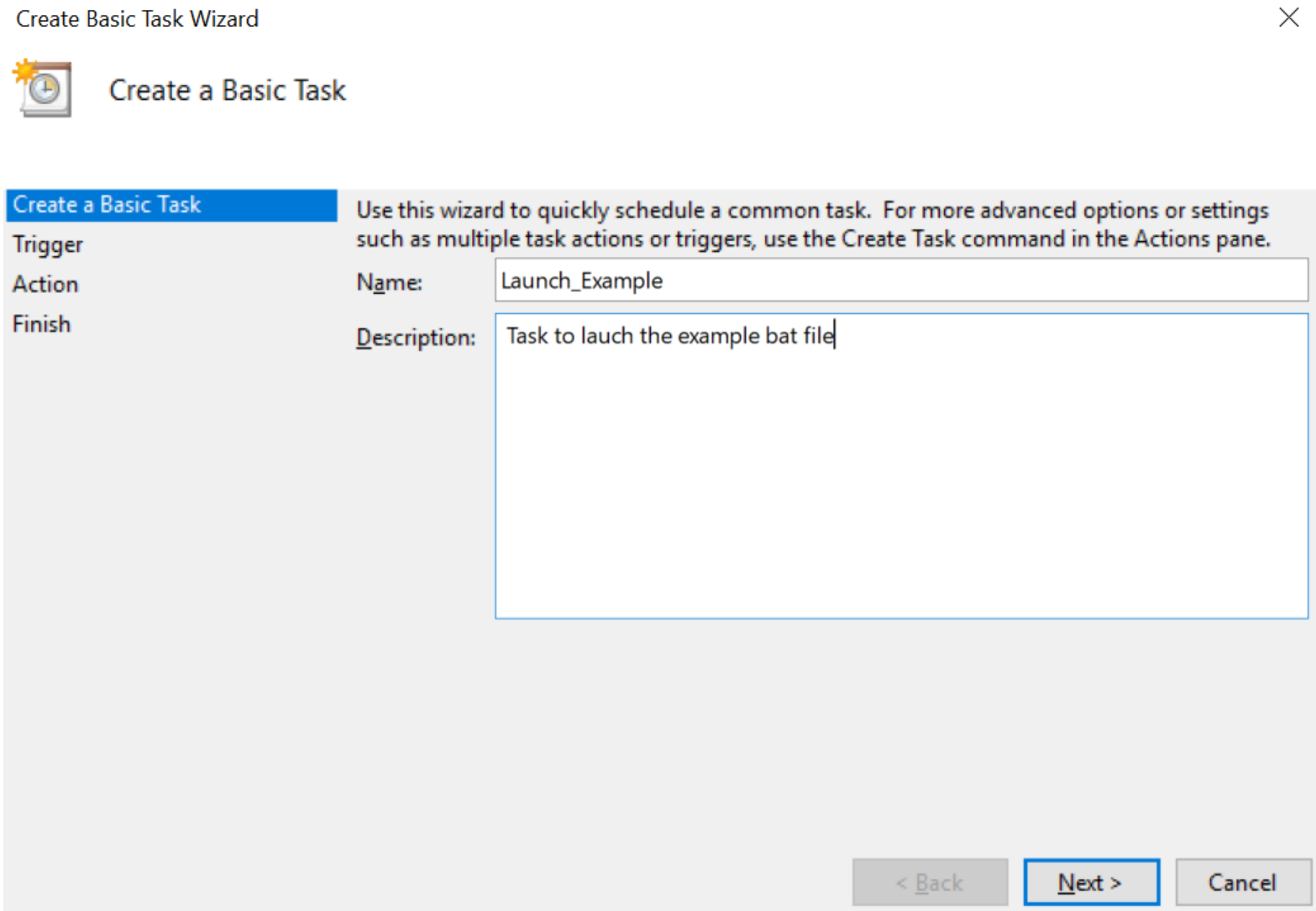
Automatizing Uploads to PORT

Scheduling a task in the Task Manager

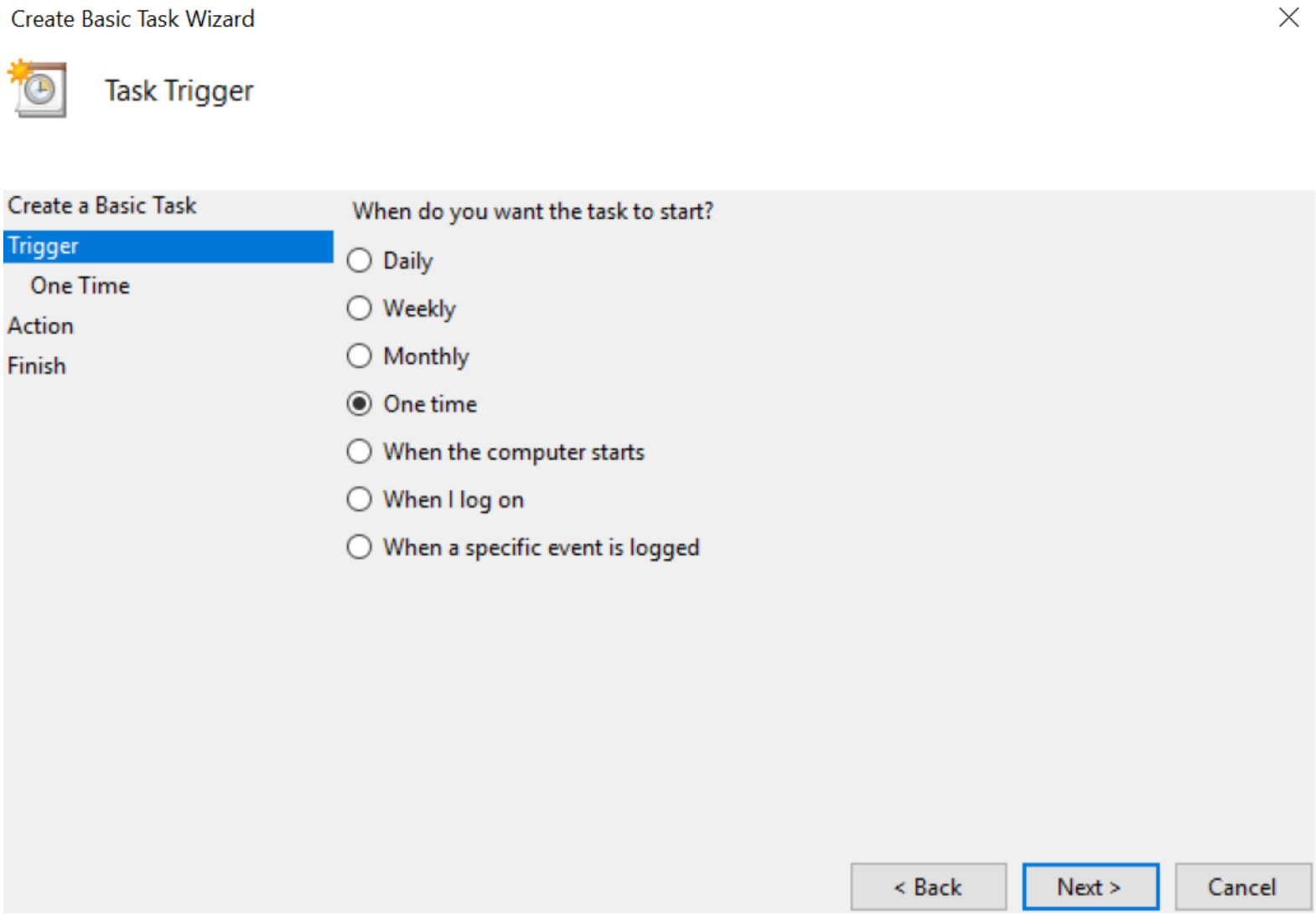
Using Window's Task Scheduler, you can program the execution of your BAT file at a desired frequency. To do so, open the **TaskManager** app and in the right-hand sidebar select Create Basic Task.



