Pythonで通信対応電源と通信(MODBUS)する方法について（Windows編）

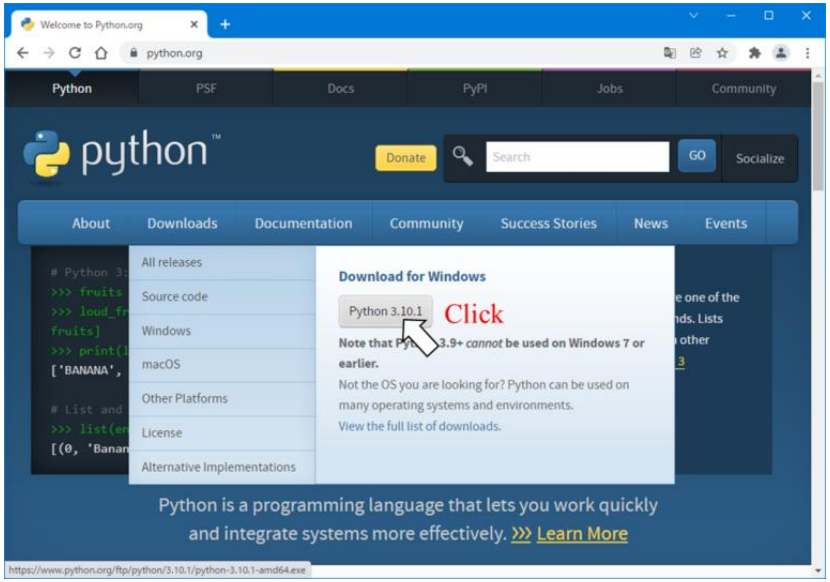
2024年4月5日　Ver 1.0

目次

1．Pythonのインストール

下記のサイトからPythonをダウンロードして、PCにインストールします。
https://www.python.org/

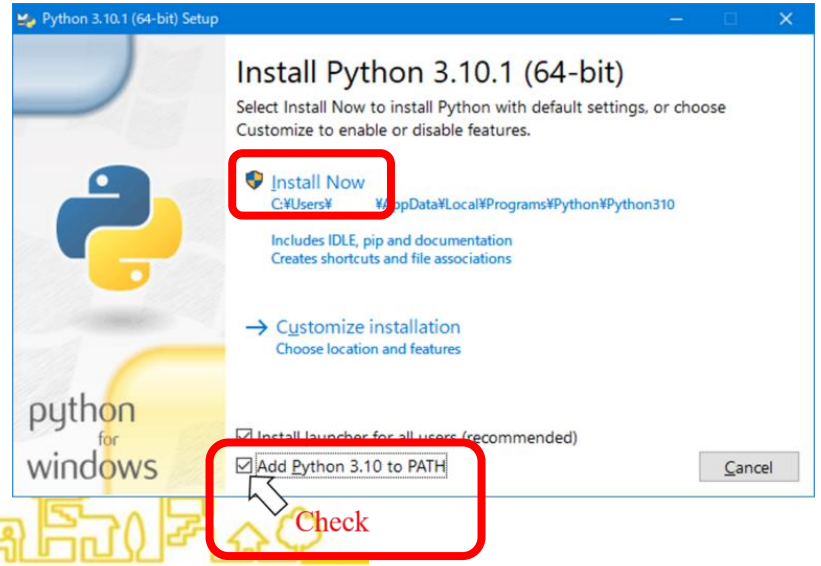
Downloads -> Python 3.xx.x からダウンロードできます。



ダウンロードしたインストーラを実行します。

インストール時
「Add Python 3.xx to PATH」
にチェックを付けます。

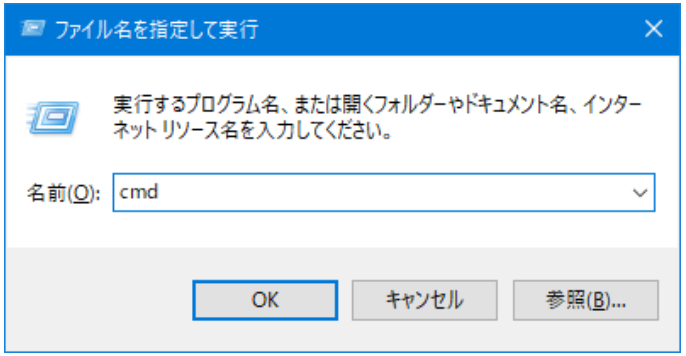
「Install Now」をクリックすると、インストールが始まります。



1．Pythonのインストール

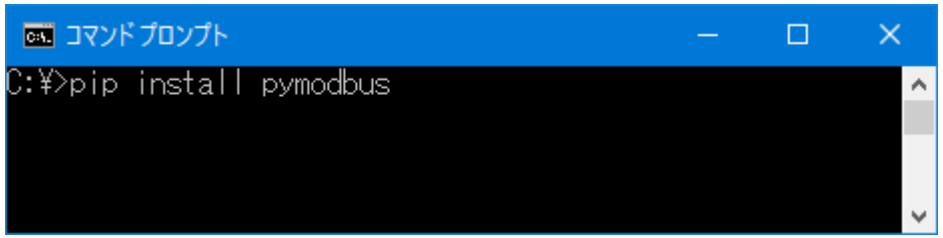
pymodbusのインストール

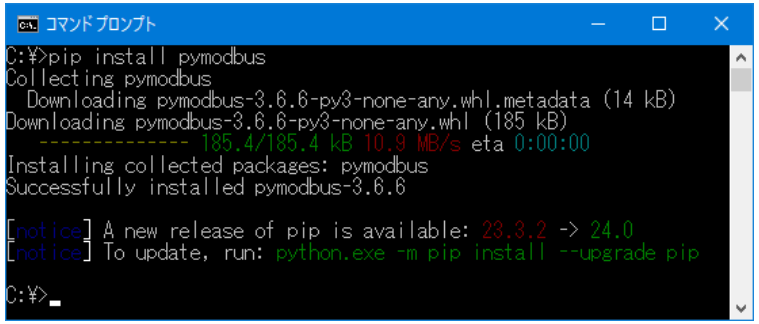
PythonでMODUS通信を行えるようにするためのライブラリをインストールします。
最初にコマンドプロンプトを立ち上げます。
「Win ＋ R」を押して、「cmd」と入力して、「OK」をクリックすると起動します。



