파이썬 프로그래밍
강의노트 #15

터틀 그래픽스 활용

터틀 그래픽스

터틀 그래픽스로 그림 그리기

- 터틀은 어린이 및 초보자가 파이썬을 쉽게 배울 수 있도록 만든 모듈인데, 거북이가 기어가는 모양대로 그림을 그린다고 해서 터틀이라고 함
- 터틀은 그림을 그리는 모듈이므로 Windows, 리눅스, macOS 그래픽 환경에서만 동작함
- 콘솔(터미널)만 있는 환경에서는 사용할 수 없음

ubunt : sudo apt install python3-tk

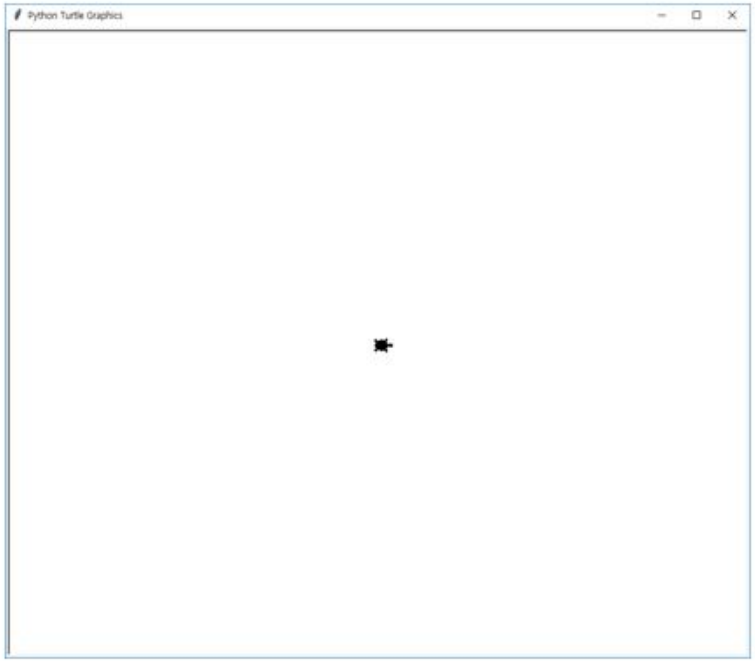
터틀 그래픽스

사각형 그리기

- IDLE을 실행하고 파이썬 셸에서 다음과 같이 입력해보자

```
>>> import turtle as t
>>> t.shape('turtle')
```

▼ **그림 21-1** 파이썬 터틀 그래픽스 창과 거북이



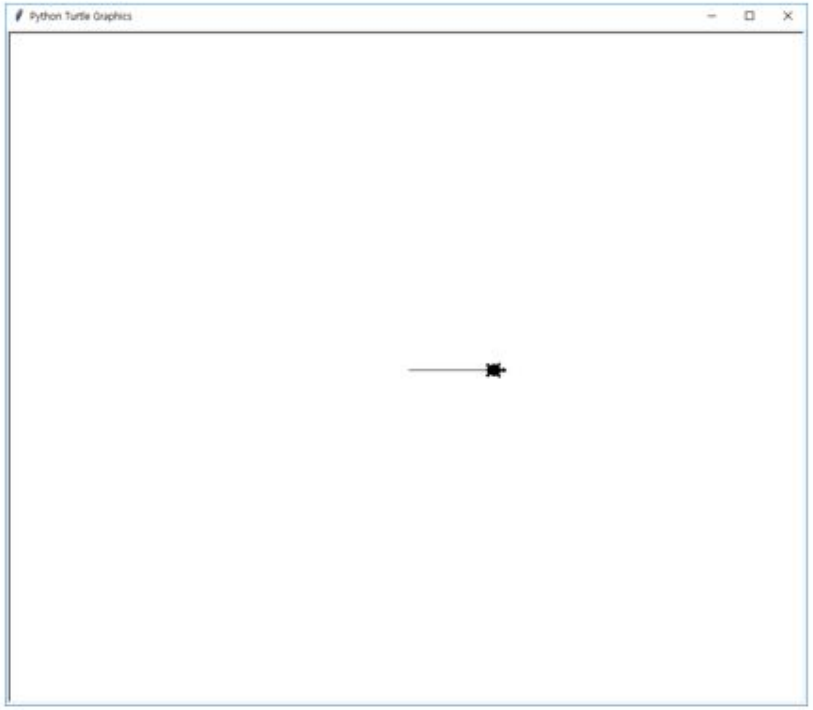
터틀 그래픽스

사각형 그리기

- 파이썬 터틀 그래픽스 창을 끄지 않은 상태에서 IDLE의 파이썬 셸 창에 다음과 같이 입력함

```
>>> t.forward(100)
```

