암호화폐 트레이딩 교육

Day 1 (1/2)

암호화폐 트레이딩 실습 환경 구축

Kangwuk Heo
calvin.heo@gmail.com

Contents

Overview

암호화폐 트레이딩 교육 과정에서, 실습을 통해서, 트레이딩이 무엇인지, 트레이딩을 어떤 방식으로 구성하고 진행하는지를 배울 수 있습니다.

암호화폐 트레이딩 환경 구축

Section 1 암호화폐 트레이딩 실습 환경을 위한 기본 베이스인 파이썬 언어를 사용하기 위한 필요 프로그램을 설치하는 과정을 순서대로 진행하는 과정을 설명합니다.

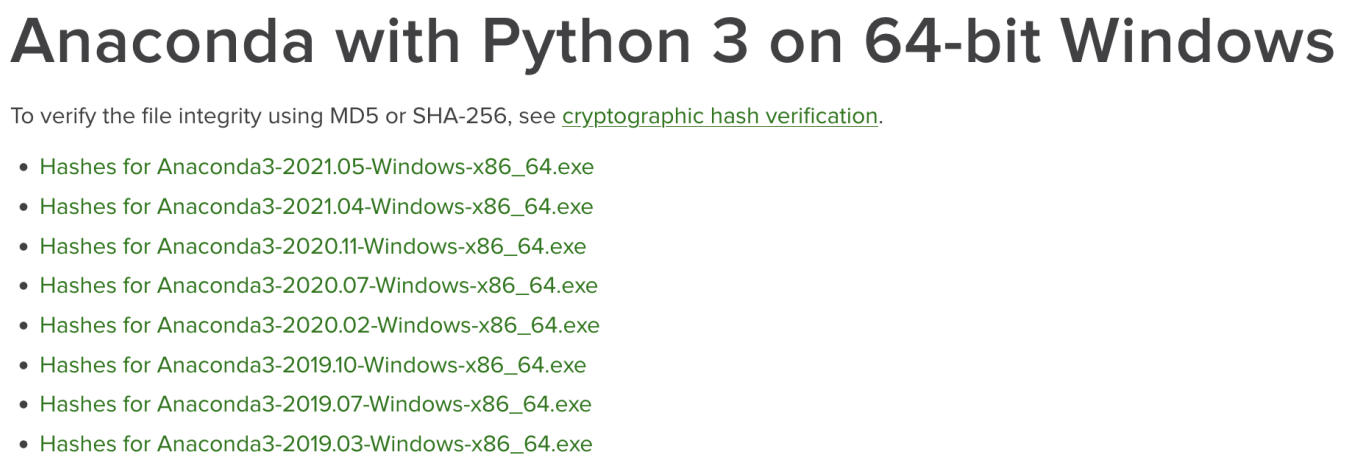
Section 1: 파이썬(Python) 설치 하기

파이썬은 Version 3 을 기반으로 설치를 진행합니다.

__1. 파이썬(Python)을 사용하기 위해서, Anaconda 툴을 다운로드 받습니다.

https://docs.anaconda.com/anaconda/install/hashes/win-3-64/

Anaconda with Python 3 on 64-bit Windows 를 선택합니다.



Hashes for Aanaconda3-2019-10-Windows-x86_64.exe 항목을 선택합니다.

Hashes for Anaconda3-2019.10-Windows-x86_64.exe

All installer files are available at https://repo.anaconda.com/archive/ .

You can verify the data integrity of the Anaconda installer files by running a local program to generate their MD5 or SHA-256 cryptographic hashes and checking the output to be sure it matches the hashes (or "checksums") below.

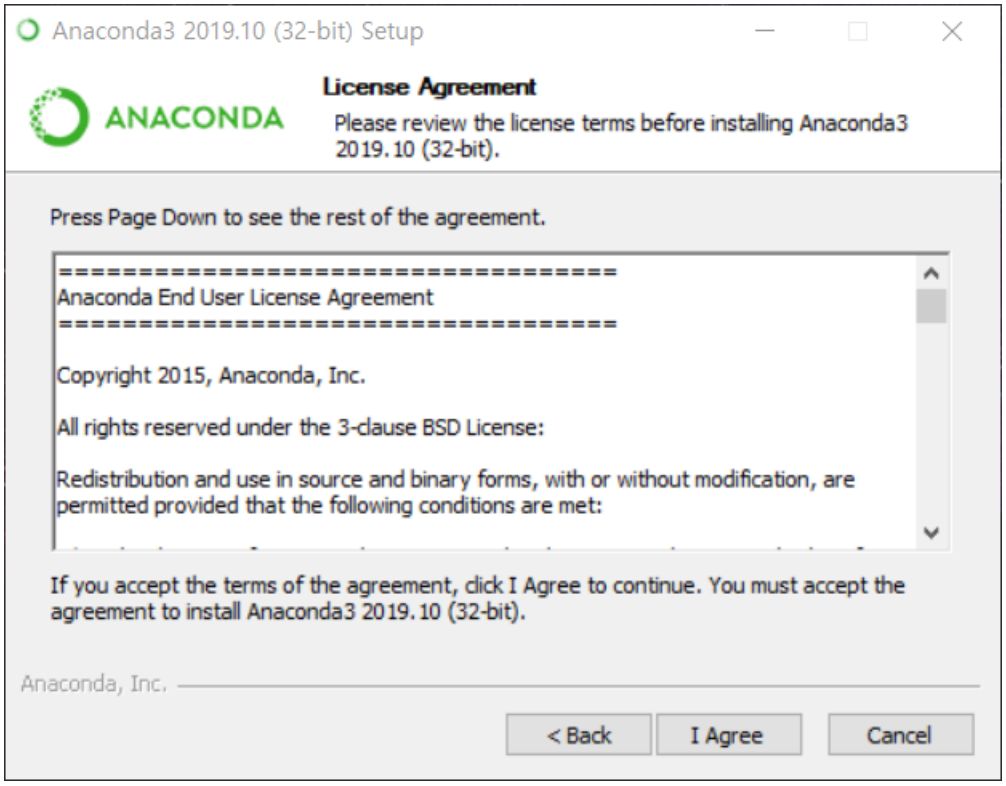
If the MD5 or SHA-256 hash that you generate does not match the one here, the file may not have downloaded completely. Please download it again and re-check. If repeated downloads produce the same result, please contact us to report the problem, including the file name, whether you used MD5 or SHA-256, the hash you generated, and the hash on the site.

exact time file was last modified, as Unix time stamp	1571149637.6830847
time file was last modified, in human readable format	2019-10-15 09:27:17
exact file size, in bytes	483964816
file size, in human friendly format	461.5 MiB
md5	`fafcdbf5feb6dc3081bf07cbb8af1dbe`
sha256	`9e632439ed40620b8518f11469ded7316eccb489d0dfc41770f72ca2b2202dd9`

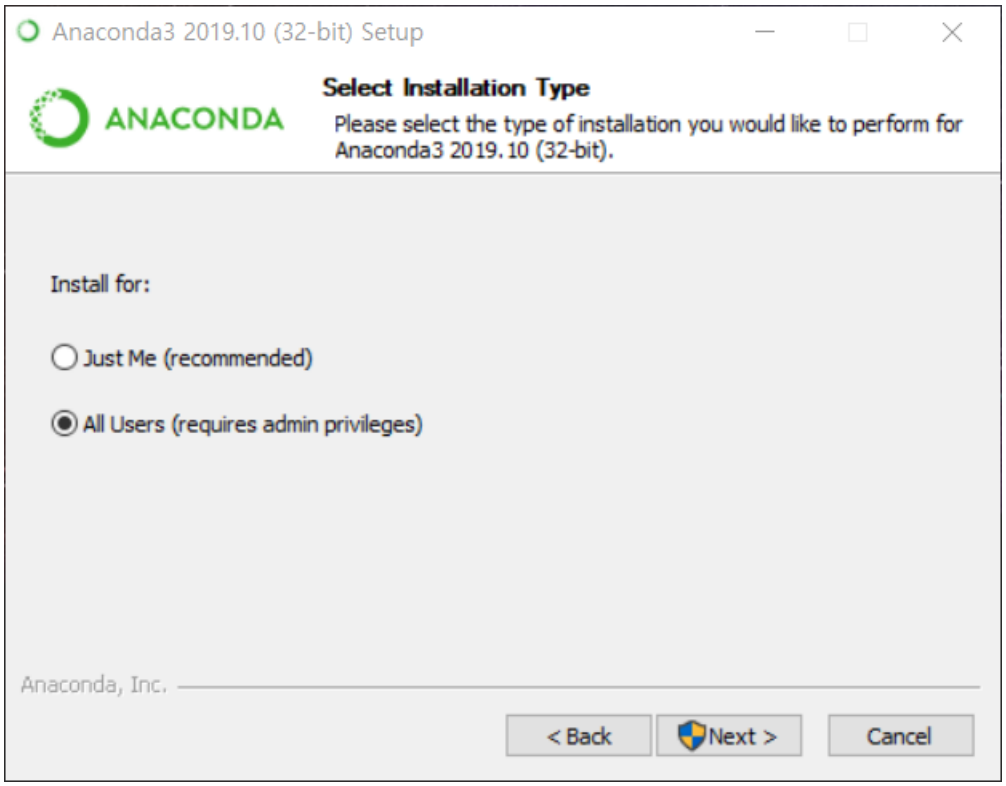
v: latest

__2. Ananconda3 windows 버전을 다운로드 받은 후, 설치파일을 실행합니다. 설치 윈도우창에서 Next 버튼을 클릭합니다.

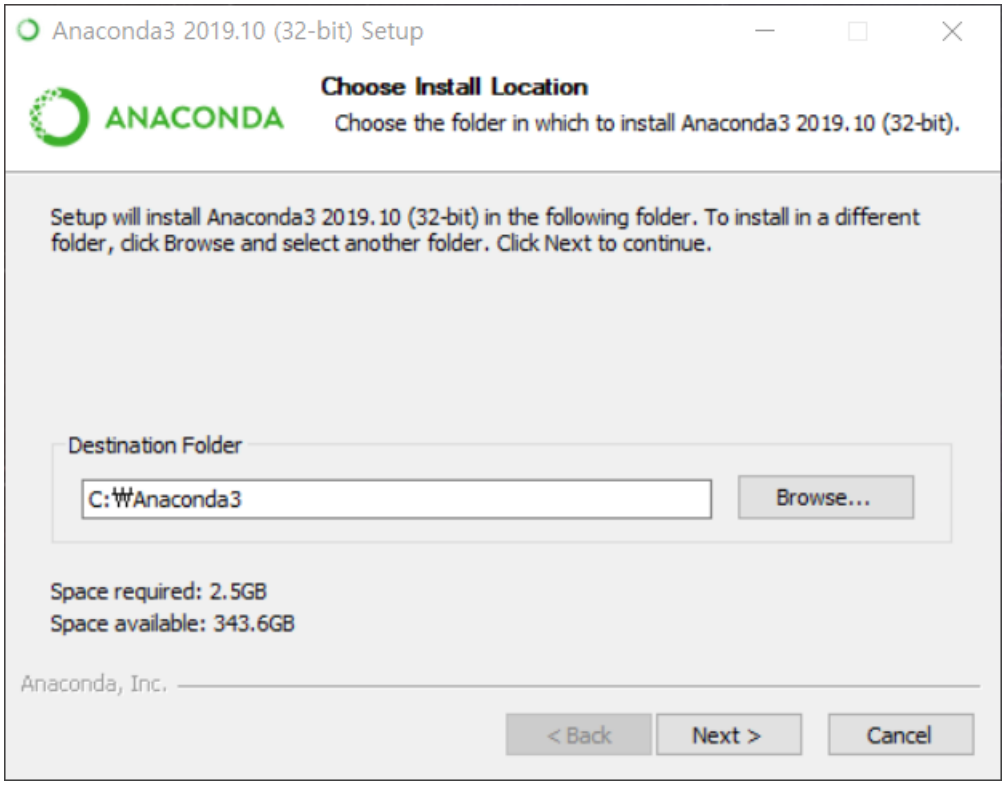
__3. 라이선스 동의 화면에서, I Agree 버튼을 클릭합니다.



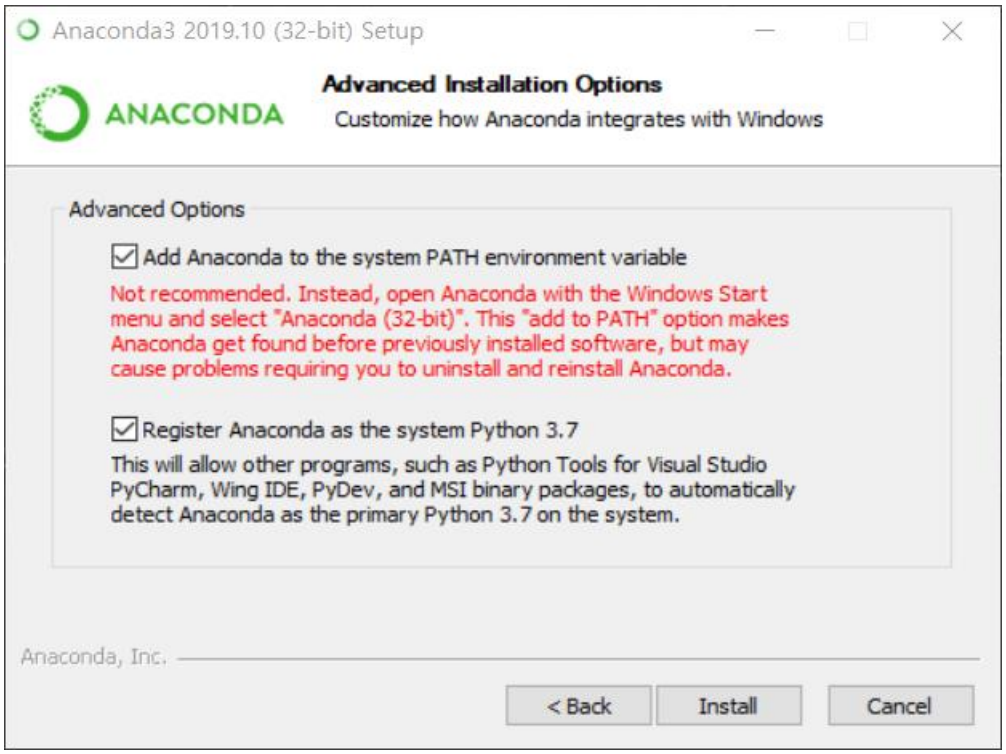
__4. Installation Type 에서 All Users 를 선택 후, Next 버튼을 클릭합니다.