起動したら、下記のコマンドを入力してEnterを押します。

```
pip install pymodbus
```

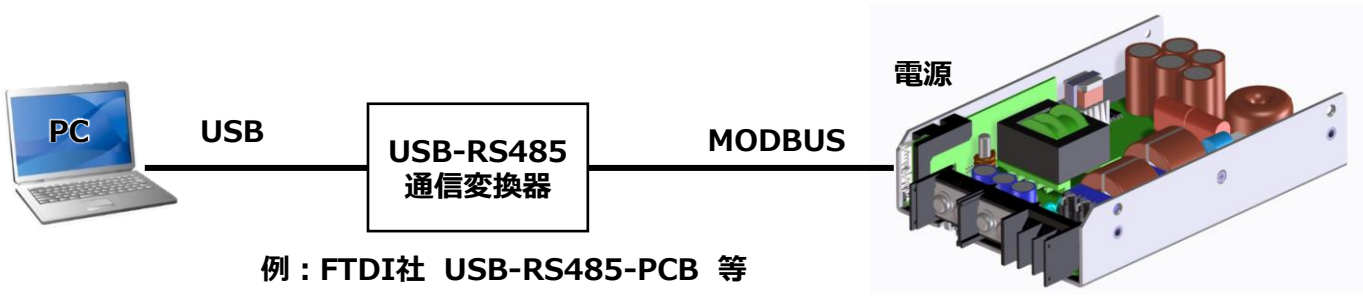




インストールが完了しましたら、コマンドプロンプトを閉じます

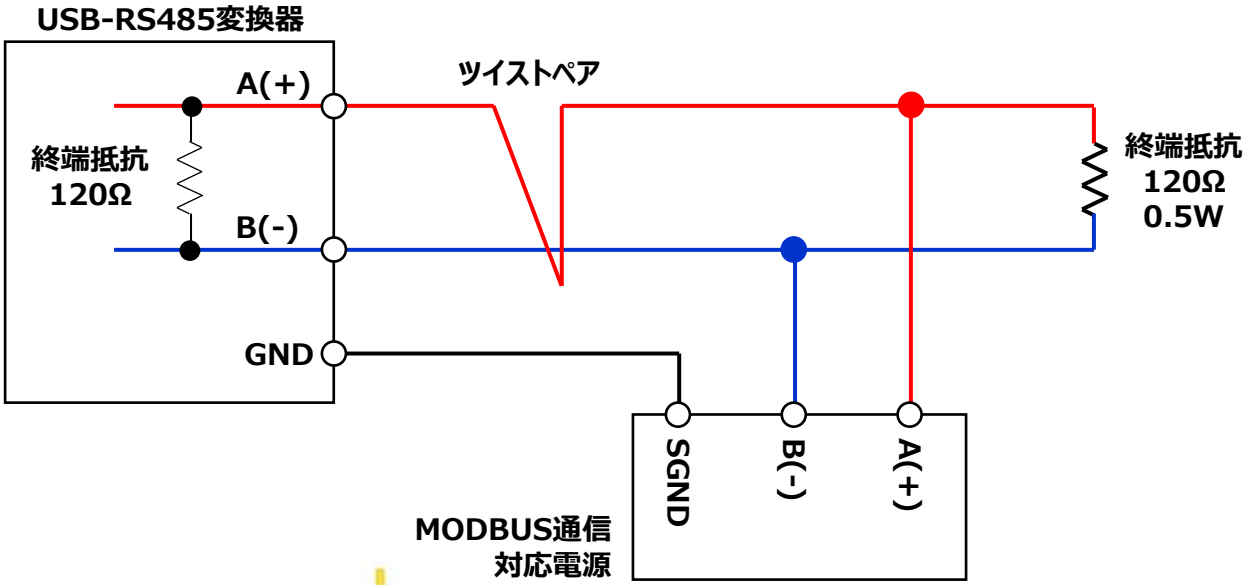
2.パソコンと電源の接続

MODBUS通信対応電源をパソコンに接続します。接続には通信変換器が必要になります。



<接続図>　　例：FTDI社　USB-RS485-PCB　等

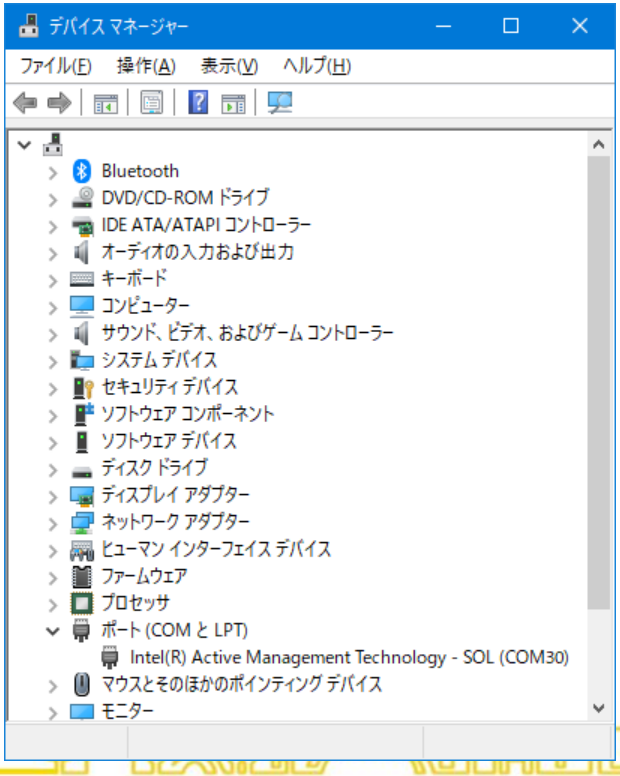
通信変換器と電源は以下のように接続します。

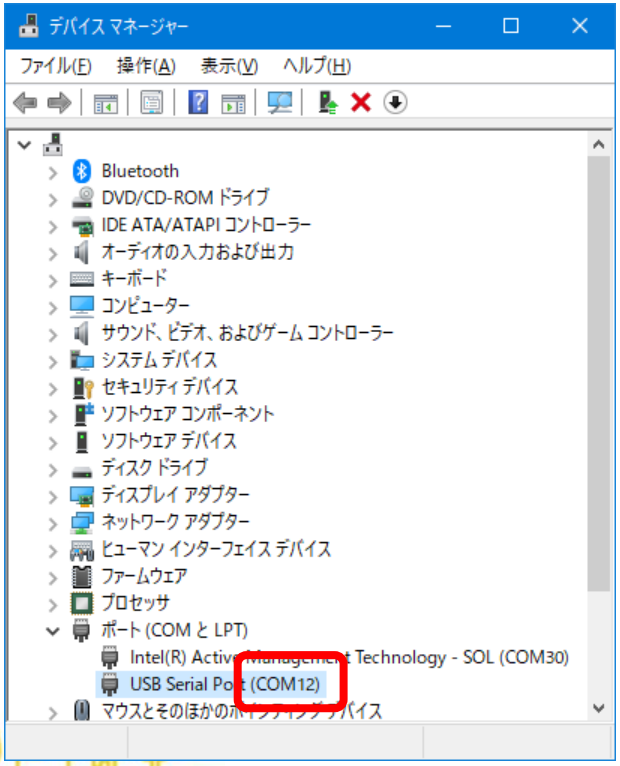