▼ **그림 21-2** 거북이를 100픽셀만큼 앞으로 이동



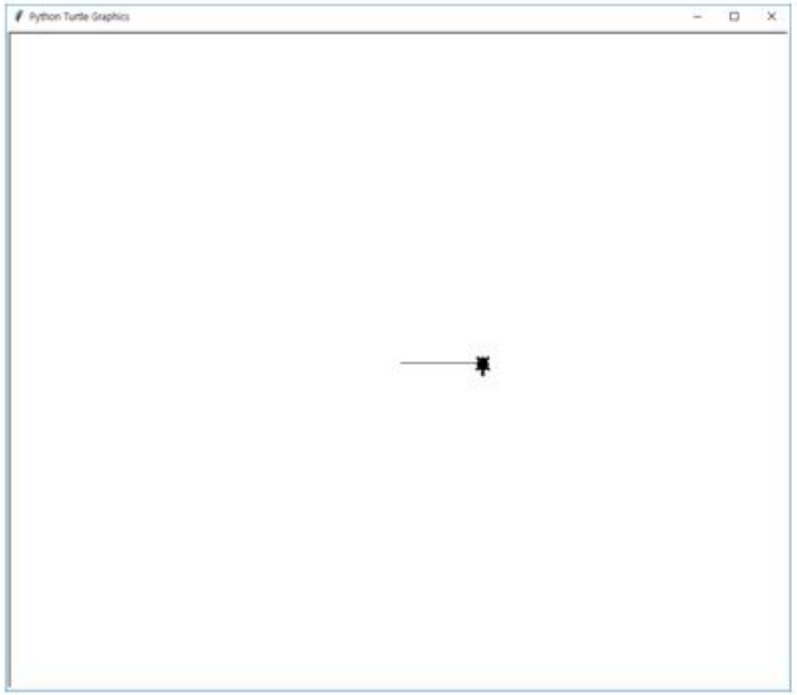
터틀 그래픽스

사각형 그리기

- 이번에는 거북이의 방향을 바꿔보자

```
>>> t.right(90)
```

▼ **그림 21-3** 거북이 방향을 바꿈

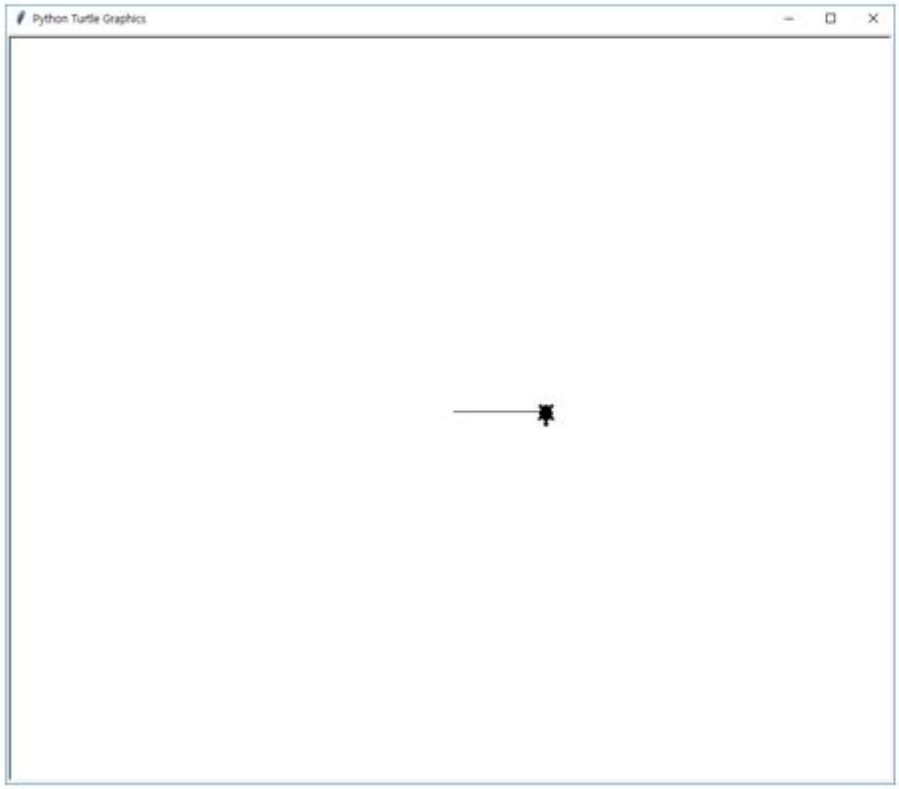


터틀 그래픽스

사각형 그리기

```
>>> t.forward(100)
```

▼ **그림 21-4** 거북이를 앞으로 이동시킴

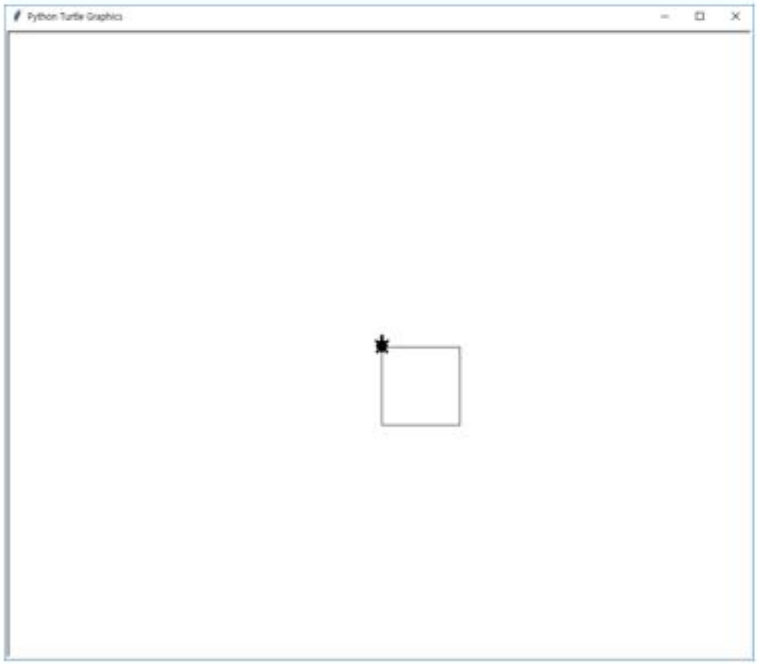


터틀 그래픽스

사각형 그리기

```
>>> t.right(90)
>>> t.forward(100)
>>> t.right(90)
>>> t.forward(100)
```