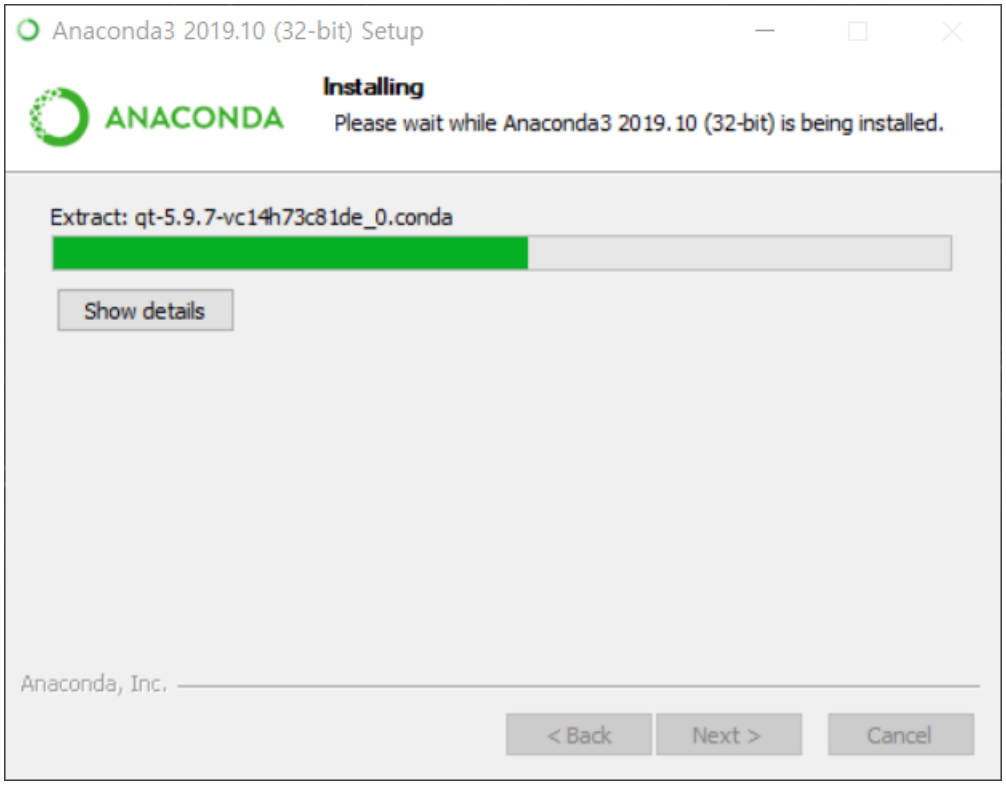
__5. Anaconda 설치 폴더를 지정 후, Next 버튼을 클릭합니다.



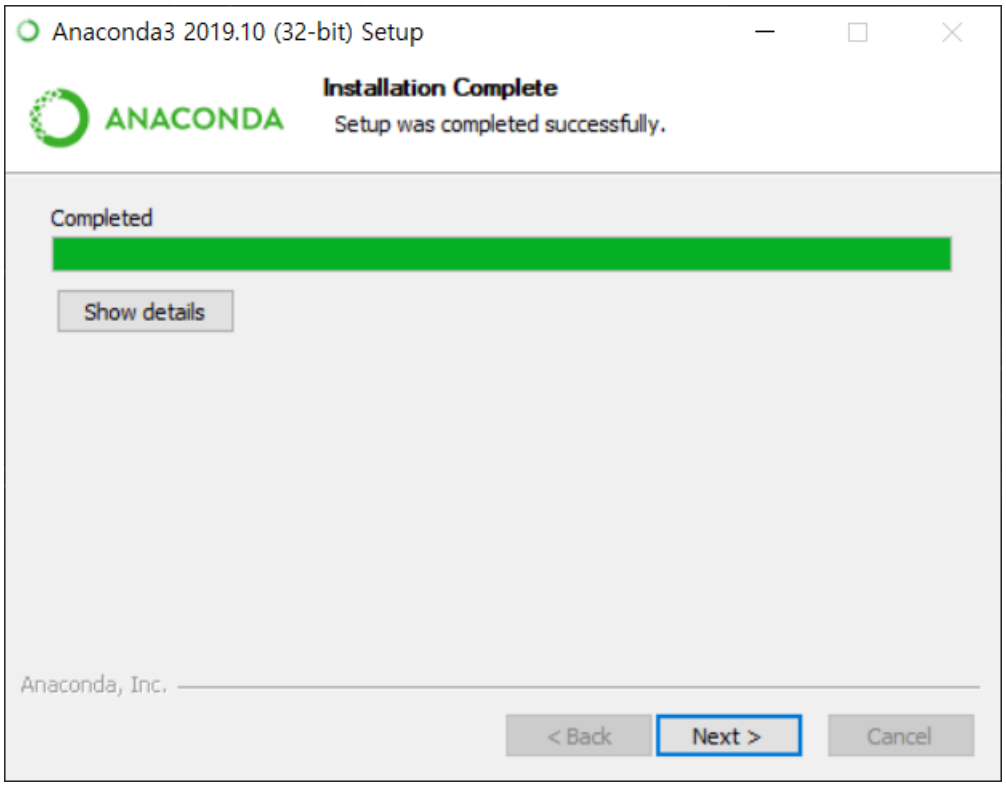
__6. 설치옵션에서 두개 모두 체크 후, Install 버튼을 클릭합니다.



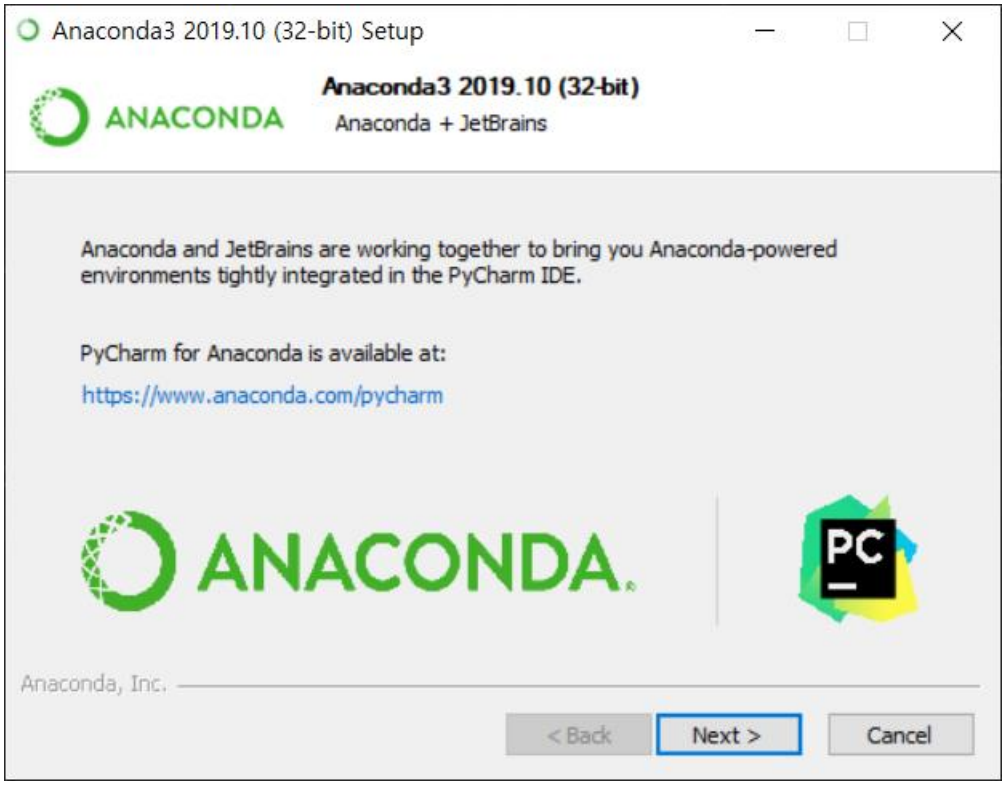
__7. Anaconda 설치 작업이 진행됩니다.



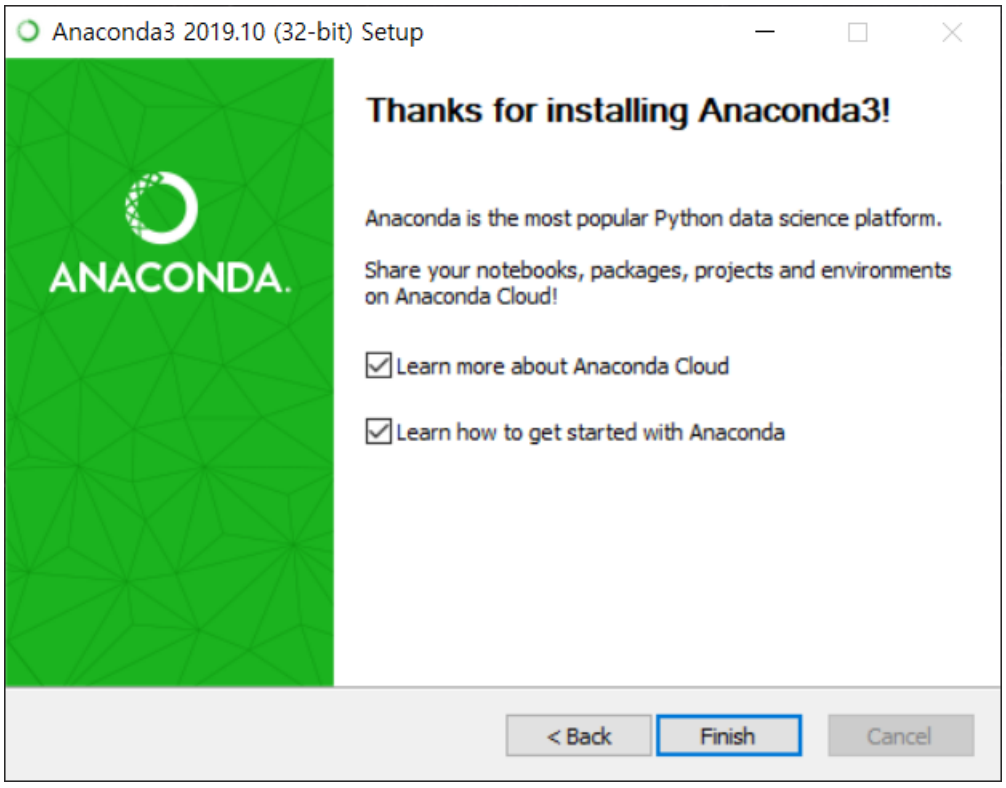
__8. 설치 작업이 완료되었다면, Next 버튼을 클릭합니다.



__9. 정상적으로 설치 완료되었으며, Next 버튼을 클릭합니다.



__10. 마지막으로 Finish 버튼을 클릭합니다.



Section 2: Python 정상 설치 여부 확인

Python 환경 구축이 정상적으로 되었는지 여부를 확인하는 작업을 진행합니다.

Spyder Editor 는 파이썬 코드를 실행하여, 그 결과를 바로 확인할 수 있는 기능을 제공합니다.
Spyder 는 IPython console, Text Editor, Exloper 세부분으로 구성되어 있으며, 파이썬 코드를 실행하고 그 결과를 확인하기 위해서, IPython console 를 사용합니다.

Anaconda3 항목의 “Spyder” 메뉴를 선택합니다.

__1. IPython console 에 Python 이 정상처리 되는지 확인합니다.

print(“Hello world”) 을 입력하고 Enter 를 Click 합니다.

Hello world 가 Console 에 보이는 것을 확인할 수 있습니다.