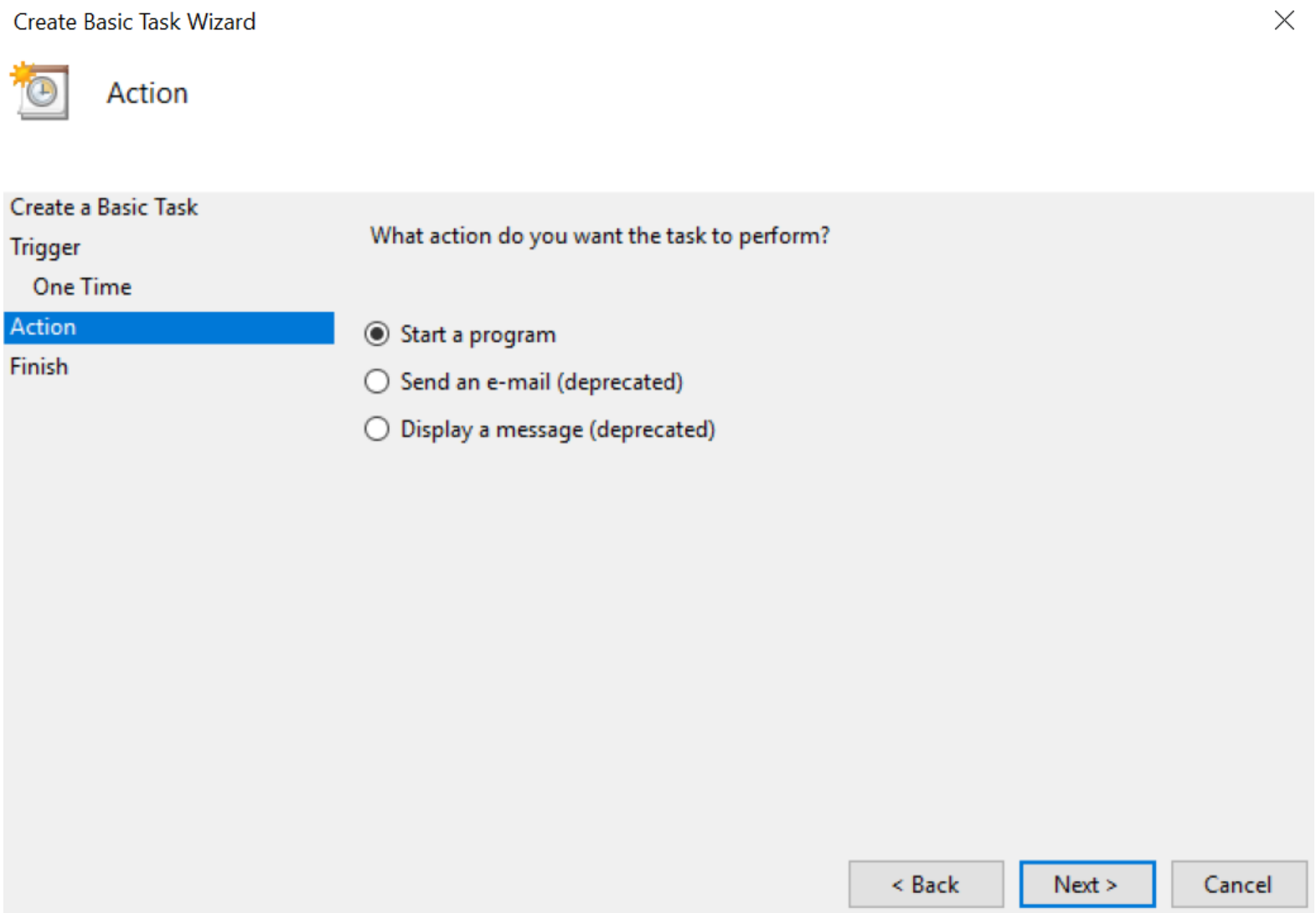
You will then be asked to name your task and describe it.