3.シリアル通信ポートの確認

シリアル通信を行うためには、電源と接続した通信変換器のシリアル通信ポートの名称が必要になります。

【調べ方】
・デバイスマネージャーを開きます。（Windowsアイコンを右クリックし、デバイスマネージャーをクリックします。）
・「ポート(COMとLPT)」を開きます。
・通信変換器をパソコンに接続します。
・新しく増えたポートの末尾の「COMxx」を記録します。(例では「COM12」)



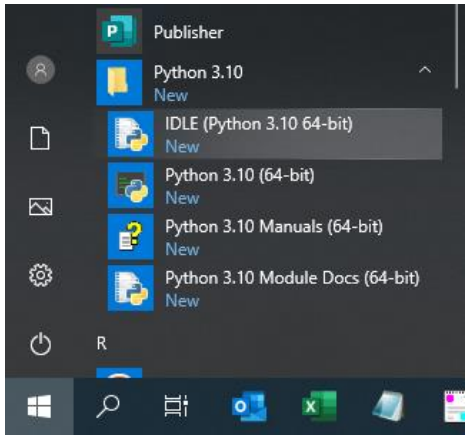


4.Pythonで通信を行う

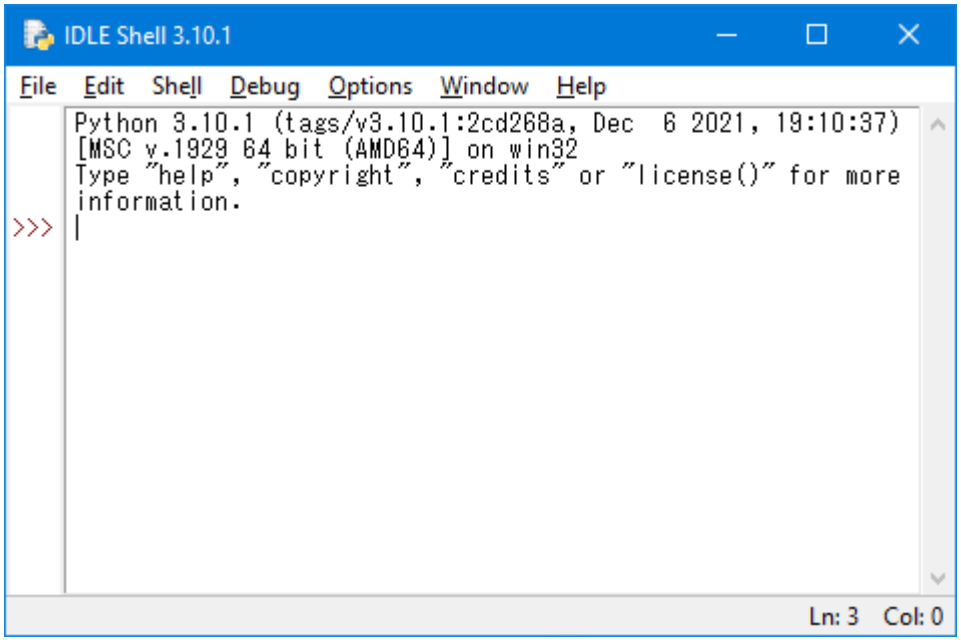
Pythonでプログラムを作成し、通信変換器を通して電源と通信を行っていきます。

