6. Market Orders

This section covers **market orders**, the most simple yet equally important order type.

In [1]:

```
import fxcmpy
import pandas as pd
import datetime as dt
con = fxcmpy.fxcmpy(config_file='fxcm.cfg')
```

6.1. Placing Orders

Before one gets started, a check of the **open positions** might be helpful.

In [2]:

```
con.get_open_positions().T
```

Out[2]:

Currently, there are no open positions. **To place orders**, one can use the methods `con.create_market_sell_order()` or `con.create_market_buy_order()`, respectively.

In [3]:

```
order = con.create_market_sell_order('EUR/USD', 100)
```

In [4]:

```
order = con.create_market_buy_order('USD/JPY', 200)
```

The new **open positions table** might then look as follows. Two new positions have been created.

In [5]:

```
con.get_open_positions().T
```

Out[5]:

	0	1
accountId	2815291	2815291
accountName	02815291	02815291
amountK	100	200
close	1.17067	112.38
com	0	0
currency	EUR/USD	USD/JPY
currencyPoint	10	17.7951
grossPL	-27	-56.9496
isBuy	False	True
isDisabled	False	False
limit	0	0
open	1.1704	112.412
ratePrecision	5	3
roll	0	0
stop	0	0
stopMove	0	0
t	1	1
time	07162018131217	07162018131224
tradeId	187549032	187549066
usedMargin	1300	2000
valueDate		
visiblePL	-2.7	-3.2

6.2. General Market Orders

`fxcmpy` allows to place more complex orders, this can be done with the method `con.open_trade()`. A detailed description of the meaning of the parameters is found in the FXCM API Documentation.

In [6]:

```
order = con.open_trade(symbol='USD/JPY', is_buy=True,
                       rate=105, is_in_pips=False,
                       amount='1000', time_in_force='GTC',
                       order_type='AtMarket', limit=120)
```

In [7]:

```
con.get_open_positions().T
```

Out[7]:

	0	1	2
accountId	2815291	2815291	2815291
accountName	02815291	02815291	02815291
amountK	100	200	1000
close	1.17068	112.385	112.385
com	0	0	0
currency	EUR/USD	USD/JPY	USD/JPY
currencyPoint	10	17.7943	88.9715
grossPL	-28	-48.0491	-177.96
isBuy	False	True	True
isDisabled	False	False	False
limit	0	0	120
open	1.1704	112.412	112.405
ratePrecision	5	3	3
roll	0	0	0
stop	0	0	0
stopMove	0	0	0
t	1	1	1
time	07162018131217	07162018131224	07162018131304
tradeId	187549032	187549066	187549170
usedMargin	1300	2000	10000
valueDate			
visiblePL	-2.8	-2.7	-2

6.3. Trade Ids

Every position has the attribute `tradeId`, here it is the column named `tradeId` of the `DataFrame` object returned by `con.get_open_positions()`.

In [8]:

```
con.get_open_positions()['tradeId']
```

Out[8]:

```
0    187549032
1    187549066
2    187549170
Name: tradeId, dtype: object
```

`fxcmpy` provides methods to get the trade `tradeIds` directly.

In [9]:

```
con.get_open_trade_ids()
```

Out[9]:
```
[187549032, 187549066, 187549170]
```

In [10]:

```
con.get_closed_trade_ids()
```

Out[10]:

```
[187518554, 187518509]
```

In [11]:

```
con.get_all_trade_ids()
```

Out[11]:

```
[187518509, 187518554, 187549032, 187549066, 187549170]
```

6.4. Editing Market Orders

To modify a trade, you need the `tradeId` value.

In [12]:

```
tradeId = con.get_open_trade_ids()[-1]
```

To **change the trade's stop or limit rate**, one can use the `change_trade_stop_limit()` method.

In [13]:

```
con.change_trade_stop_limit(tradeId, is_in_pips=False,
                            is_stop=False, rate=115)
```

As seen below, the limit rate of the last trade has changed to 115.

In [14]:

```
con.get_open_positions().T
```

Out[14]:

	0	1	2
accountId	2815291	2815291	2815291
accountName	02815291	02815291	02815291
amountK	100	200	1000
close	1.17068	112.391	112.391
com	0	0	0
currency	EUR/USD	USD/JPY	USD/JPY
currencyPoint	10	17.7934	88.9668
grossPL	-28	-37.3695	-124.565
isBuy	False	True	True
isDisabled	False	False	False
limit	0	0	115
open	1.1704	112.412	112.405
ratePrecision	5	3	3
roll	0	0	0
stop	0	0	0
stopMove	0	0	0
t	1	1	1
time	07162018131217	07162018131224	07162018131304
tradeId	187549032	187549066	187549170
usedMargin	1300	2000	10000
valueDate			
visiblePL	-2.8	-2.1	-1.4

6.5. Closing Positions

Positions can be closed individually via the `tradeId` ...