▼ **그림 21-5** 거북이로 사각형 그리기



터틀 그래픽스

사각형 그리기

- 앞으로 이동: `forward`, `fd`
- 뒤로 이동: `backward`, `bk`, `back`
- 왼쪽으로 회전: `left`, `lt`
- 오른쪽으로 회전: `right`, `rt`

터틀 그래픽스

사각형 그리기

- 지금까지 그린 사각형을 fd와 rt로 만들어보면 다음과 같은 코드가 됨

```
import turtle as t

t.shape('turtle')

t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
```

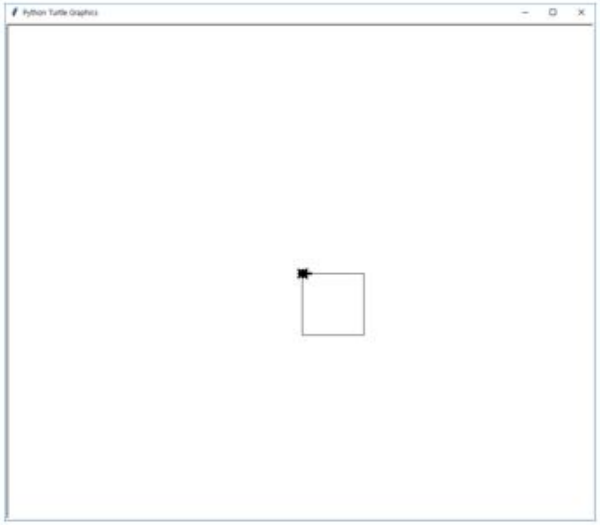
터틀 그래픽스

다각형 그리기

square.py

```
import turtle as t

t.shape('turtle')
for i in range(4):    # 사각형이므로 4번 반복
    t.forward(100)
    t.right(90)
```

▼ **그림 21-6** for 반복문으로 사각형 그리기



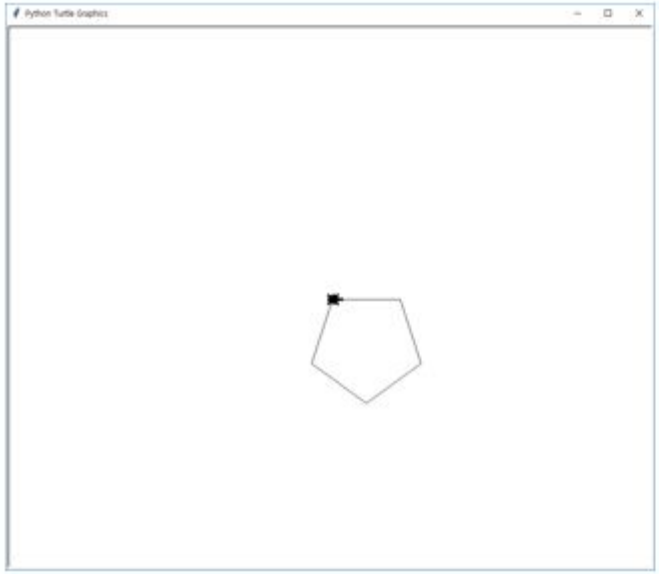
터틀 그래픽스

오각형 그리기

pentagon.py

```
import turtle as t

t.shape('turtle')
for i in range(5):       # 오각형이므로 5번 반복
    t.forward(100)
    t.right(360 / 5)     # 360을 5로 나누어서 외각을 구함
```

▼ **그림 21-7** for 반복문으로 오각형 그리기



터틀 그래픽스

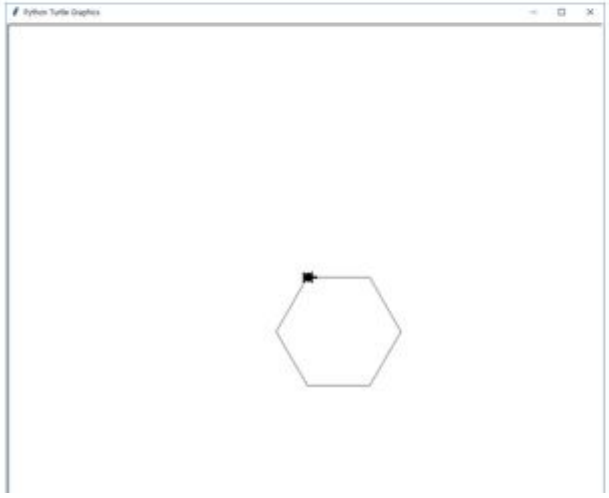
다각형 그리기

polygon.py

```
import turtle as t

n = int(input())        # 사용자의 입력을 받음
t.shape('turtle')
for i in range(n):      # n번 반복
    t.forward(100)
    t.right(360 / n)    # 360을 n으로 나누어서 외각을 구함
```

```
6 (입력)
```

▼ **그림 21-8** 입력한 숫자에 해당하는 다각형 그리기



터틀 그래픽스

다각형 그리기

- 프로그래밍은 이런 방식으로 소스 코드를 일반화해 나가는 과정임
- 공통된 부분을 일반화해서 원하는 결과를 얻어내는 과정이 프로그래밍이며 컴퓨테이셔널 씽킹임