まずは、Pythonのプログラムを作成実行するIDLEを起動します。

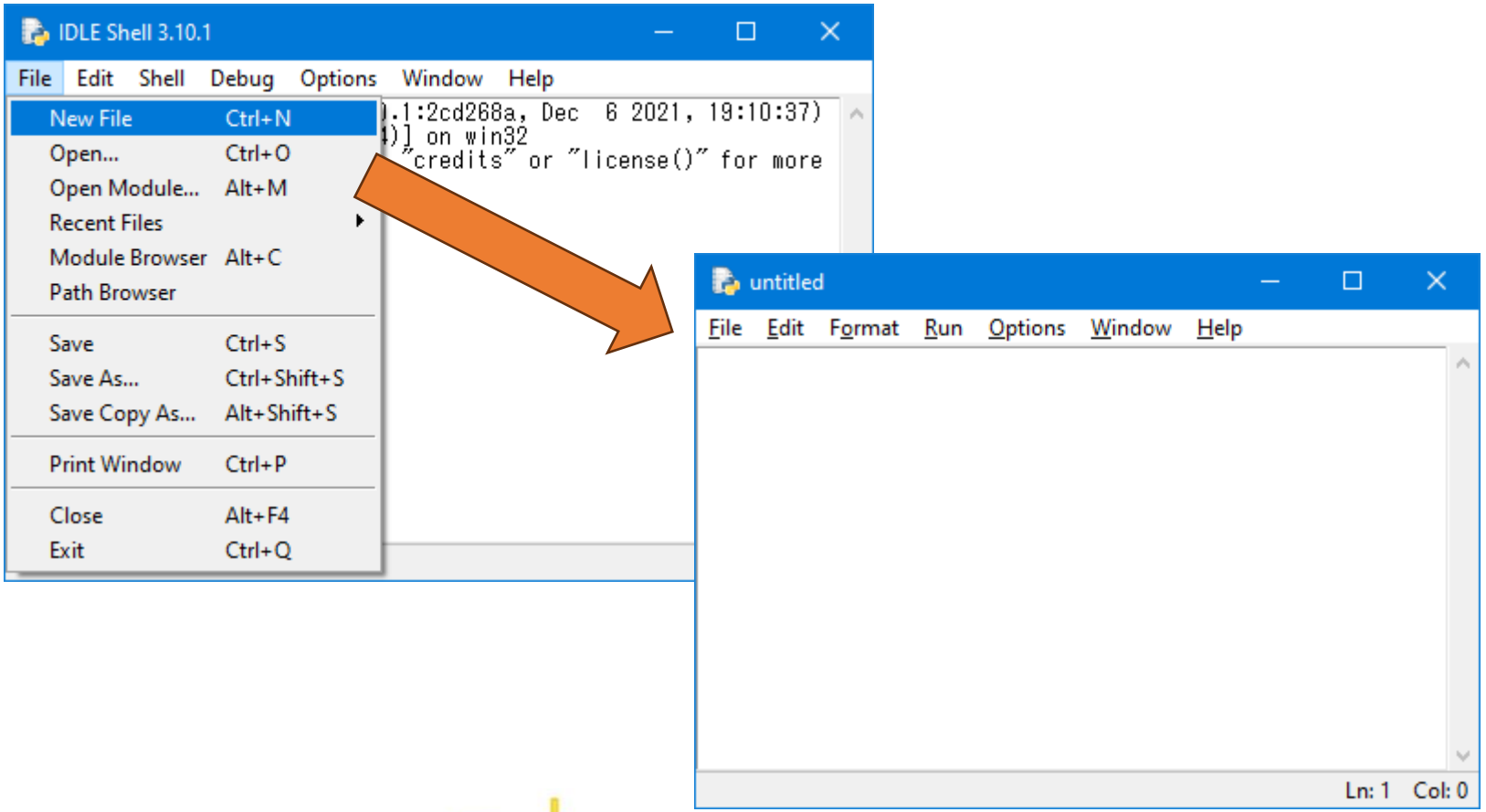
スタートメニュー -> Python 3.xx -> IDLE(Python 3. …)
で起動できます。



IDLEが起動すると下記の画面が表示されます。



File -> New File をクリックして、プログラムを記述するエディタを用意します。



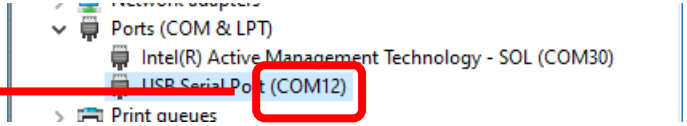
4.Pythonで通信を行う

(1)Inputレジスタの読み込み

MODBUS通信を行うプログラムを記述していきます。Inputレジスタ(開始アドレス:0000h)を読み出す例です。

```
from pymodbus.client import ModbusSerialClient

try :
    client = ModbusSerialClient(
        port     = "COM12",
        baudrate = 19200,
        parity   = 'E',
        method   = "rtu",
        timeout  = 0.1)

    res = client.connect()
    if res == False :
        raise Exception("通信変換器に接続できません")

    reg_addr = 0
    res = client.read_input_registers(
      address=reg_addr, count=1, slave=1)
    if res.isError() :
        raise Exception(res)

    print(f"{reg_addr:04X}h : {res.registers[0]}")
except Exception as e:
    raise e
finally :
    client.close()
```