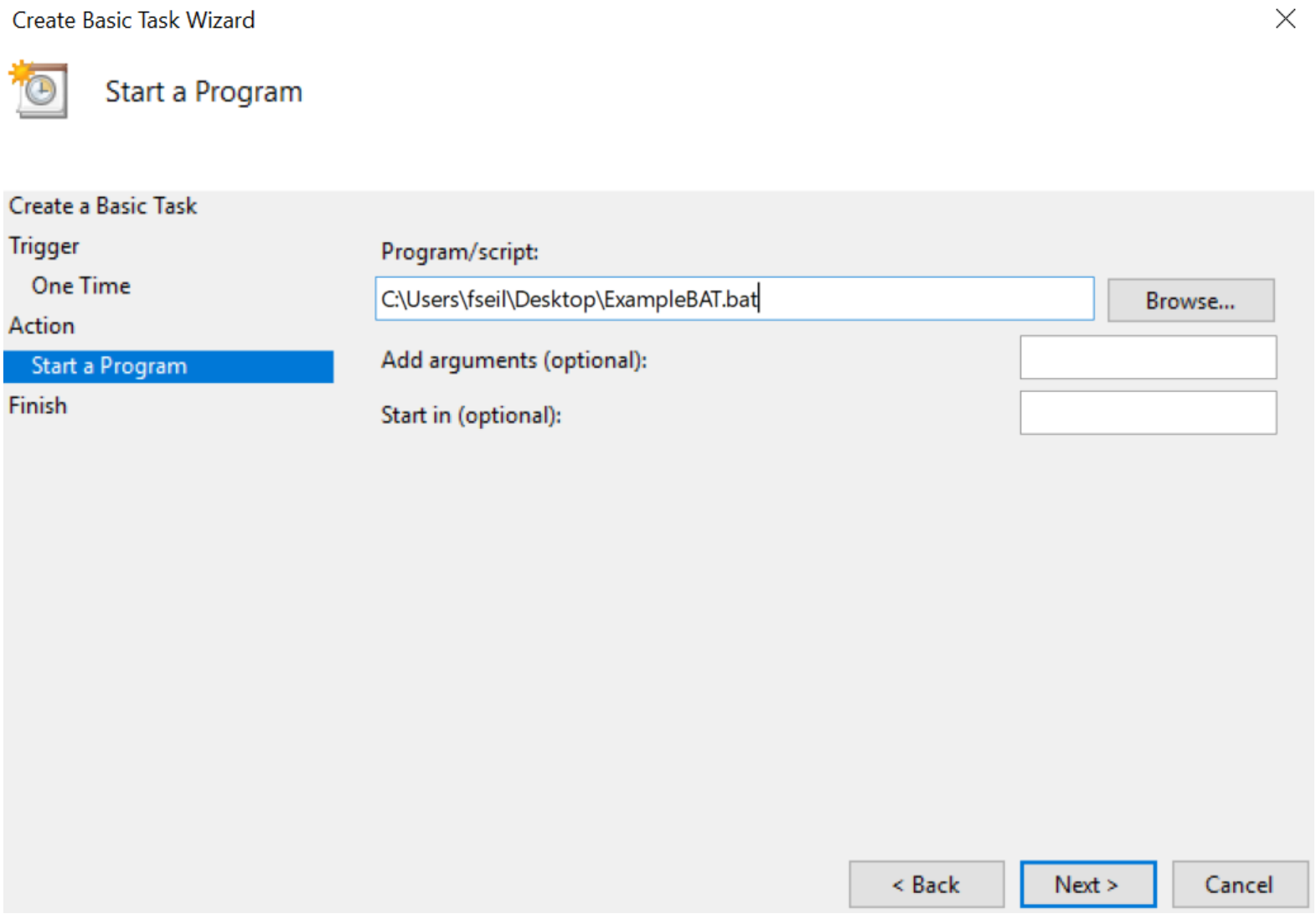
You can then select the frequency at which the task will run.



You will select the nature of the task. In our case, it's starting a program.



You can then specify the path to your bat file.



After the finish, your task is created. It will run at the specified frequency whenever your computer is turned on. Tasks can be deactivated and managed in the main view of the Task Manager app.