__2. IPython console 에서 자료구조 함수 중에서 Dict(딕셔너리)를 이용해 처리되는 결과 값을 확인해 봅니다.

Prices = {‘BTC’ : 8300000, ‘XRP’ : 514} 딕셔너리 배열을 구성하기 위해서 아래와 같이 실행합니다.

```
prices = {}
prices[‘BTC’] = 8300000
prices[‘XRP’] = 514
print(prices)
```

{‘BTC’ : 830000, ‘XRP’ : 514}

__3. Dict(딕셔너리)에서 Key 값만 추출해서, 결과 값을 확인해 봅니다.

```
prices.keys()
```

dict_keys([‘BTC’, ‘XRP’])

__4. Dict(딕셔너리)에서 value 값을 추출해서, 결과 값을 확인해 봅니다.

prices.values()

dict_values([8300000, 514])

__5. Dict(딕셔너리)에서 저장된 데이터를 삭제해 봅니다.

del prices[‘XRP’]

print(prices) -> {“BTC’ : 8300000}

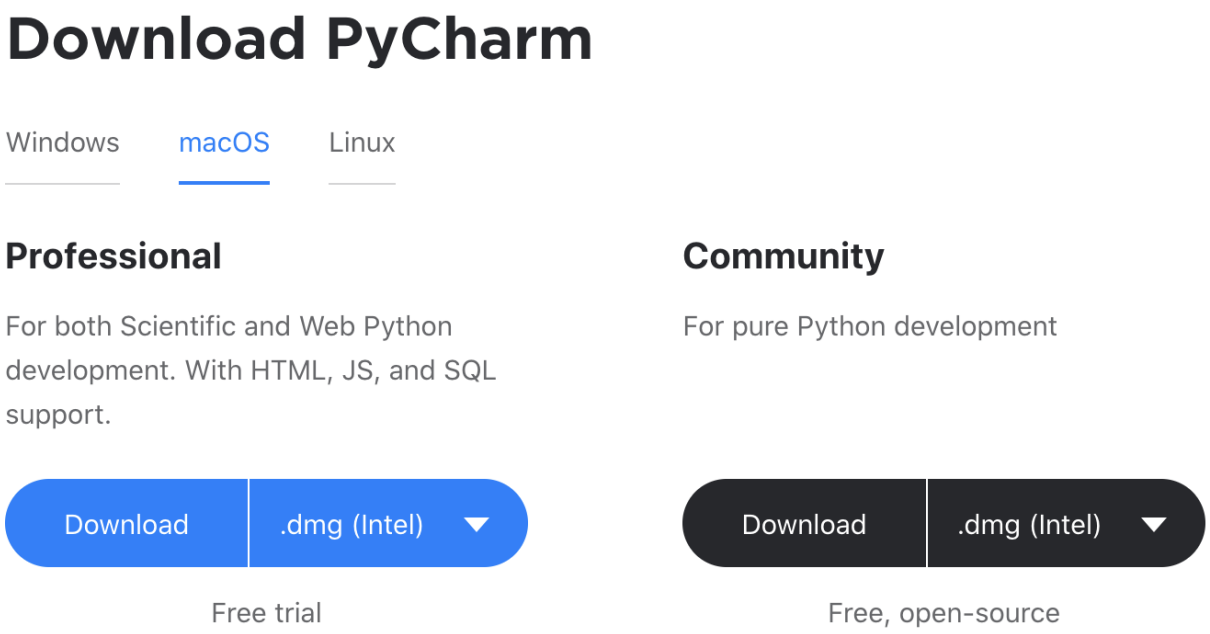
Section 3: PyCharm 설치 하기

PyCharm 은 Python 의 대표적인 IDE (Integrated Development Environment) 중 하나입니다.
Python 으로 개발을 할때, 다양한 기능을 활용하여, 개발을 도와주는 역할을 합니다.

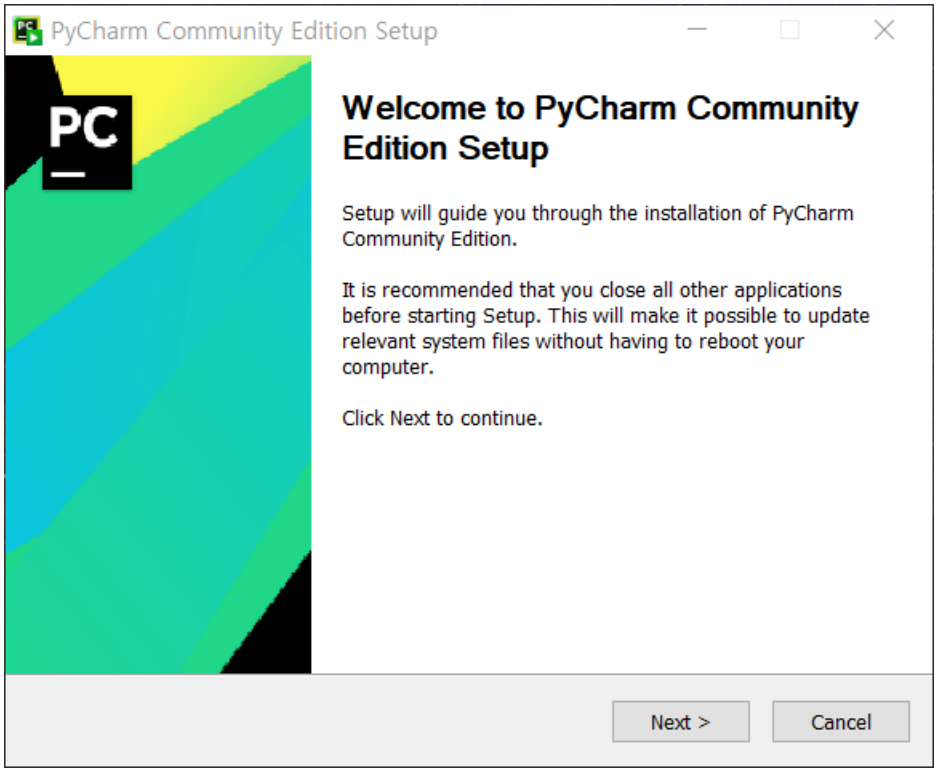
PyCharm 설치 작업을 진행합니다.

__1. PyCharm 웹 사이트에 접속해서, Community 버전을 다운로드 받습니다.

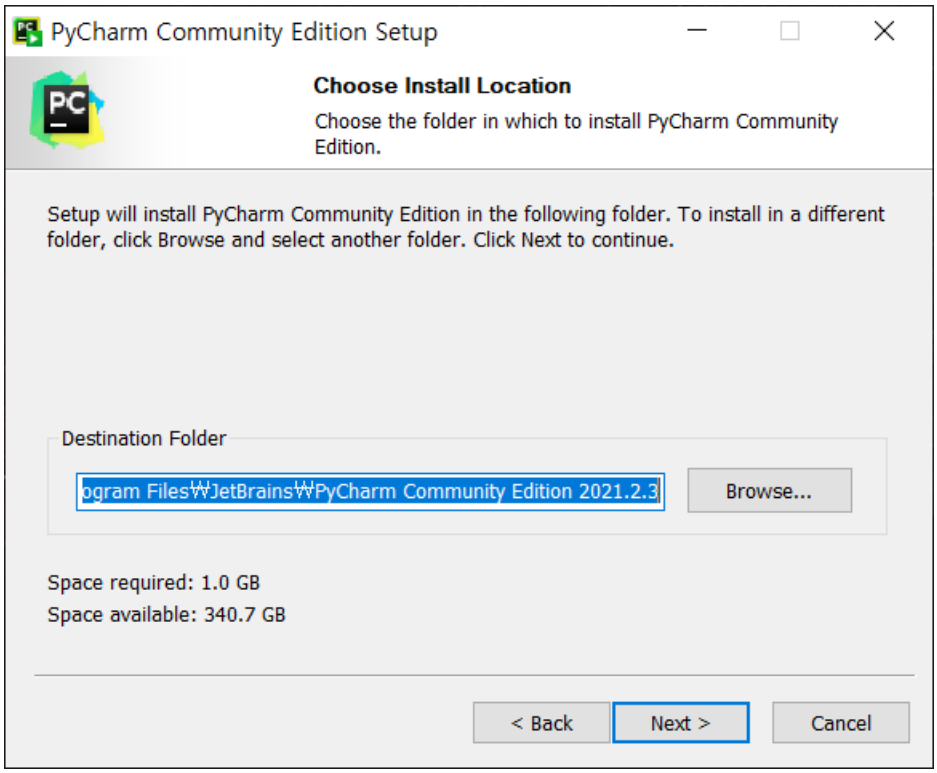
https://www.jetbrains.com/pycharm/download



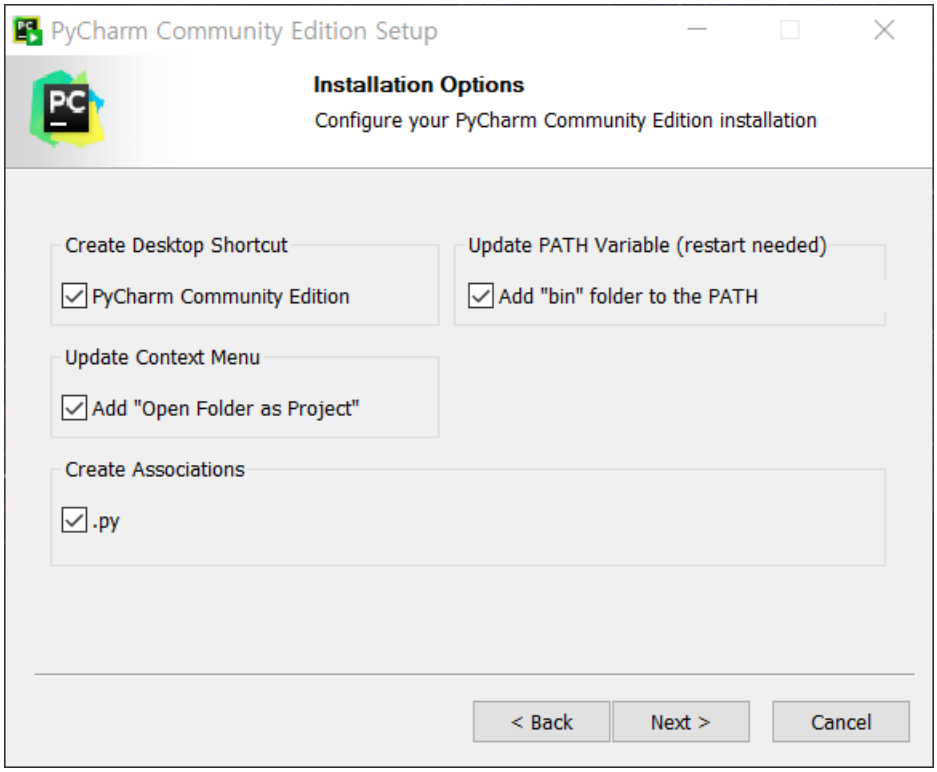
__2. 다운로드 받은 Community version 설치파일을 실행합니다.



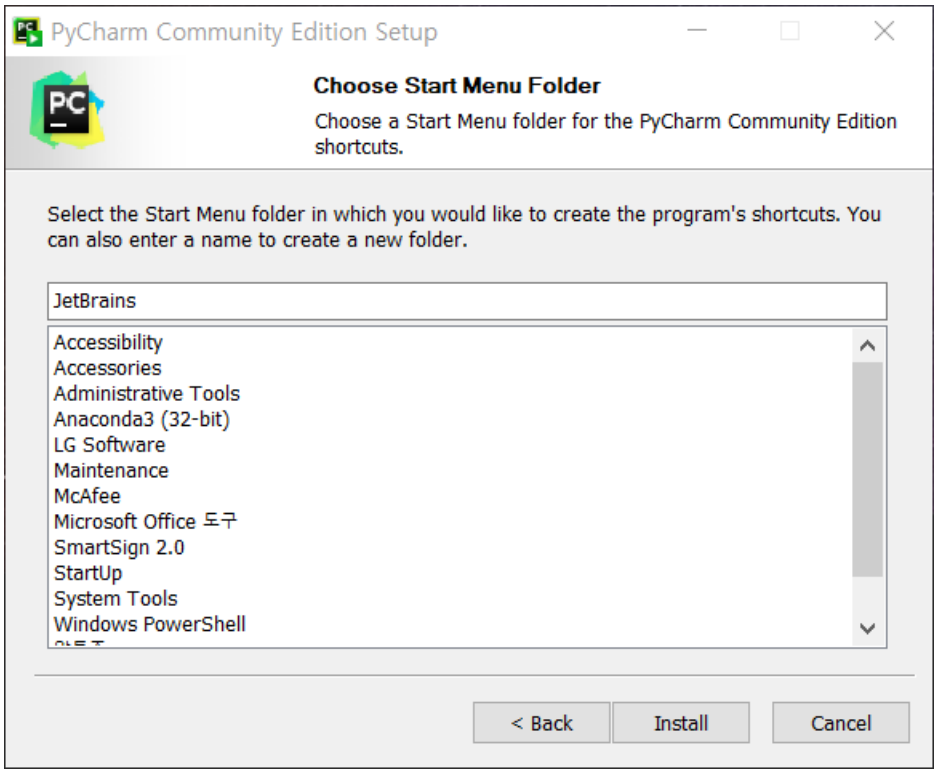
__3. 설치 경로를 확인 후, Next 버튼을 클릭합니다.



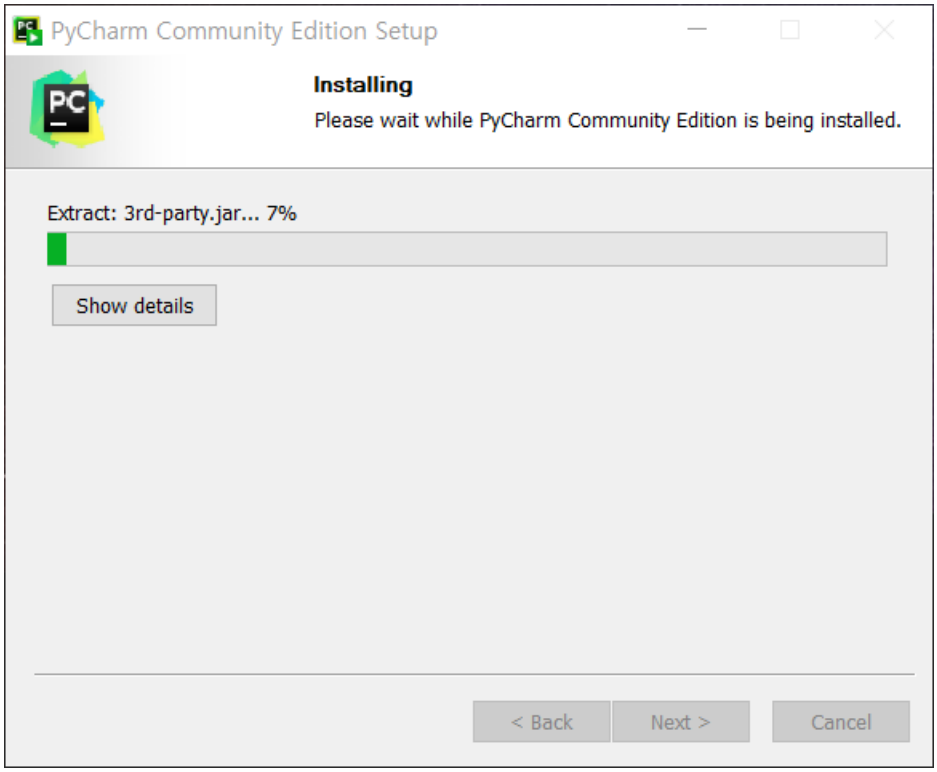
__4. 설치옵션을 선택 후, Next 버튼을 클릭합니다..