左記のプログラムをエディタに記述します。

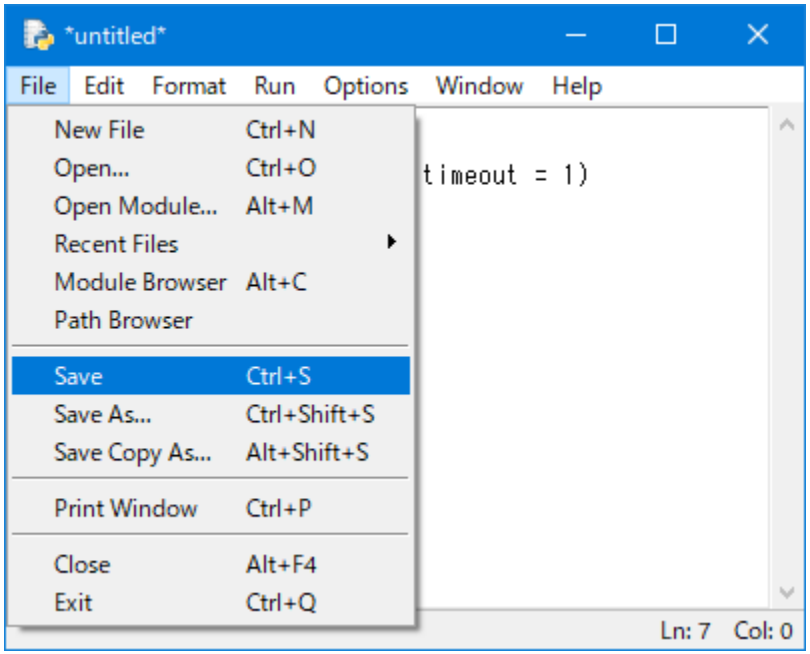


port = "COM12"の行は
先ほど調べた通信変換器のCOMポート番号に直してください。

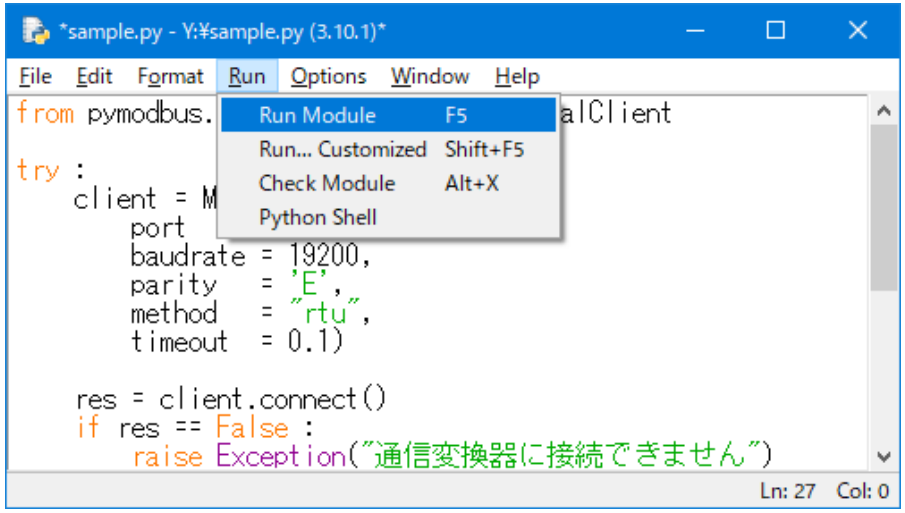
address ：読込みを開始するレジスタのアドレス
count ：読み込むレジスタ数
slave ：電源のアドレス

4.Pythonで通信を行う

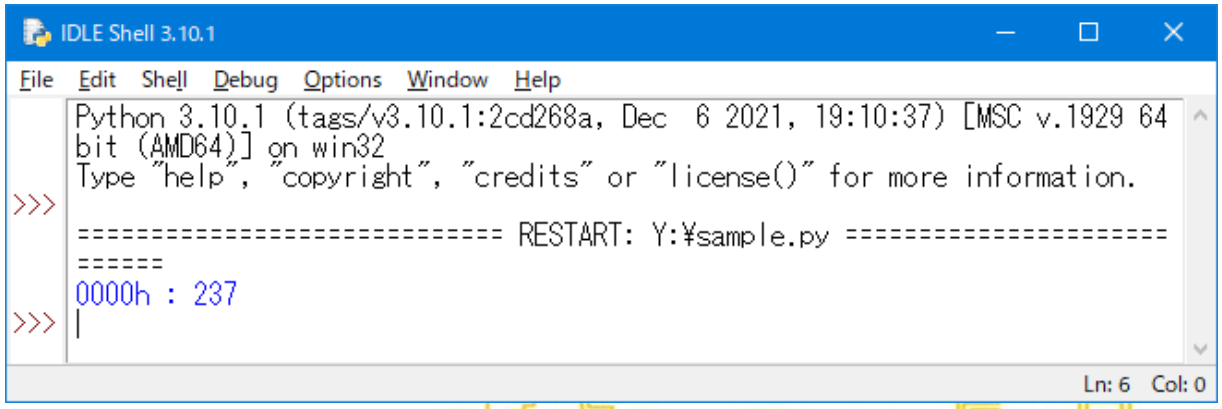
入力を行った後は、File -> Save で保存を行います。



Run -> Run Module をクリックし、プログラムを実行します。



実行すると別ウィンドウが開かれ、以下のようにInputレジスタのアドレスと値が表示されます。




4.Pythonで通信を行う

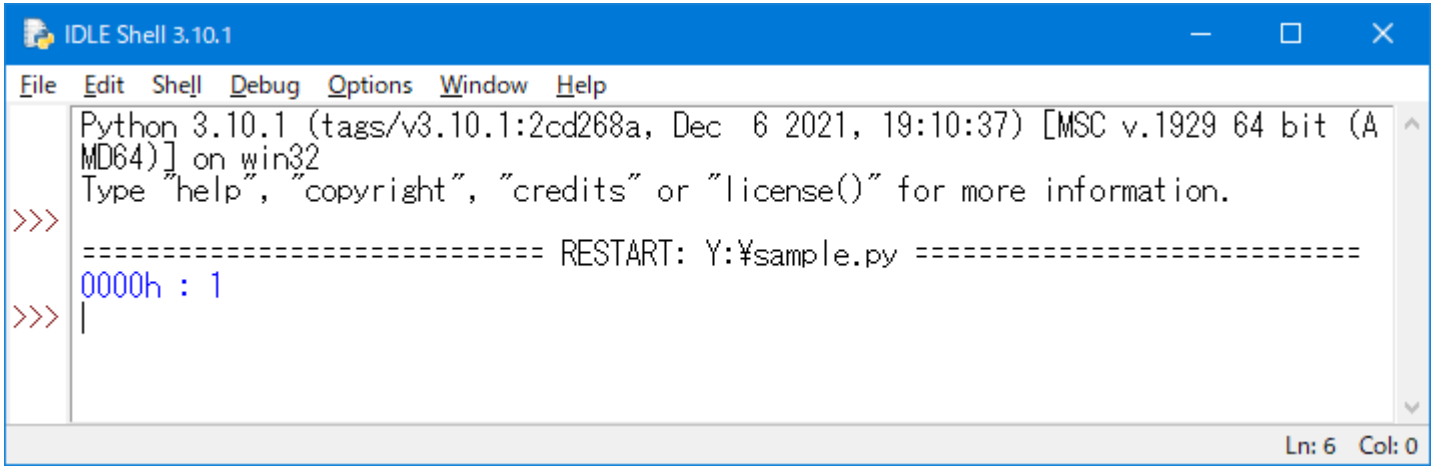
(2)Holdingレジスタの読み込み
Holdingレジスタ(開始アドレス:0000h)を読み出す例です。

```
from pymodbus.client import ModbusSerialClient

try :
    client = ModbusSerialClient(
        port     = "COM12",
        baudrate = 19200,
        parity   = 'E',
        method   = "rtu",
        timeout  = 0.1)

    res = client.connect()
    if res == False :
        raise Exception("通信変換器に接続できません")

    reg_addr = 0
    res = client.read_holding_registers(
      address=reg_addr, count=1, slave=1)
    if res.isError() :
        raise Exception(res)

    print(f"{reg_addr:04X}h : {res.registers[0]}")
except Exception as e:
    raise e
finally :
    client.close()
```