터틀 그래픽스

다각형에 색칠하기

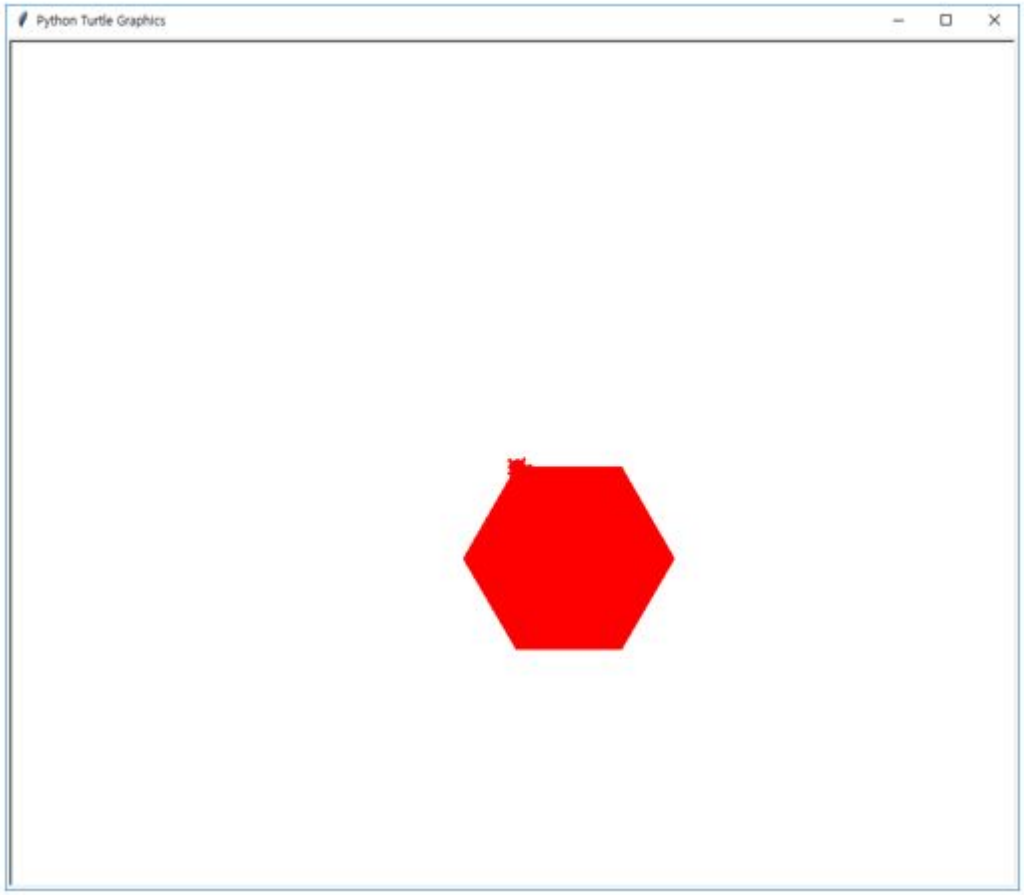
- 여기서는 숫자 입력 과정은 생략하고 n에 6을 지정하여 육각형으로 만들어보자

red_hexagon.py

```
import turtle as t

n = 6    # 육각형
t.shape('turtle')
t.color('red')          # 펜의 색을 빨간색으로 설정
t.begin_fill()          # 색칠할 영역 시작
for i in range(n):      # n번 반복
    t.forward(100)
    t.right(360 / n)    # 360을 n으로 나누어서 외각을 구함
t.end_fill()            # 색칠할 영역 끝
```

터틀 그래픽스

▼ **그림 21-9** 빨간색 육각형 그리기



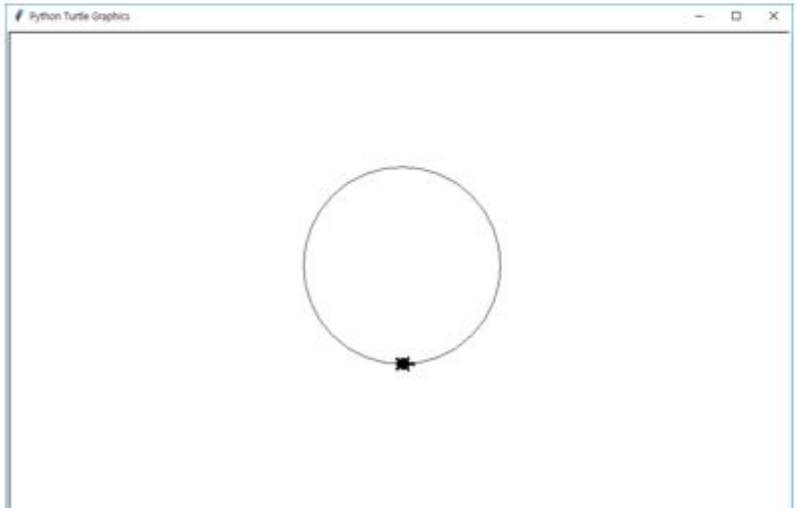
터틀 그래픽스

복잡한 도형 그리기

- 터틀에서 원을 그릴 때는 circle을 사용함

```
>>> import turtle as t
>>> t.shape('turtle')
>>> t.circle(120)
```