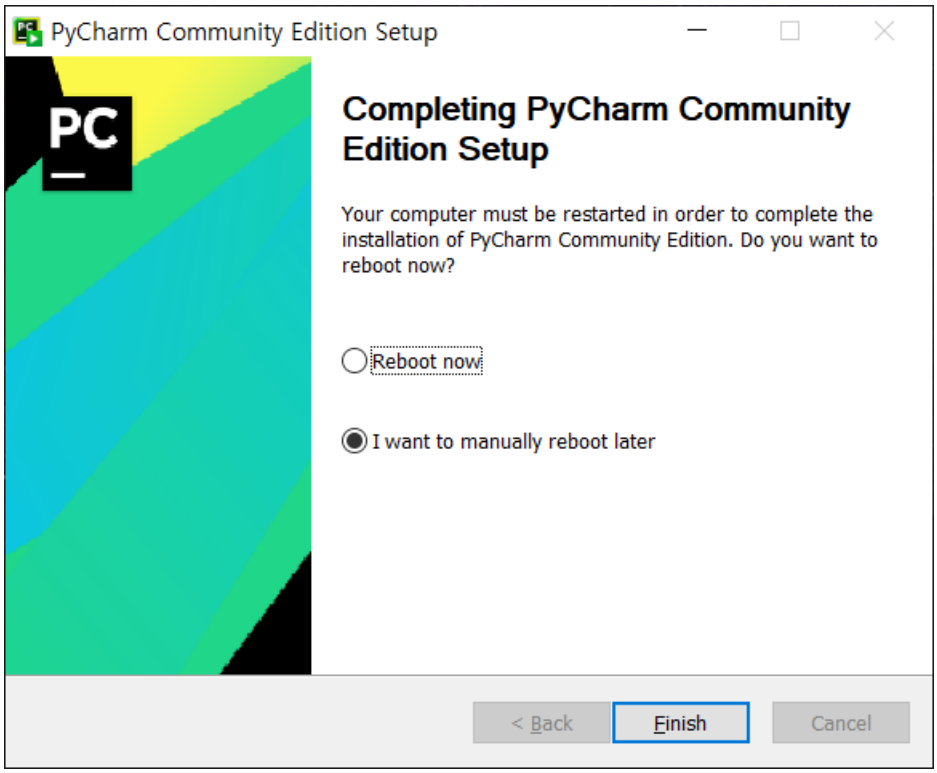
__5. 시작메뉴에 표시될 명칭을 확인후, Install 버튼을 클릭합니다.



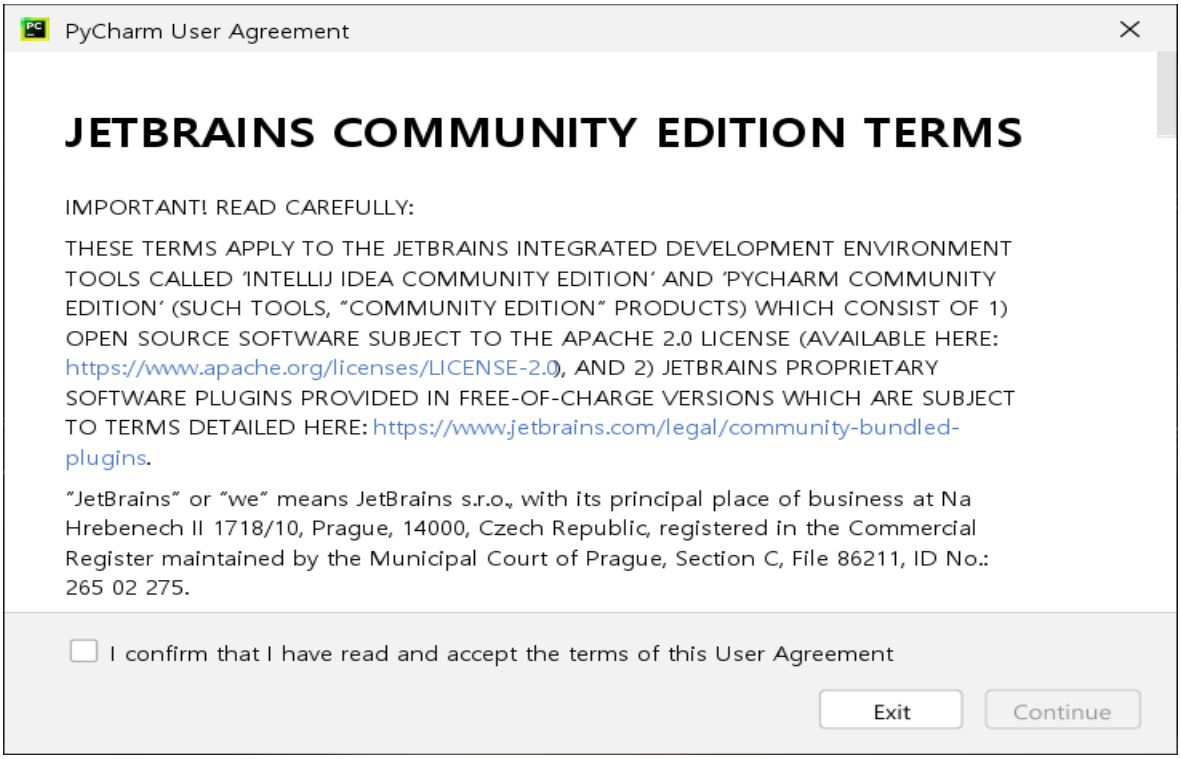
__6. 설치 작업이 진행됩니다.



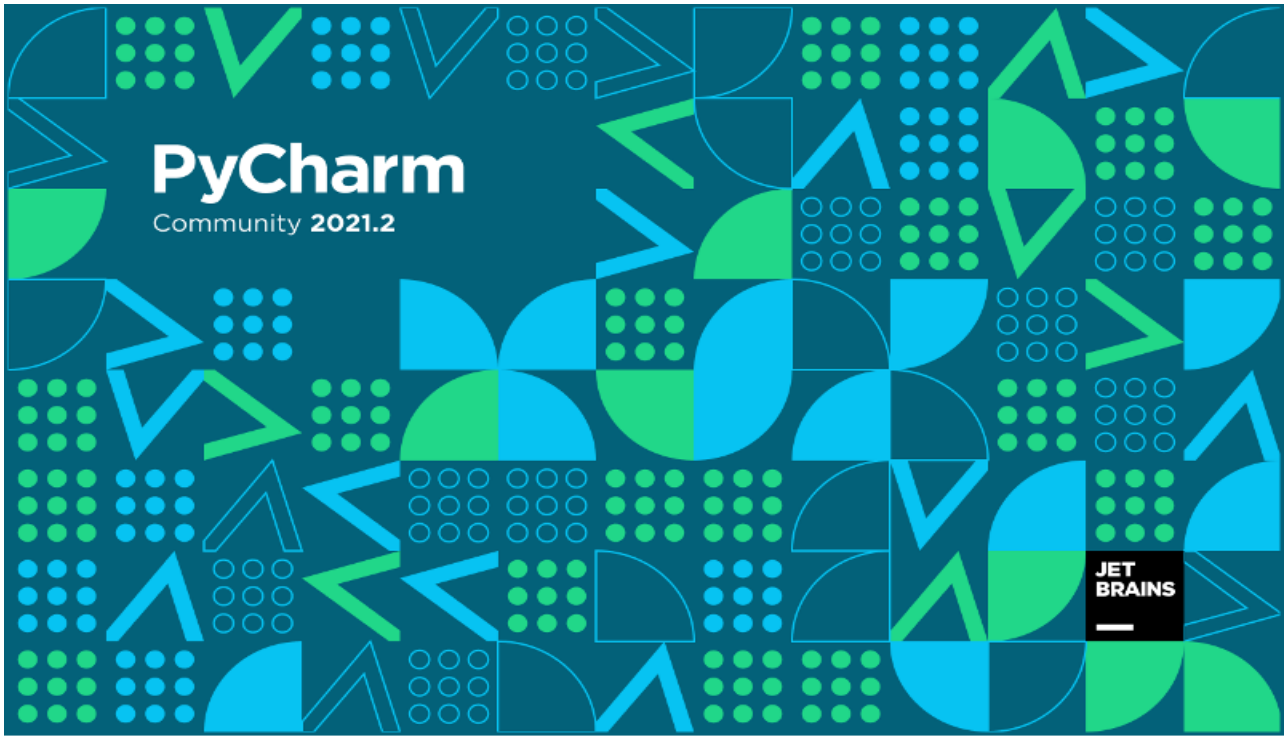
__7. 설치 작업이 정상적으로 진행되었다면, 재부팅 여부를 확인후, Finish 버튼을 클릭합니다..



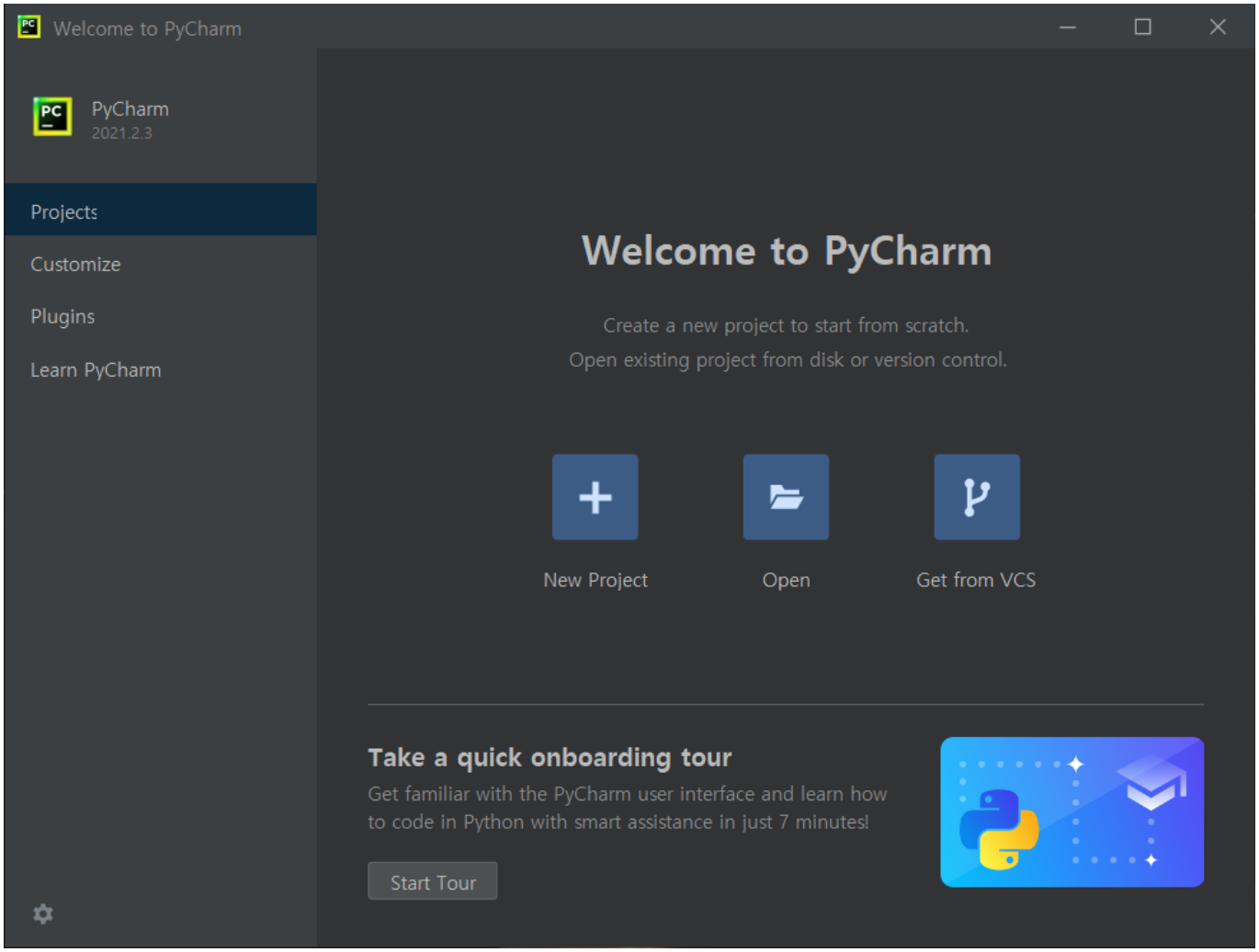
__8. 설치 완료 후, 바탕화면에 PyCharm Community Edition 이라는 아이콘이 보이게 됩니다. 아이콘을 클릭합니다. PyCharm 을 처음 실행하게 되면, User Agreement 창이 보이게 되는데, 여기서 I confirm 후, Continue 버튼을 클릭합니다.