address ：読込みを開始するレジスタのアドレス
count ：読み込むレジスタ数
slave ：電源のアドレス

4.Pythonで通信を行う

実行すると、以下のようにHoldingレジスタのアドレスと値が表示されます。



4.Pythonで通信を行う

(3)Holdingレジスタの書き込み
Holdingレジスタ (開始アドレス:0000h)に0を書き込む例です。

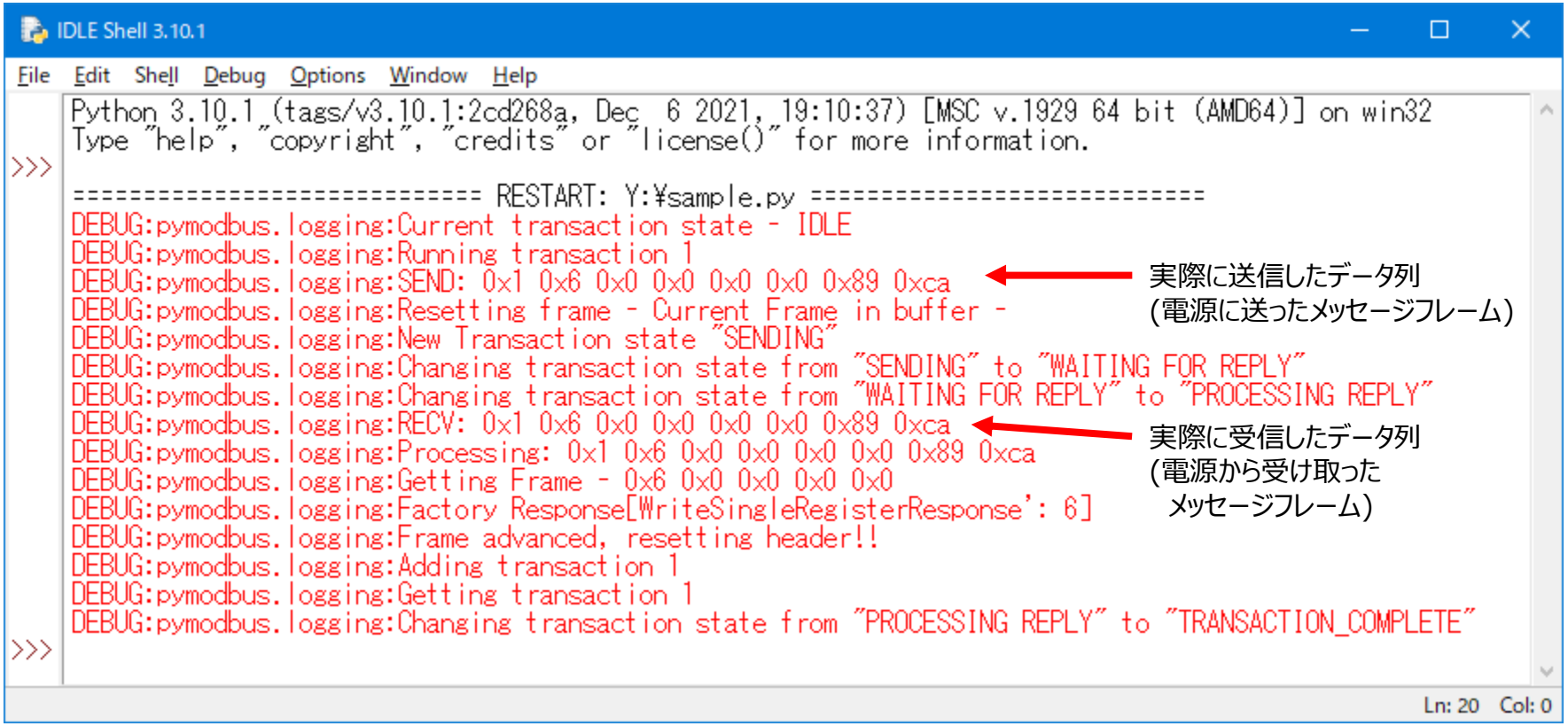
```
from pymodbus.client import ModbusSerialClient

import logging
logging.basicConfig()
log = logging.getLogger()
log.setLevel(logging.DEBUG)

try :
    client = ModbusSerialClient(
        port     = "COM12",
        baudrate = 19200,
        parity   = 'E',
        method   = "rtu",
        timeout  = 0.1)

    res = client.connect()
    if res == False :
        raise Exception("通信変換器に接続できません")

    res = client.write_register(
      address=0, value=0, slave=1)
    if res.isError() :
        raise Exception(res)

except Exception as e:
    raise e
finally :
    client.close()
```