▼ **그림 21-10** 원 그리기



터틀 그래픽스

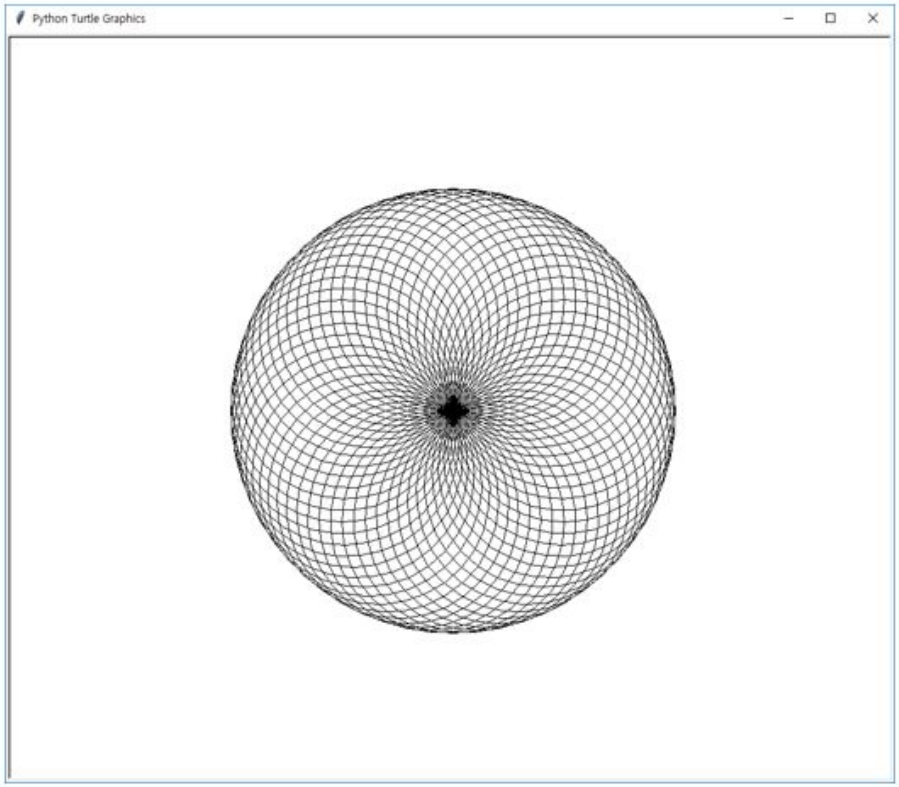
원을 반복해서 그리기

circles.py

```
import turtle as t

n = 60    # 원을 60번 그림
t.shape('turtle')
t.speed('fastest')      # 거북이 속도를 가장 빠르게 설정
for i in range(n):
    t.circle(120)       # 반지름이 120인 원을 그림
    t.right(360 / n)    # 오른쪽으로 6도 회전
```

터틀 그래픽스

▼ **그림 21-11** 원을 반복해서 그리기



터틀 그래픽스

원을 반복해서 그리기

- 'fastest': 0
- 'fast': 10
- 'normal': 6
- 'slow': 3
- 'slowest': 1

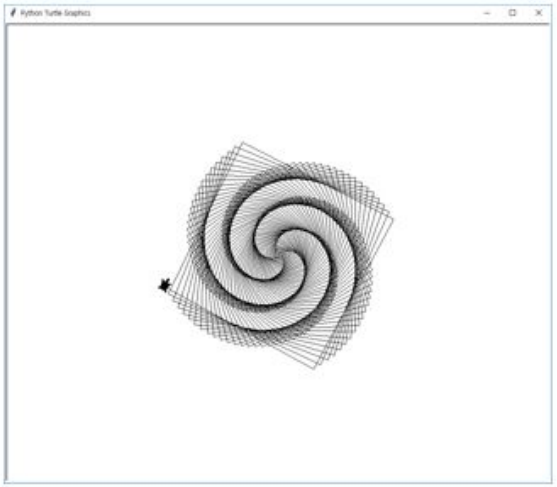
터틀 그래픽스

선으로 복잡한 무늬 그리기

vortex.py

```
import turtle as t

t.shape('turtle')
t.speed('fastest')      # 거북이 속도를 가장 빠르게 설정
for i in range(300):    # 300번 반복
    t.forward(i)        # i만큼 앞으로 이동. 반복할 때마다 선이 길어짐
    t.right(91)         # 오른쪽으로 91도 회전
```

▼ **그림 21-12** 선으로 복잡한 무늬 그리기



PyQT 활용

- PyQt는 Python 프로그래밍 언어를 위한 크로스 플랫폼 GUI(Graphical User Interface) 툴킷입니다. 이 툴킷은 Qt 라이브러리의 기능을 Python 언어로 사용할 수 있도록 하는 래퍼(wrapper) 역할을 합니다.

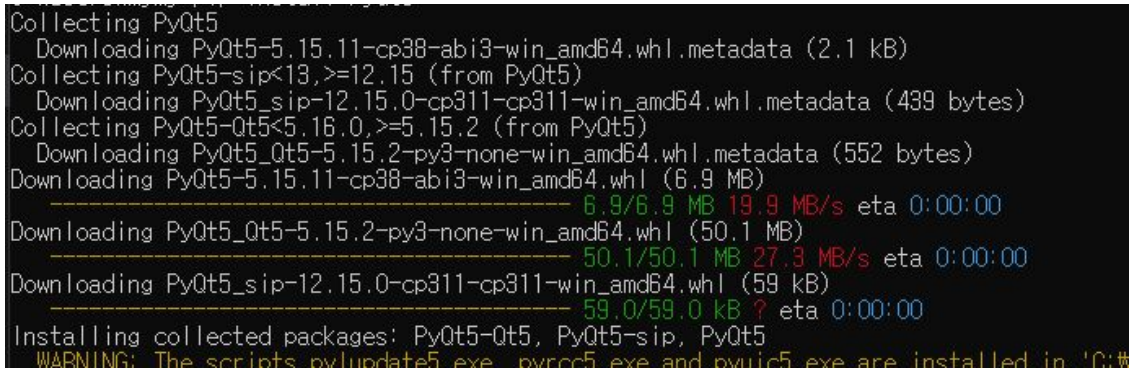
- Qt는 다양한 운영 시스템에서 동일하게 작동하는 응용 프로그램을 개발할 수 있도록 돕는 인기 있는 프레임워크입니다.

- PyQt를 사용하면, 개발자는 Qt가 제공하는 강력한 기능들을 Python 언어의 편리함과 결합하여 사용할 수 있습니다.

- 예를 들어, 윈도우, 다이얼로그, 버튼, 텍스트 박스와 같은 표준 위젯, 더 고급 위젯인 테이블 뷰, 트리 뷰 등을 사용하여 복잡한 사용자 인터페이스를 쉽게 구성할 수 있습니다.