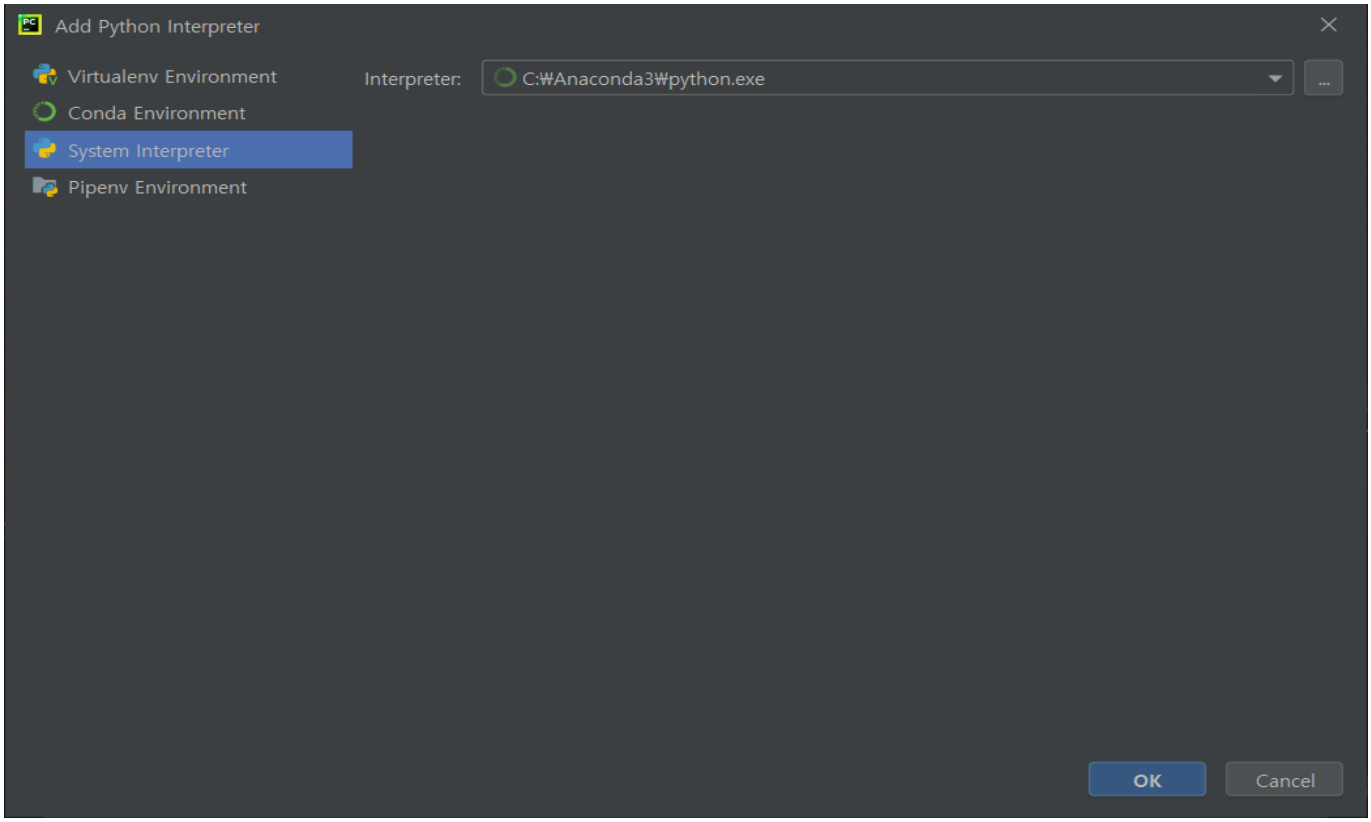
__9. PyCharm 툴이 정상적으로 실행됩니다.

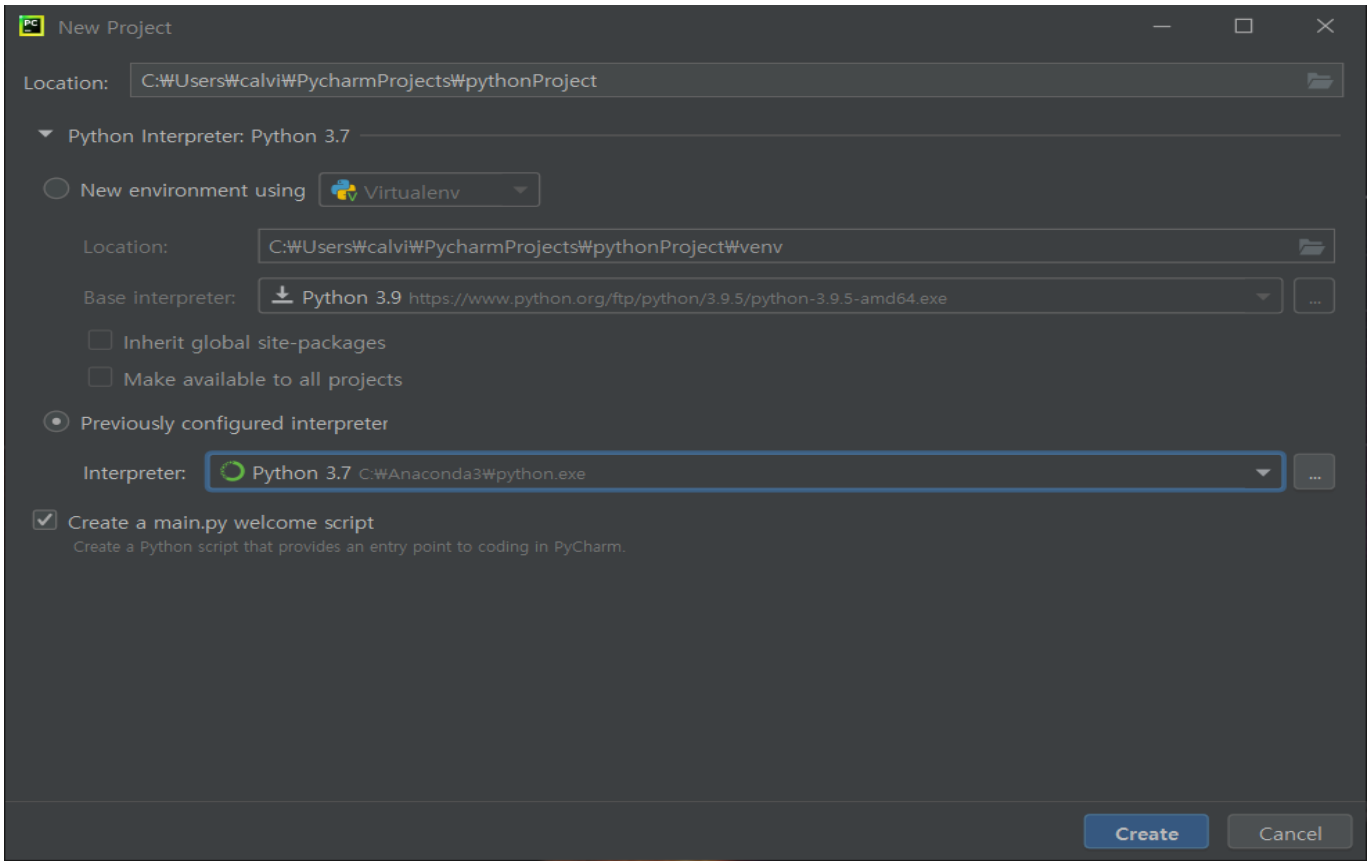


__10. PyCharm 툴이 실행되면, Welcome to PyCharm 윈도우 툴이 보이게 됩니다.

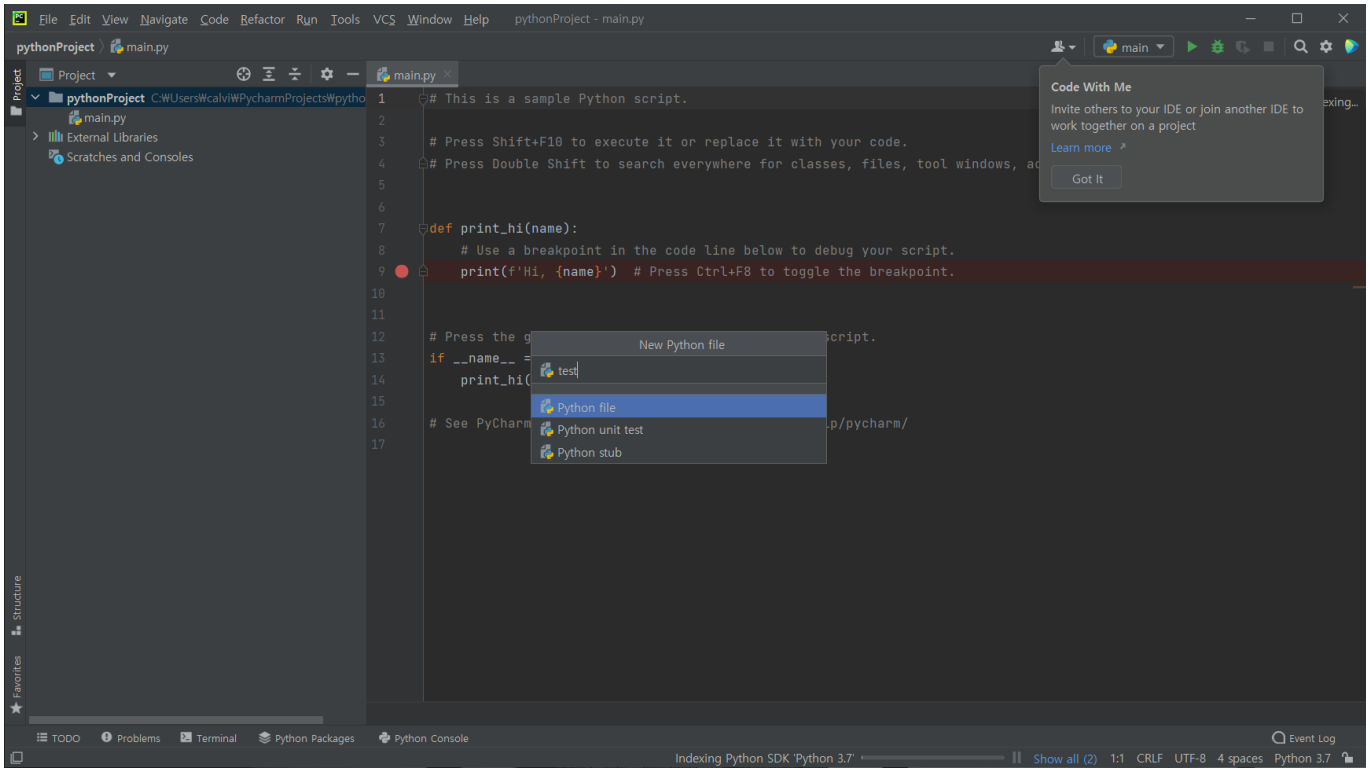


__11. 정상적으로 설치되었는지를 확인하기 위해, Python 파일을 하나 생성하여, 실행해 봅니다. 실행전, New Project 를 클릭 후, Interpeter 에서 System Interpeter 에서 Anaconda3 을 선택합니다.


Add Python Interpreter
Virtualenv Environment
Conda Environment
System Interpreter
Pipenv Environment
Interpreter:
C:₩Anaconda3₩python.exe
...
OK
Cancel


New Project
Location:
C:₩Users₩calvi₩PycharmProjects₩pythonProject
Python Interpreter: Python 3.7
New environment using
Virtualenv
Location:
C:₩Users₩calvi₩PycharmProjects₩pythonProject₩venv
Base Interpreter:
Python 3.9 https://www.python.org/ftp/python/3.9.5/python-3.9.5-amd64.exe
Inherit global site-packages
Make available to all projects
Previously configured interpreter
Interpreter:
Python 3.7 C:₩Anaconda3₩python.exe
Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm.
Create
Cancel