書き込み時の通信の経過を確認するために
ログを表示する設定を追加
（経過の表示が不要の場合は、この4行は不要です。）

address　：書込むレジスタのアドレス
value　　：書込む値
slave　　：電源のアドレス

4.Pythonで通信を行う

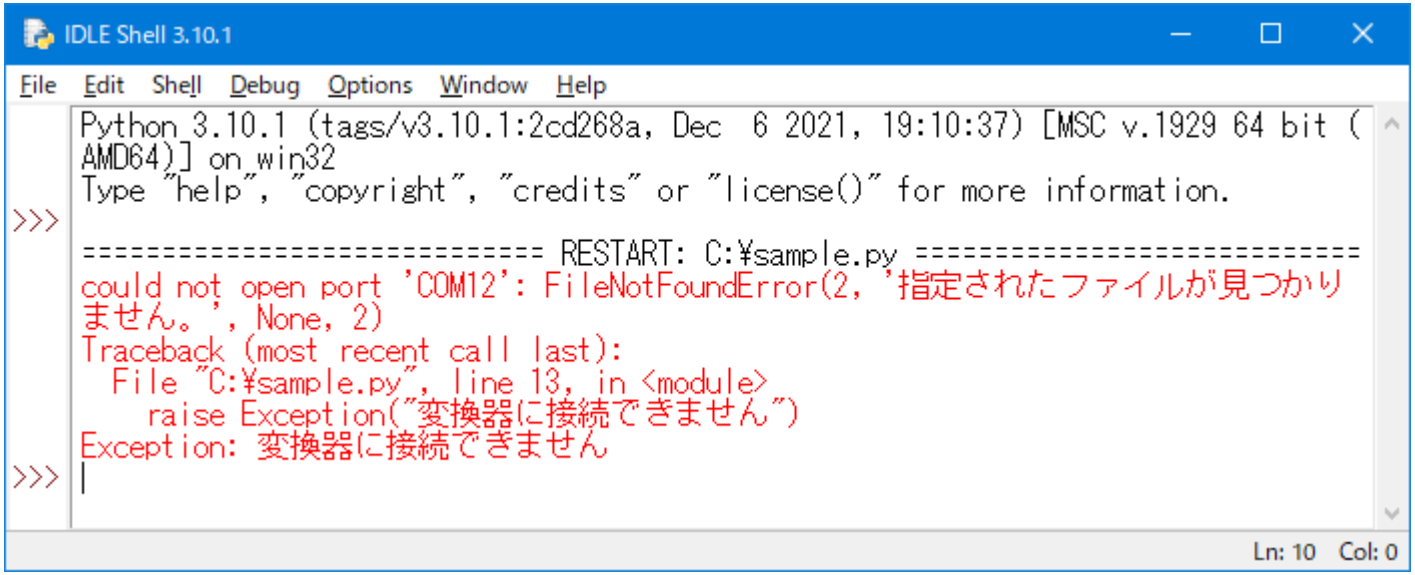
実行すると、以下のように通信の経過が表示され、Holdingレジスタに書き込みを行います。