설치

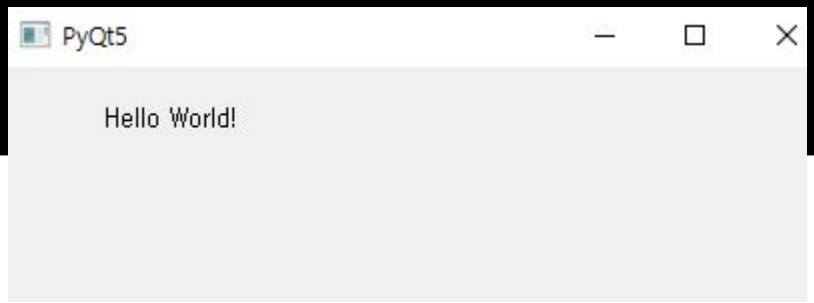
- pip install PyQt5



- pip install pyqt5-tools

Hello QT

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
   app = QApplication(sys.argv)
   w = QWidget()
   b = QLabel(w)
   b.setText("Hello World!")
   w.setGeometry(100,100,200,50)
   b.move(50,20)
   w.setWindowTitle("PyQt5")
   w.show()
   sys.exit(app.exec_())

if __name__ == '__main__':
   window()
```



PyQT

- PyQt5 API는 이전 버전과 자동으로 호환되지 않으므로 Python 코드관련

- PyQt4 모듈은 관련 변경을 통해 수동으로 업그레이드해야 합니다.

Class 로 확장하기

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
class window(QWidget):
    def __init__(self, parent = None):
        super(window, self).__init__(parent)
        self.resize(200,50)
        self.setWindowTitle("PyQt5")
        self.label = QLabel(self)
        self.label.setText("Hello World")
        font = QFont()
        font.setFamily("Arial")
        font.setPointSize(16)
        self.label.setFont(font)
        self.label.move(50,20)

def main():

   app = QApplication(sys.argv)
   ex = window()
   ex.show()
   sys.exit(app.exec_())

if __name__ == '__main__':
   main()
```

PyQt API

Sr.No.	Modules & Description
1	**QtCore** Core non-GUI classes used by other modules
2	**QtGui** Graphical user interface components
3	**QtMultimedia** Classes for low-level multimedia programming
4	**QtNetwork** Classes for network programming
5	**QtOpenGL** OpenGL support classes
6	**QtScript** Classes for evaluating Qt Scripts
7	**QtSql** Classes for database integration using SQL
8	**QtSvg** Classes for displaying the contents of SVG files
9	**QtWebKit** Classes for rendering and editing HTML
10	**QtXml** Classes for handling XML
11	**QtWidgets** Classes for creating classic desktop-style UIs.
12	**QtDesigner** Classes for extending Qt Designer

PyQt5 개발 툴
Qt 개발에 유용한 유틸리티 모음입니다

Tool Name	Description
assistant	Qt Assistant documentation tool
pyqt5designer	Qt Designer GUI layout tool
linguist	Qt Linguist translation tool
lrelease	compile ts files to qm files
pylupdate5	extract translation strings and generate or update ts files
qmake	Qt software build tool
pyqt5qmlscene	QML file viewer
pyqmlviewer	QML file viewer
pyrcc5	Qt resource file compiler
pyuic5	Qt User Interface Compiler for generating code from ui files
pyqmltestrunner	running unit tests on QML code
qdbus	command-line tool to list D-Bus services
QDoc	documentation genearator for software projects.
Qhelpgenerator	generating and viewing Qt help files.
qmlimportscanner	parses and reports on QML imports

구조

- QOobject 클래스는 클래스의 최상위 클래스입니다 모든 Qt 개체의 기본 클래스입니다.
- QPaintDevice 클래스는 칠할 수 있는 모든 개체에 대한 기본 클래스입니다.
- QAApplication 클래스는 GUI 어플리케이션의 주요 설정 및 제어 흐름을 관리하며
  창 요소와 다른 요소에 의해 생성된 이벤트가 있는 메인 이벤트 루프를 포함합니다
  소스를 처리하고 발송합니다. 또한 시스템 전체 및 애플리케이션 전체를 처리합니다
- QObject 및 QPaintDevice 클래스에서 파생된 QWidget 클래스는 모두를 위한 기본
  클래스입니다
- 사용자 인터페이스 개체. QDialog 및 QFrame 클래스도 QWidget 클래스에서 파생됩니다.
  자신 만의 하위 클래스 시스템을 가지고 있습니다.

Sr.No.	Widgets & Description
1	**QLabel** Used to display text or image
2	**QLineEdit** Allows the user to enter one line of text
3	**QTextEdit** Allows the user to enter multi-line text
4	**QPushButton** A command button to invoke action
5	**QRadioButton** Enables to choose one from multiple options
6	**QCheckBox** Enables choice of more than one options
7	**QSpinBox** Enables to increase/decrease an integer value
8	**QScrollBar** Enables to access contents of a widget beyond display aperture
9	**QSlider** Enables to change the bound value linearly.
10	**QComboBox** Provides a dropdown list of items to select from
11	**QMenuBar** Horizontal bar holding QMenu objects

12	**QStatusBar** Usually at bottom of QMainWindow, provides status information.
13	**QToolBar** Usually at top of QMainWindow or floating. Contains action buttons
14	**QListView** Provides a selectable list of items in ListMode or IconMode

15	**QPixmap** Off-screen image representation for display on QLabel or QPushButton object
16	**QDialog** Modal or modeless window which can return information to parent window