__12. 생성된 Project 에서 마우스 오른쪽 버튼을 클릭하면, New Python file 을 선택하여, 파일명을 입력 후, 파일을 생성합니다.

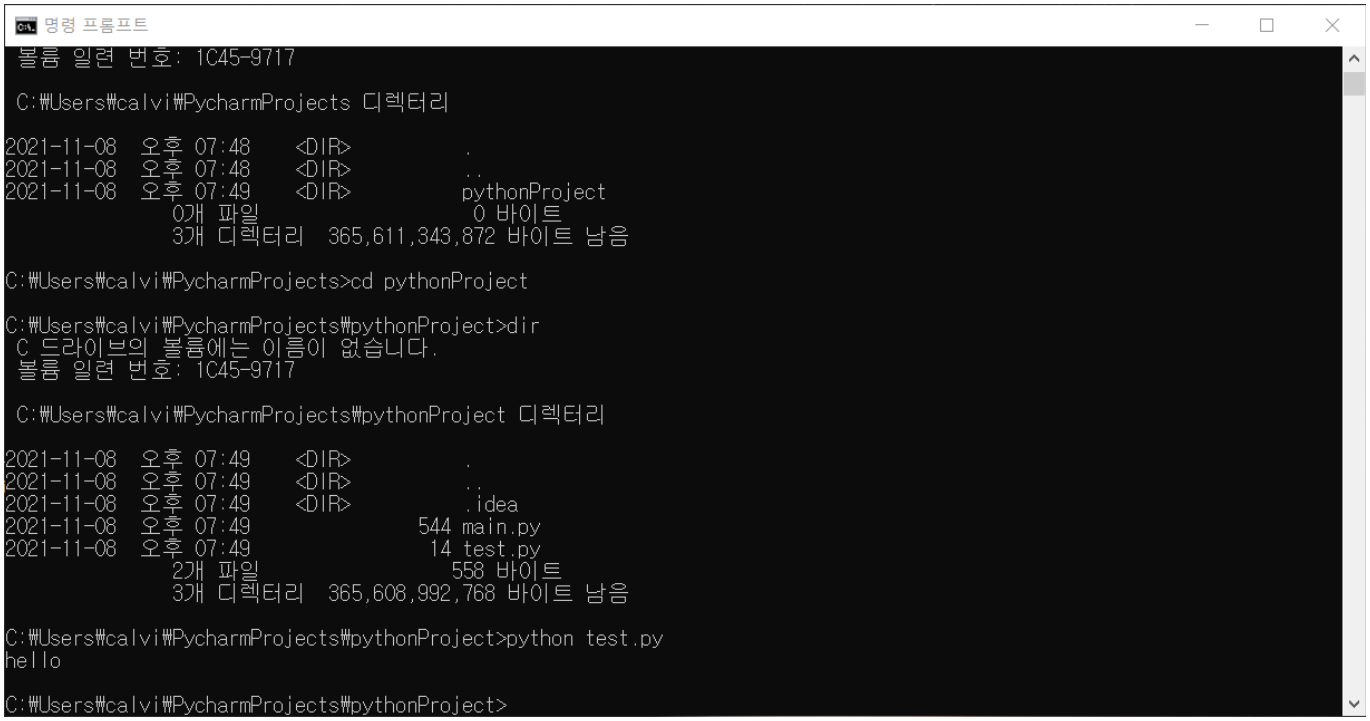


__13. 파이썬 로직이 정상적으로 처리되는지 확인합니다.
print(“hello”)

로직을 입력 후, 마우스 오른쪽 버튼 클릭시 보이는 메뉴에서 run 을 실행합니다.

정상적으로 Hello 문구가 보이는지 확인합니다.

다른 방법으로는 윈도우 Prompt 창에서 생성한 프로젝트의 디렉토리로 이동 후, python test.py 를 실행하여 결과를 확인합니다.



정상적으로 결과가 보인다면, Python 을 컴파일하고 실행할 수 있는 환경이 준비되었습니다.

Section 4: PyQt

파이썬의 기본 환경은 콘솔 기반의 터미널 창에서 명령어를 입력하여, 처리하는 형태에서의 제약사항인 GUI (Graphical User Interface) 기반을 지원하기 위한 모듈을 활용할 수 있습니다.

TKInter / wxPython / PyQt 모듈을 통해서 GUI 형태의 기반 프로그램을 구성할 수 있습니다.
Section 1 에서 설치한 아나콘다에는 PyQt5 가 포함되어 있습니다.

__1. PyQt 모듈 기반으로 정상 동작하지는 여부를 확인합니다.

.

```
import sys
from PyQt5.QtWidgets import *

app = QApplication(sys.argv)

btn = QPushButton("Trading")
btn.show()

app.exec_()
```

기본적인 Python 프로그램은 라인상에 코딩된 코드를 처리하고 종료되게 되지만, GUI 프로그램은 창에 있는 X 버튼을 클릭하기전까지는 종료가 되지 않도록 처리하는 이벤트 루프가 필요합니다.

__2. 이벤트루프가 적용되지 않는 GUI 프로그램 코드의 동작을 확인합니다.

```
import sys
from PyQt5.QtWidgets import *

app = QApplication(sys.argv)
label = QLabel("Trading")
label.show()
```

__3. 이벤트루프가 적용되어 있는 GUI 프로그램 코드의 동작을 확인합니다.

```
import sys
from PyQt5.QtWidgets import *

app = QApplication(sys.argv)
label = QLabel("Trading")
label.show()
app.exec_()
```

__4. GUI 프로그램의 기본 코드를 실행해 봅니다.

```
import sys
from PyQt5.QtWidgets import *

class TestWindow(QMainWindow):
    def __init__(self):
        super().__init__()

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

__5. 윈도우 크기 조절을 setGeometry() 함수를 이용해서 실행해 봅니다.

```
import sys
from PyQt5.QtWidgets import *


class TestWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 200, 300, 400)


app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

__6. 윈도우 타이틀 및 타이틀 아이콘을 반영해 봅니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *


class TestWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 200, 300, 400)
        self.setWindowTitle("PyQt")
        self.setWindowIcon(QIcon("icon.png"))


app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

__7. 윈도우에 버튼을 추가해 봅니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *


class TestWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 200, 300, 200)
        self.setWindowTitle("PyQt")
        self.setWindowIcon(QIcon("icon.png"))

        btn = QPushButton("button1", self)
        btn.move(10, 10)

        btn2 = QPushButton("button2", self)
        btn2.move(10, 40)

        btn3 = QPushButton("button3", self)
        btn3.move(10, 70)



app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

버튼이 정상적으로 생성되어, 윈도우 보이는 것을 확인하였다면, 버튼에 이벤트를 반영하여, 버트 클릭시 이벤트가 발생하는지 확인해 봅니다.

__8. 버튼에 이벤트를 반영해 봅니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *


class TestWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 200, 300, 200)
        self.setWindowTitle("PyQt")
        self.setWindowIcon(QIcon("icon.png"))

        btn = QPushButton("button1", self)
        btn.move(10, 10)
        btn.clicked.connect(self.btn_clicked)

        btn2 = QPushButton("button2", self)
        btn2.move(10, 40)
        btn2.clicked.connect(self.btn2_clicked)

        btn3 = QPushButton("button3", self)
        btn3.move(10, 70)
        btn3.clicked.connect(self.btn3_clicked)


    def btn_clicked(self):
        print("button1 clicked!")


    def btn2_clicked(self):
        print("button2 clicked!")

    def btn3_clicked(self):
        print("button3 clicked!")


app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

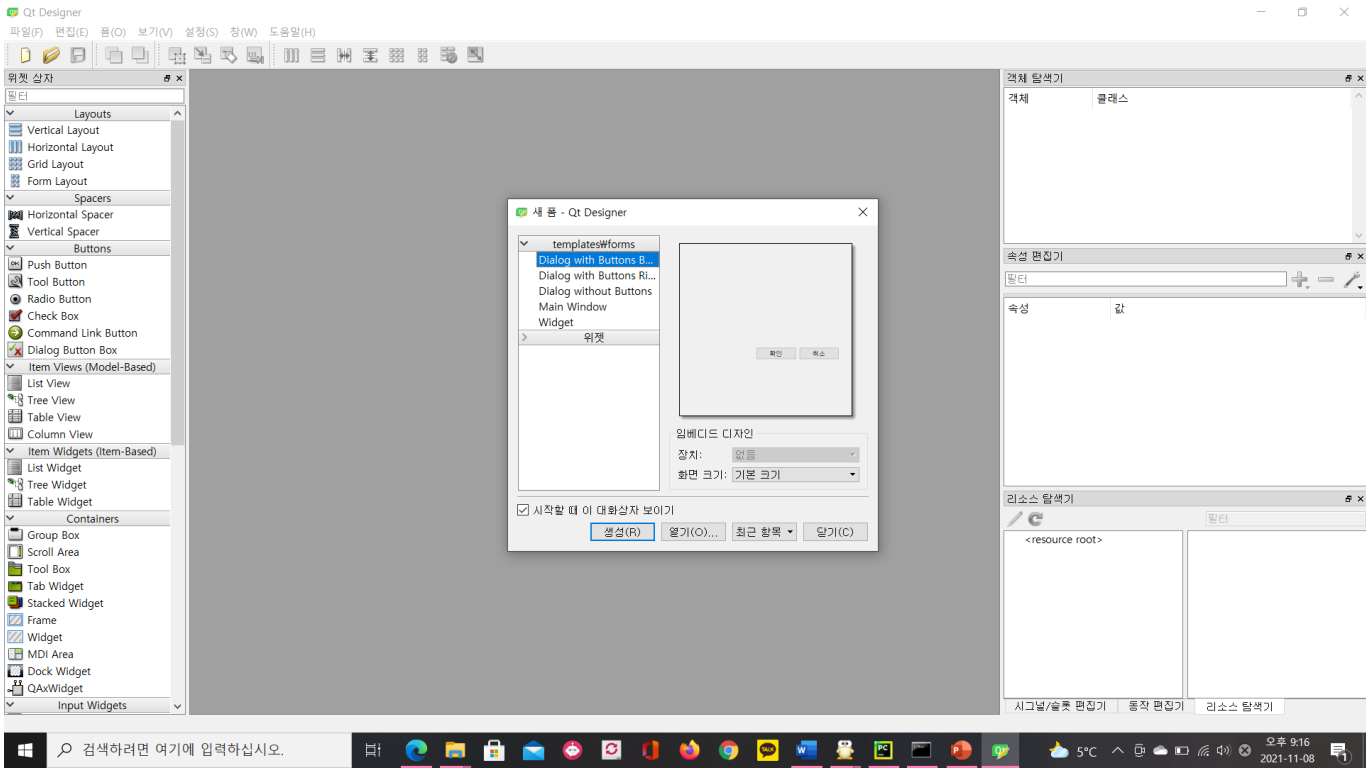
Section 5: Qt Designer

아나콘다에서는 UI 컴포넌트를 통해서 UI 를 쉽게 구성할 수 있는 툴을 제공하는데 툴 이름이 Qt Designer 입니다.

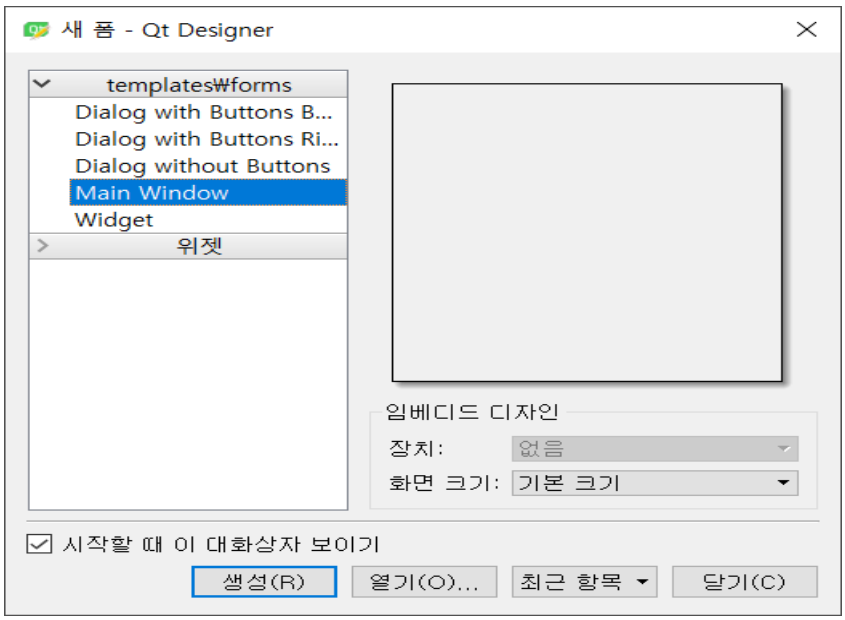
Qt Designer 는 아나콘다 설치 디렉터리에서 designer.exe 파일을 클릭하면 실행됩니다.

```
C:\Anaconda3\Library\bin
```

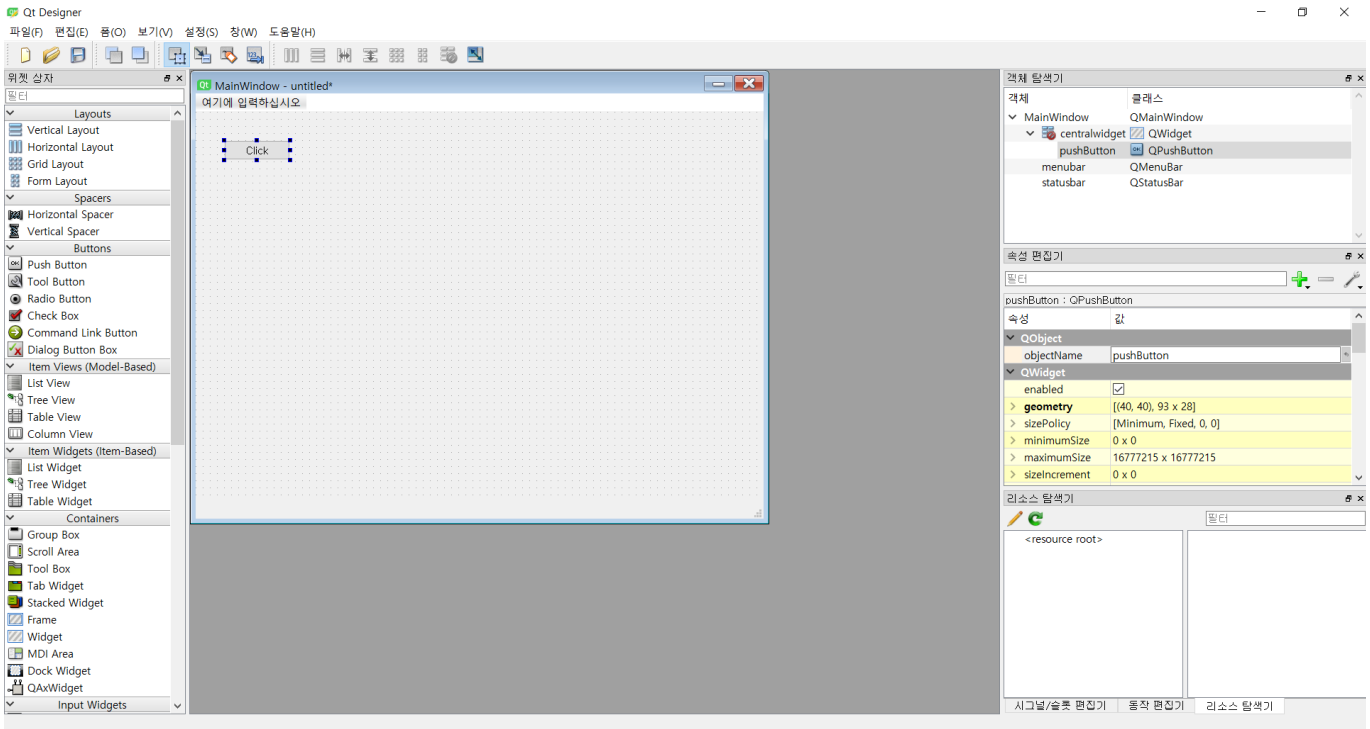
__1. Qt Designer 가 실행되면, 프로그램 창이 보이게 됩니다.