4.Pythonで通信を行う

トラブルシューティング

＜症状＞

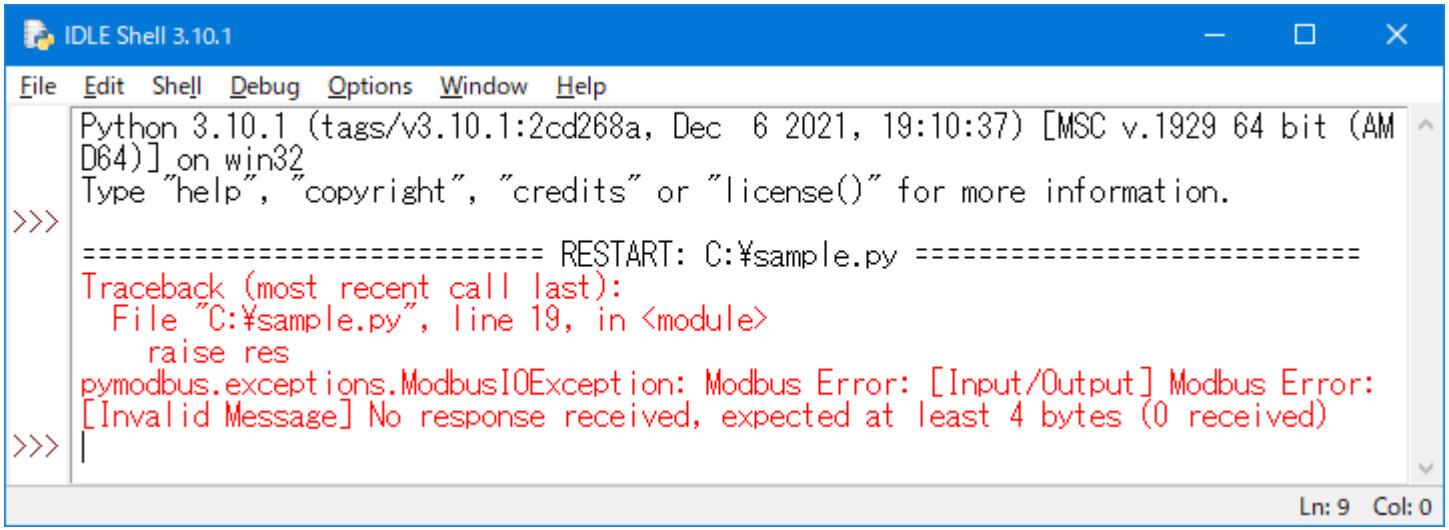


＜原因＞
通信変換器がPCに接続されていないか、COMポート番号が間違っています。

＜対処＞
PCに通信変換器が接続されているか確認し、デバイスマネージャーでCOMポート番号を確認して下さい。

トラブルシューティング

＜症状＞

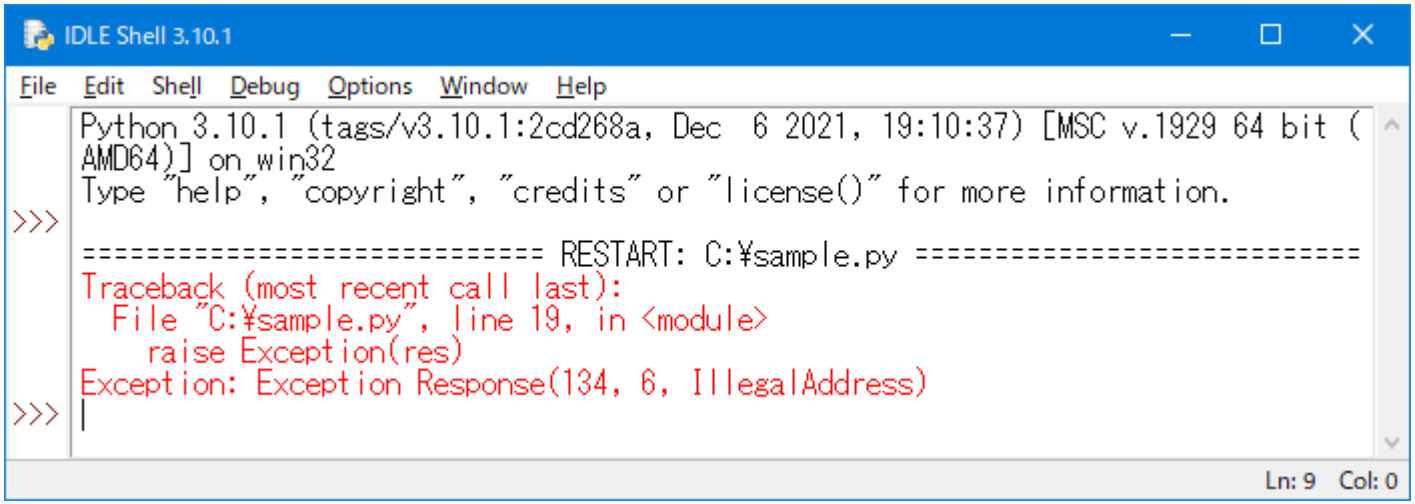


＜原因＞
電源から通信の返信を受信できず、タイムアウトしたためです。

＜対処＞
通信変換器と電源の間の接続や断線がないか確認の上、電源に入力が供給されているか確認して下さい。
read_input_registers / read_holding_registers / write_register 関数の引数 slave に
正しい電源のアドレスが設定されているか確認して下さい。

トラブルシューティング

＜症状＞



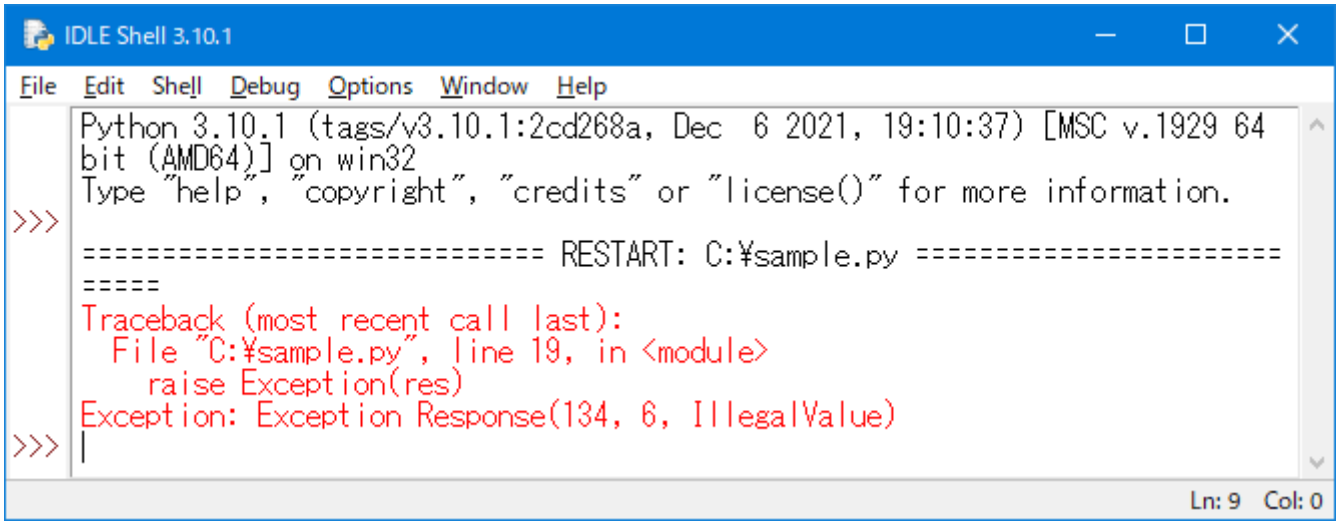
＜原因＞
製品で規定されていないレジスタのアドレスを指定したためです。

＜対処＞
read_input_registers / read_holding_registers / write_register 関数の引数 address に指定してるレジスタアドレスをご使用の電源の通信マニュアルでご確認下さい。

4.Pythonで通信を行う

トラブルシューティング

＜症状＞

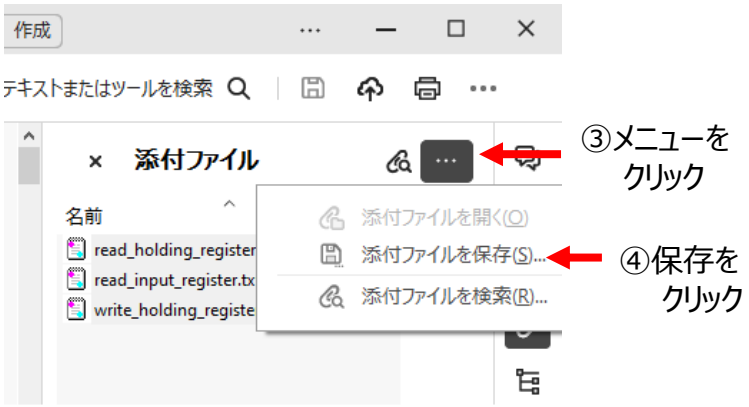


＜原因＞
書込む値が範囲外ためです。

＜対処＞
write_register 関数の引数 value に指定してる値が範囲内か、ご使用の電源の通信マニュアルでご確認下さい。

5.ソースコード

説明で使用したソースコードをテキストファイルとして、このPDFの添付ファイルに収録しています。
下記にAcrobat Readerでの添付ファイルの取り出し例を示します。



注意

本資料に記載されている内容は本資料発行時点のものであり、製品の仕様変更および改良などのために予告なく変更することがあります。最新版はコーセルのホームページをご確認ください。

本資料の内容につきましては、正確さを期するために万全の注意を払っておりますが、本資料中の誤記や情報の抜け、あるいは、情報の使用に起因する間接障害を含むいかなる損害に対しても、弊社は責任を負いかねますので、あらかじめご了承ください。

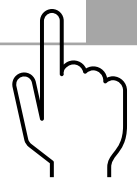
技術お問い合わせ専用ホットライン

■フリーダイヤル： 0120-52-8151

営業時間9：00～12：00／13：00～17：00（土曜・日曜・祝日・当社休日を除く）

お問い合わせは「コーセル　サポート」で検索

コーセル　サポート



COSEL

profile of COSEL CO.,LTD.

『 顧客起点のニーズを捉え、高付加価値製品とサービスの実現を図る 』