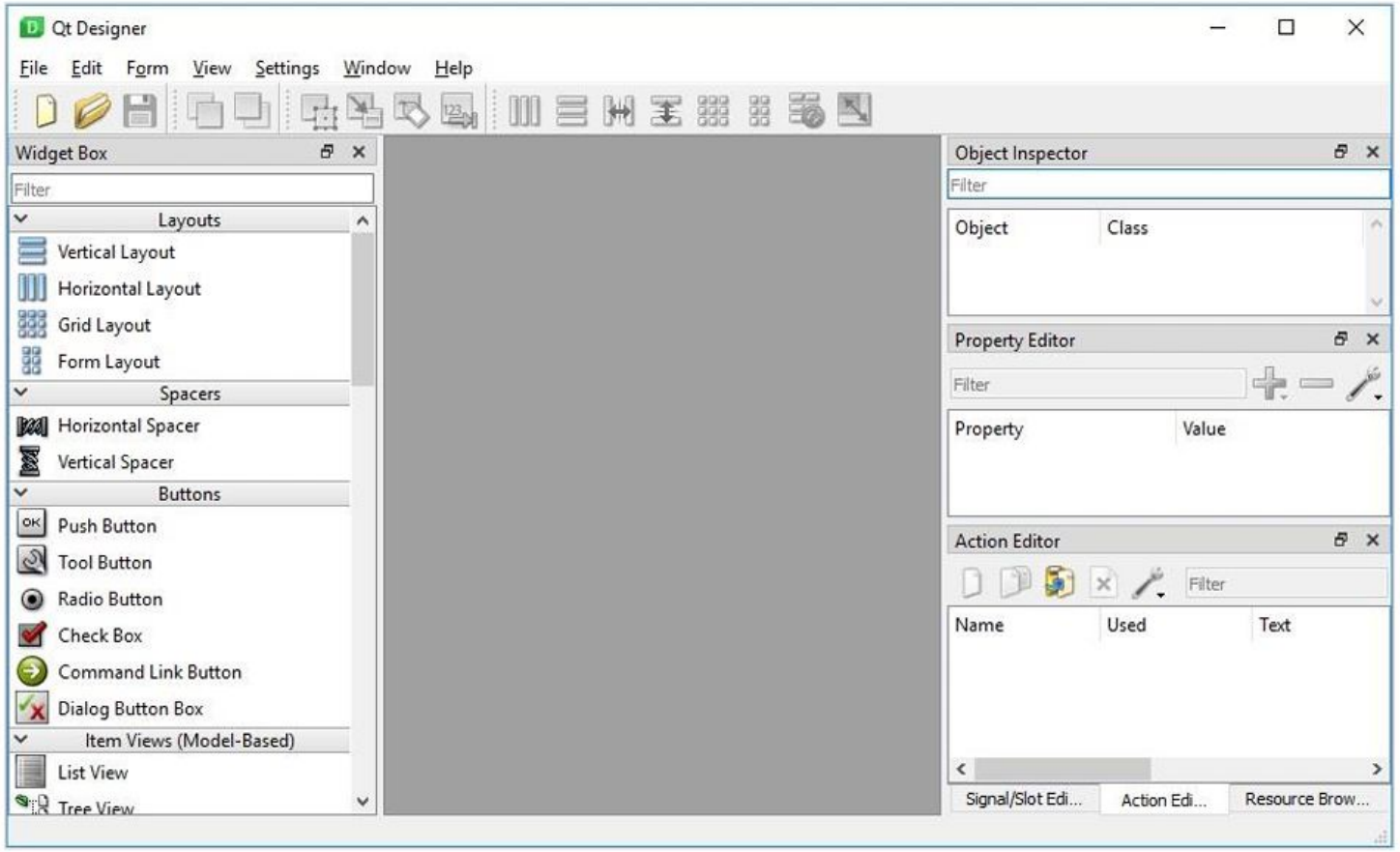
pyQT

- 일반적인 GUI 기반 응용 프로그램의 최상위 수준 창은 QMain Windows 위젯에 의해 생성됩니다
  object. 위에 나열된 일부 위젯은 이 기본 창에서 지정된 위치를 차지합니다
  다른 것들은 다양한 레이아웃 관리자를 사용하여 중앙 위젯 영역에 배치됩니다.

- 다음 다이어그램은 QMain Windows 프레임워크를 보여 줍니다:

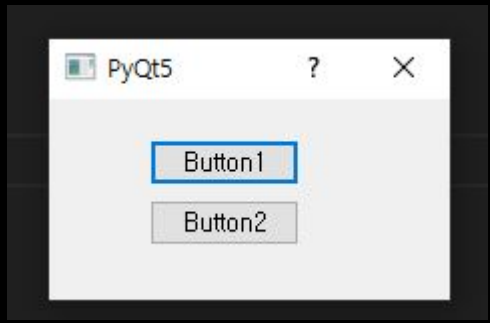
- PyQt 설치 프로그램은 Qt Designer라는 GUI 빌더 도구와 함께 제공됩니다. 간단한 드래그를 사용하여
  그리고 드롭 인터페이스, 코드를 작성할 필요 없이 GUI 인터페이스를 빠르게 구축할 수 있습니다.