__2. 새폼에서 Main Window 를 선택후, 생성 버튼을 클릭합니다.



__3. 위젯 상자에서 Push Button 을 Drag 해서 MainWindow 창에 Drop 한후, Push Button 이름을 Click 으로 입력합니다.



__4. 파일 메뉴에서 “다른이름으로 저장” 메뉴를 선택해서 파일명을 입력후, 저장합니다.

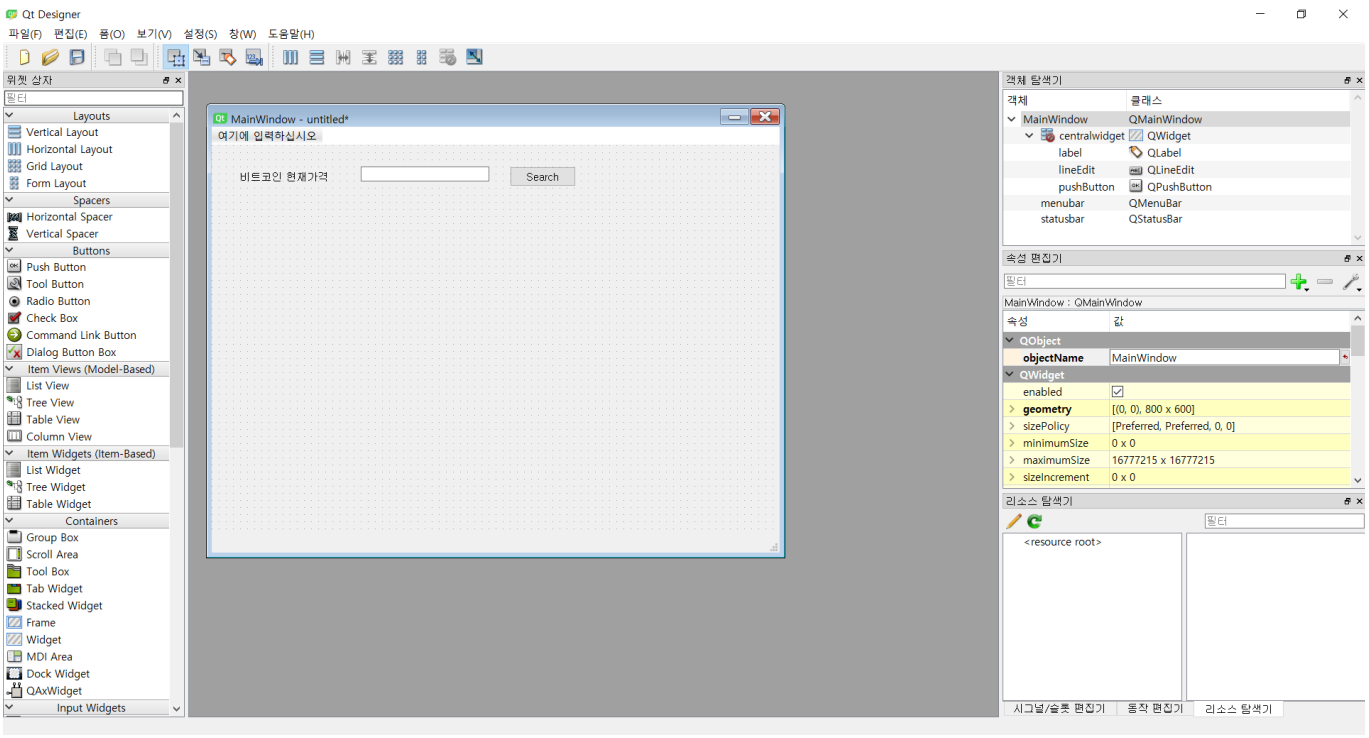
__5. PyCharm 툴에서 Qt Designer 에서 생성한 UI 를 호출하고, UI 에서 포함되어 있는 버튼에 이벤트를 적용하는 작업을 진행합니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

form_class = uic.loadUiType("testwindow.ui")[0]


class TestWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.btn_clicked)

    def btn_clicked(self):
        print("button clicked!")

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

Section 6: 시세조회기 만들기 (Using **Qt Designer)**

Qt Desginer 를 통해서 거래소의 시세를 조회하는 UI 를 생성하고, 거래소에서 제공하는 파이썬 모듈을 설치하여, 시세정보를 조회해 봅니다.

__1. Qt Designer 에서, 시세 조회 UI 를 생성합니다.



MainWindow 에 아래와 같이 위젯을 Drag&Drop 을 통해 시세조회 UI 를 구성합니다.

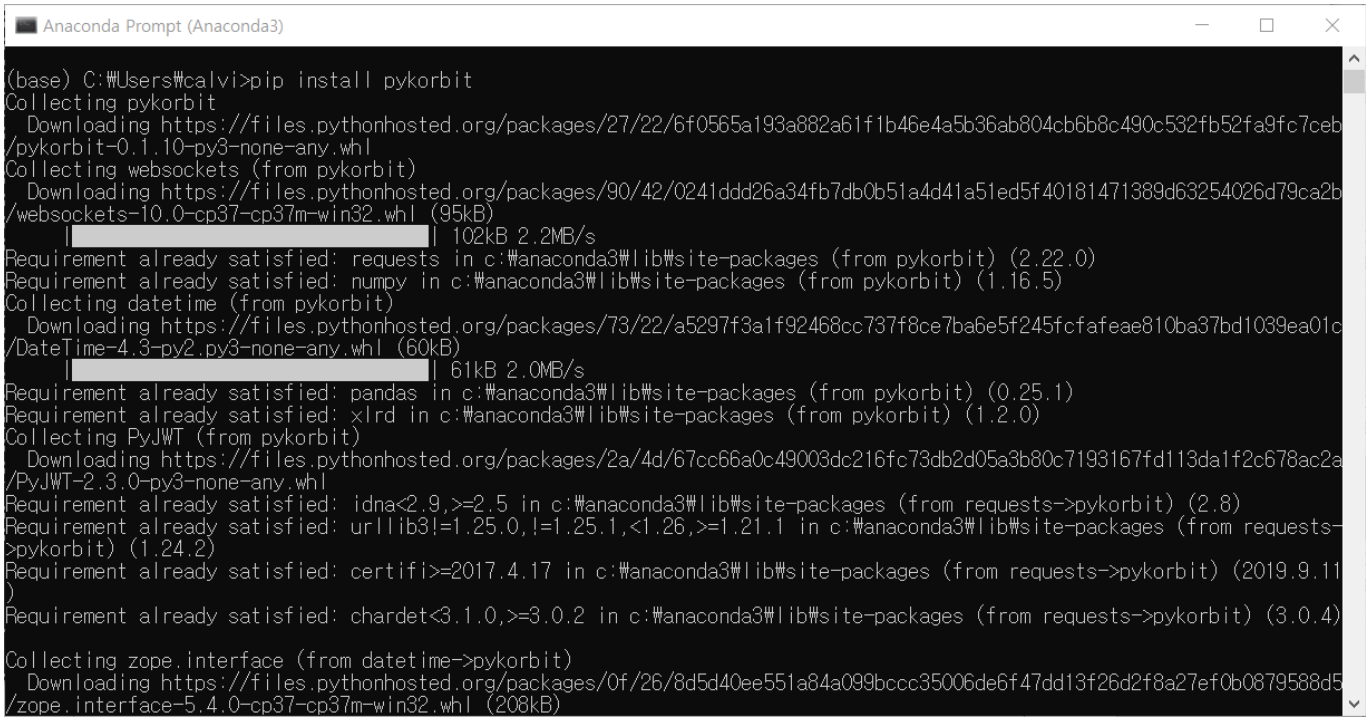
- Label
- LineEdit
- Pushbutton

__2. 생성된 UI 를 저장합니다.

__3. 거래소의 시세조회 정보를 가져오기 위해서, 거래소가 제공하는 파이썬 모듈을 설치합니다.

Anaconda3 메뉴에서 Anaconda Prompt (Anaconda3) 아이콘을 클릭하면, Anaconda Prompt 창이 띄게 됩니다. 여기서 아래의 명령어를 실행하면, 거래소의 모듈이 설치됩니다.

```
pip install pykorbit
```



__4. 거래소 모듈이 정상적으로 설치되었다면, PyCharm 툴에서 거래소 모듈을 연동하여, Qt Designer 툴에서 생성한 UI 를 기반으로 시세 정보를 조회하는 로직을 작성합니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
import pykorbit

form_class = uic.loadUiType("search.ui")[0]

class TestWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.inquiry)

    def inquiry(self):
        price = pykorbit.get_current_price("BTC")
        print(price)

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

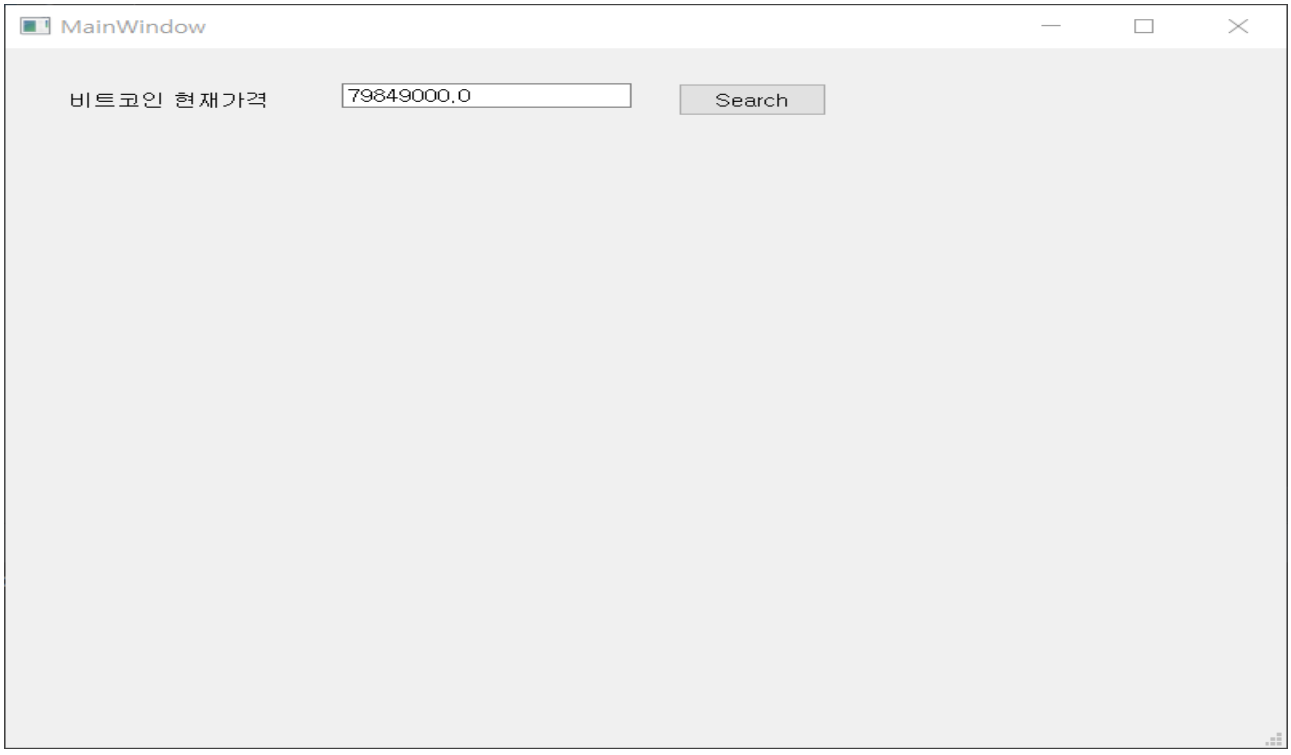
__5. 시세 정보 조회 로직을 실행하면, Window 창이 보이게 되는데, 여기서 Search 버튼을 클릭하면 시세 정보를 가져오게 됩니다.

```
C:\Anaconda3\python.exe C:/Users/calvi/Documents/Source/06_01.py
79870000.0
79870000.0
79870000.0
```