In [15]:

```
con.close_trade(trade_id=tradeId, amount=1000)
```

In [16]:

```
con.get_open_positions().T
```

Out[16]:

	0	1
accountId	2815291	2815291
accountName	02815291	02815291
amountK	100	200
close	1.17068	112.391
com	0	0
currency	EUR/USD	USD/JPY
currencyPoint	10	17.7934
grossPL	-28	-37.3695
isBuy	False	True
isDisabled	False	False
limit	0	0
open	1.1704	112.412
ratePrecision	5	3
roll	0	0
stop	0	0
stopMove	0	0
t	1	1
time	07162018131217	07162018131224
tradeId	187549032	187549066
usedMargin	1300	2000
valueDate		
visiblePL	-2.8	-2.1

... or for all positions in an instrument ...

In [17]:

```
con.close_all_for_symbol('EUR/USD')
```

In [18]:

```
con.get_open_positions().T
```

Out[18]:

	0
accountId	2815291
accountName	02815291
amountK	200
close	112.4
com	0
currency	USD/JPY
currencyPoint	17.7919
grossPL	-21.3523
isBuy	True
isDisabled	False
limit	0
open	112.412
ratePrecision	3
roll	0
stop	0
stopMove	0
t	1

	0
time	07162018131224
tradeId	187549066
usedMargin	2000
valueDate	
visiblePL	-1.2

... or simply for all positions over all instruments.

In [19]:

```
con.close_all()
```

After this method call, there are naturally no open positions left.

In [20]:

```
con.get_open_positions()
```

Out[20]:

The **closed positions** are found in the table returned by `con.get_closed_postions()`.

In [21]:

```
con.get_closed_positions().T
```

Out[21]:

	0	1	2	3	4
accountName	02815291	02815291	02815291	02815291	02815291
amountK	200	100	1000	100	200
close	112.391	1.17254	112.39	1.17062	112.398
closeTime	07162018131121	07162018112750	07162018131337	07162018131345	07162018131354
com	0	0	0	0	0
currency	USD/JPY	EUR/USD	USD/JPY	EUR/USD	USD/JPY
currencyPoint	17.7922	10	88.9608	10	17.7922
grossPL	97.87	-26	-133.46	-22	-24.91
isBuy	True	False	True	False	True
open	112.336	1.17228	112.405	1.1704	112.412
openTime	07162018112746	07162018112734	07162018131304	07162018131217	07162018131224
ratePrecision	3	5	3	5	3
roll	0	0	0	0	0
t	2	2	2	2	2
tradeId	187518554	187518509	187549170	187549032	187549066
valueDate					
visiblePL	5.5	-2.6	-1.5	-2.2	-1.4

6.6. The `fxcmpy_open_position` Class

For convenience, `fxcmpy` provides a class called `fxcmpy_open_positions`. All open positions are stored in the attribute `fxcmpy.open_pos`, a dictionary with the position's `tradeId` as key and the `fxcmpy_open_position` instance as value. Since there are no open positions left, the `dict` object is empty.

In [22]:

```
con.open_pos
```

Out[22]:

```
{}
```

Therefore, create an order ...

In [23]:

```
order = con.create_market_sell_order('EUR/USD', 100)
```

... to see the `fxcmpy_open_position` objects in the `dict` object.

In [24]:

```
con.open_pos
```

Out[24]:

```
{187549393: <fxcmpy.fxcmpy_open_position.fxcmpy_open_position at 0x7f2ff71c45f8>}
```

To get a single `fxcmpy_open_position` object, one can use `con.get_open_position()`.

In [25]:

```
tradeId = con.get_open_trade_ids()[0]
```

In [26]:

```
pos = con.get_open_position(tradeId)
```

In [27]:

```
pos
```

Out[27]:

```
<fxcmpy.fxcmpy_open_position.fxcmpy_open_position at 0x7f2ff71c45f8>
```

In [28]:

```
print(pos)
```

```
accountId:        2815291
accountName:      02815291
amountK:          100
close:            1.17049
com:              0
currency:         EUR/USD
currencyPoint:    10
grossPL:          -24
isBuy:            False
isDisabled:       False
limit:            0
open:             1.17025
roll:             0
stop:             0
stopMove:         0
time:             2018-07-16 13:14:07
tradeId:          187549393
usedMargin:       1300
valueDate:
visiblePL:        -2.4
```

The `fxcmpy_open_position` object has `get` methods for all of its attributes, for example:

In [29]:

```
pos.get_amount()
```

Out[29]:

```
100
```

In [30]:

```
pos.get_currency()
```

Out[30]:

```
'EUR/USD'
```

Close the position with the `pos.close()` method.

In [31]:

```
pos.close()
```

In [32]:

```
con.get_open_positions()
```

Out[32]:

In [33]:

```
con.close()
```