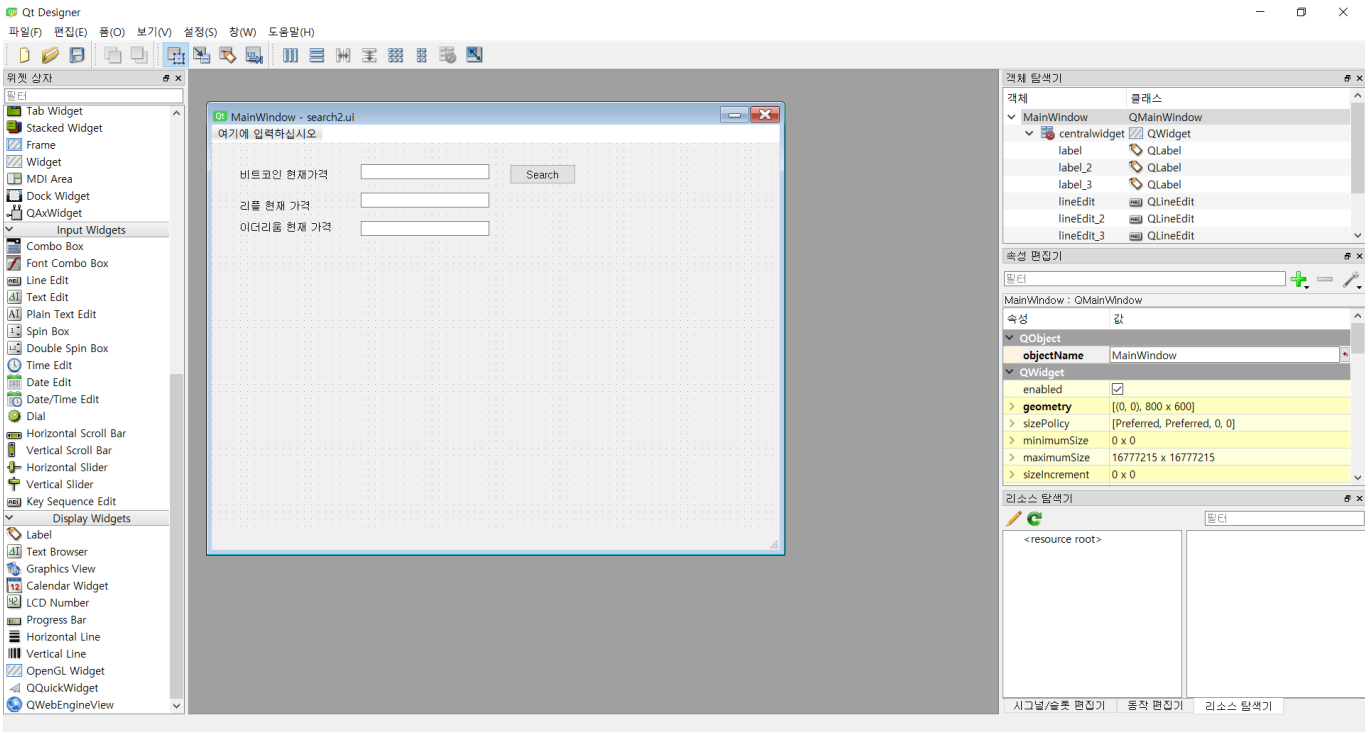
정상적으로 시세 정보를 가져오는 것을 확인하였다면, 시세조회 로직상에서 아래와 같은 함수를 추가합니다.

```
self.lineEdit.setText(str(price))
```

함수를 반영 후, Run 을 실행하여, Search 버튼을 통해, 시세정보가 LineEdit 에 반영되는 것을 확인합니다.



__6. 거래소 모듈을 통해서, 타 암호화폐의 시세정보를 추가하여, 조회하는 화면을 Qt Designer 를 통해서 생성합니다. (XRP, ETC 추가)



__7. 시세조회 로직에서 새로추가된 XRP, ETC 시세 정보를 가져오는 로직을 추가합니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
import pykorbit

form_class = uic.loadUiType("search2.ui")[0]

class TestWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.inquiry)

    def inquiry(self):
        price = pykorbit.get_current_price("BTC")
        price2 = pykorbit.get_current_price("XRP")
        price3 = pykorbit.get_current_price("ETC")
        print(price)
        print(price2)
        print(price3)

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

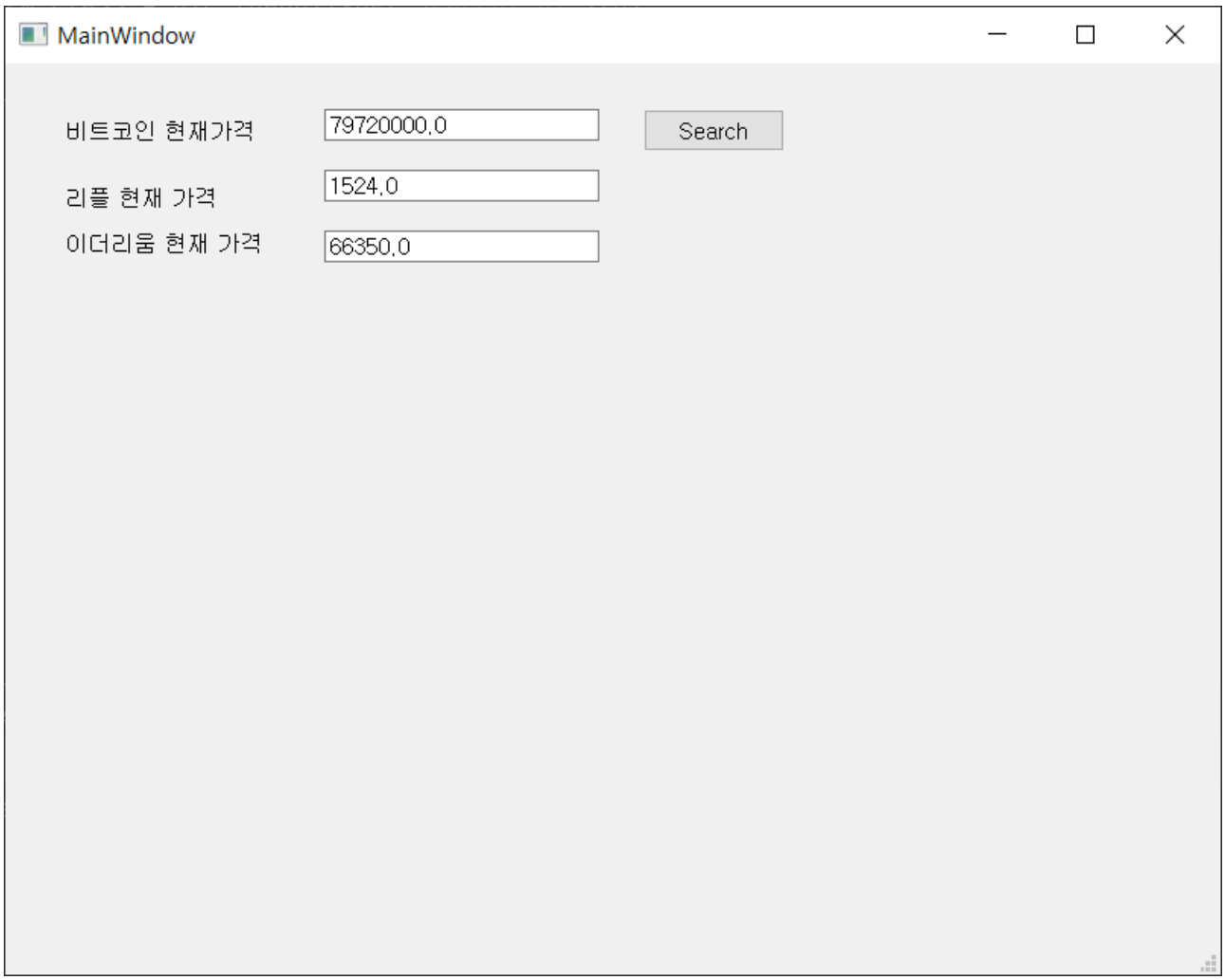
XRP, ETC 시세정보를 가져오는 로직을 반영 후, 정상으로 가져오는지 확인합니다.

```
C:\Anaconda3\python.exe C:/Users/calvi/Documents/Source/06_03.py
79719000.0
1526.0
66350.0
```

__8. 시세조회 로직에서 UI 의 LineEdit 에 시세정보를 입력하는 로직을 추가합니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
import pykorbit

form_class = uic.loadUiType("search2.ui")[0]

class TestWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.inquiry)

    def inquiry(self):
        price = pykorbit.get_current_price("BTC")
        price2 = pykorbit.get_current_price("XRP")
        price3 = pykorbit.get_current_price("ETC")
        self.lineEdit.setText(str(price))
        self.lineEdit_2.setText(str(price2))
        self.lineEdit_3.setText(str(price3))

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

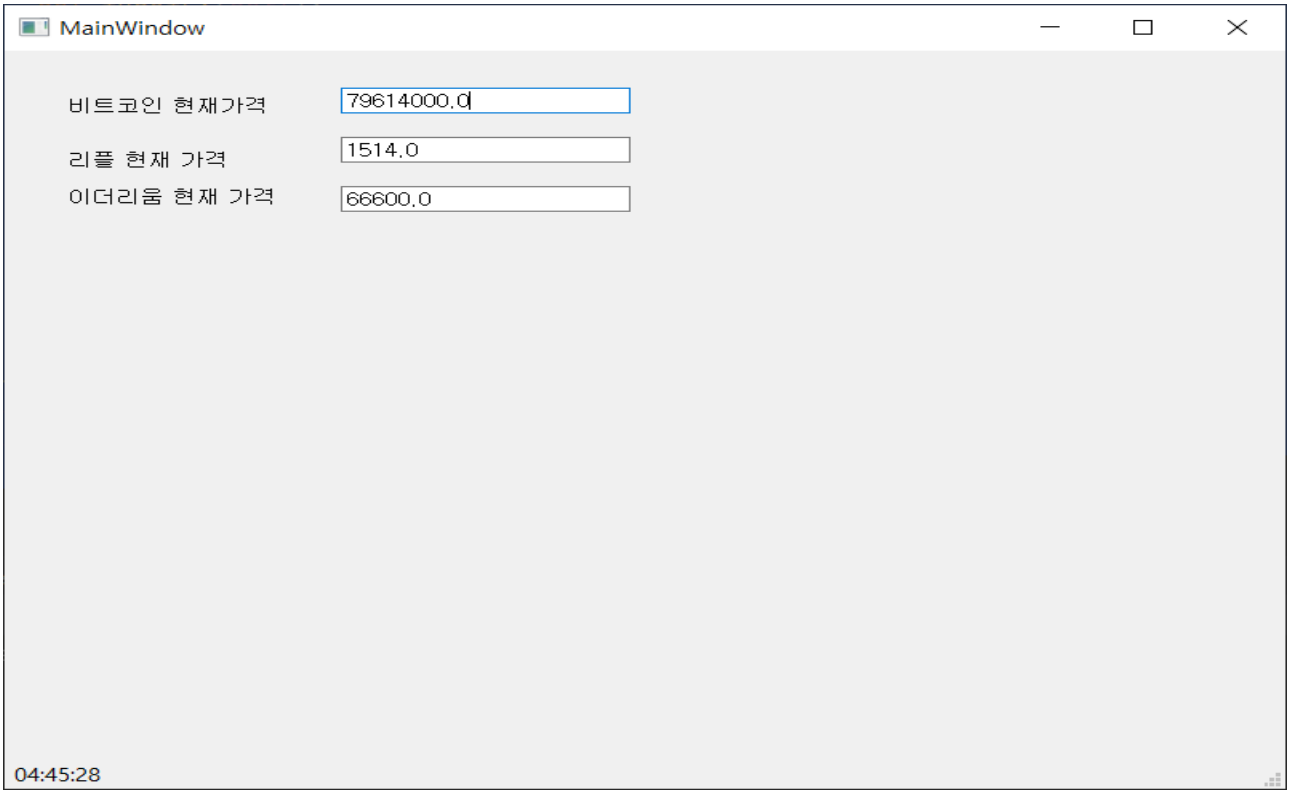
__9. 시세조회 UI 상에서 시세정보가 정상적으로 LineEdit 에 입력되는지 확인합니다.



__10. 주기적으로 시세정보를 가져오는 로직을 만들어 봅니다.
주긱적으로 데이터를 가져오는 방식은 Qtimer 를 활용하여, Interval 을 통해서 로직을 구현할 수 있습니다.

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
from PyQt5.QtCore import *
import pykorbit

form_class = uic.loadUiType("search3.ui")[0]

class TestWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.timer = QTimer(self)
        self.timer.start(1000)
        self.timer.timeout.connect(self.inquiry)

    def inquiry(self):
        cur_time = QTime.currentTime()
        str_time = cur_time.toString("hh:mm:ss")
        self.statusBar().showMessage(str_time)
        price = pykorbit.get_current_price("BTC")
        price2 = pykorbit.get_current_price("XRP")
        price3 = pykorbit.get_current_price("ETC")
        self.lineEdit.setText(str(price))
        self.lineEdit_2.setText(str(price2))
        self.lineEdit_3.setText(str(price3))

app = QApplication(sys.argv)
window = TestWindow()
window.show()
app.exec_()
```

로직을 Run 하게 되면, 윈도우 창에서 1 초 기준으로 시세정보를 가져오는 것을 확인합니다.



지금까지 진행한 거래소의 모듈외에 타 거래소의 모듈을 통해서 테스트를 진행할 수 있습니다.

연습문제

pip install pyupbit 을 설치 후, 파이썬 파일을 하나 생성하여, 거래소의 티켓 조회 및 현재가 조회 로직을 반영하여, 기존에 생성한 UI 에 적용하는 로직을 구현해 보세요

Hint

- 파이썬 로직에는 import pyupbit
- 티켓 조회 : tickers = pyupbit.get_tickers()
- 현재가 조회 : price = pyupbit.get_current_price(“BTC”)