- 개발 도구의 일부인 Qt Designer 응용 프로그램을 시작하고 스크립트에 설치합니다
  가상 환



- 다음 예제에서는 QDialog에 두 개의 QPushButton 개체(b1 및 b2)를 추가합니다 window. b1, b2 클릭시 함수 b1_clicked(), b2_clicked()를 호출합니다

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

def window():
   app = QApplication(sys.argv)
   win = QDialog()
   b1 = QPushButton(win)
   b1.setText("Button1")
   b1.move(50,20)
   b1.clicked.connect(b1_clicked)

   b2 = QPushButton(win)
   b2.setText("Button2")
   b2.move(50,50)
   b2.clicked.connect(b2_clicked)

   win.setGeometry(100,100,200,100)

   win.setWindowTitle("PyQt5")
   win.show()
   sys.exit(app.exec_())

def b1_clicked():
   print ("Button 1 clicked")

def b2_clicked():
   print ("Button 2 clicked")

if __name__ == '__main__':
   window()
```

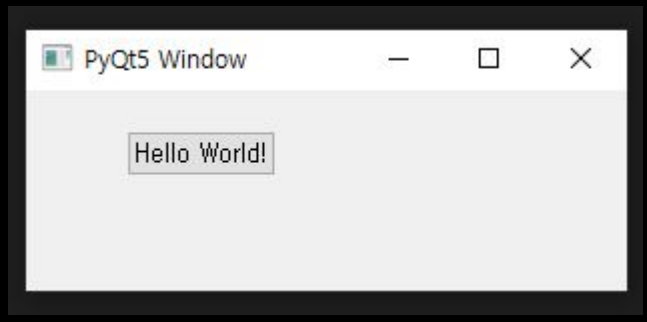


setGeometry()

- 절대값을 지정하여 컨테이너 창 내부에 GUI 위젯을 배치할 수 있습니다 픽셀 단위로 측정된 좌표입니다. 좌표는 의 치수에 상대적입니다 setGeometry() 메서드에 의해 정의된 창입니다.

- 다음 코드는, 300 x 100 픽셀 치수의 최상위 레벨 윈도우 모니터의 위치 (50,20)에 표시됩니다.

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

def window():
    app = QApplication(sys.argv)
    w = QWidget()

    b = QPushButton(w)
    b.setText("Hello World!")
    b.move(50,20)

    w.setGeometry(100,100,300,100)
    w.setWindowTitle('PyQt5 Window')
    w.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    window()
```



- 위젯이 창에 추가되어 50픽셀 위치에 배치됩니다

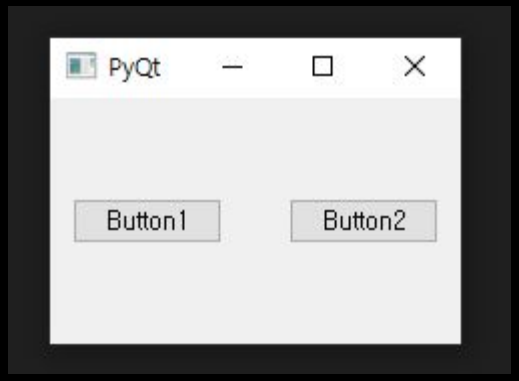
- 그러나 이 절대 포지셔닝은 다음과 같은 이유로 인해 적합하지 않습니다:
  – 윈도우의 크기가 조정되어도 위젯의 위치는 변경되지 않습니다.
  - 서로 다른 디스플레이 장치에서 외관이 균일하지 않을 수 있습니다
  – 전체 양식을 재설계해야 할 수도 있기 때문에 레이아웃을 수정하기가 어렵습니다.

- QBoxLayout 클래스는 위젯을 세로 또는 가로로 정렬합니다.
- 파생 클래스는
  - QVBoxLayout(위젯을 세로 배열)
  - QHBoxLayout(가로).

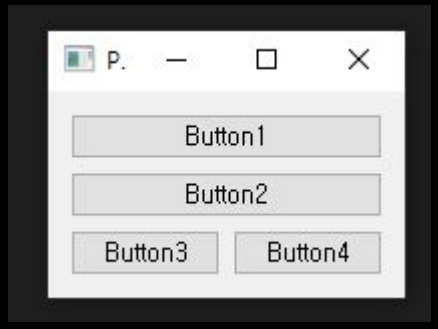
Sr.No.	Methods & Description
1	**addWidget()** Adds a widget to the BoxLayout
2	**addStretch()** Creates empty stretchable box
3	**addLayout()** Adds another nested layout

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

def window():
    app = QApplication(sys.argv)
    win = QWidget()
    b1 = QPushButton("Button1")
    b2 = QPushButton("Button2")
    vbox = QVBoxLayout()
    vbox.addWidget(b1)
    vbox.addStretch()
    vbox.addWidget(b2)
    win.setLayout(vbox)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())
if __name__ == '__main__':
    window()
```

```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
    app = QApplication(sys.argv)
    win = QWidget()
    b1 = QPushButton("Button1")
    b2 = QPushButton("Button2")
    hbox = QHBoxLayout()
    hbox.addWidget(b1)
    hbox.addStretch()
    hbox.addWidget(b2)
    win.setLayout(hbox)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())
if __name__ == '__main__':
    window()
```



```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
    app = QApplication(sys.argv)
    win = QWidget()
    b1 = QPushButton("Button1")
    b2 = QPushButton("Button2")
    vbox = QVBoxLayout()
    vbox.addWidget(b1)
    vbox.addStretch()
    vbox.addWidget(b2)
    hbox = QHBoxLayout()
    b3 = QPushButton("Button3")
    b4 = QPushButton("Button4")
    hbox.addWidget(b3)
    hbox.addStretch()
    hbox.addWidget(b4)
    vbox.addStretch()
    vbox.addLayout(hbox)
    win.setLayout(vbox)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    window()
```



- GridLayout 클래스 개체는 행과 열로 배열된 셀 그리드와 함께 나타납니다
  클래스에 addWidget() 메서드가 포함되어 있습니다. 숫자를 지정하여 위젯을 추가할 수 있습니다

- 셀의 행과 열. 선택적으로, 행과 열에 대한 스패닝 팩터를 지정하면 위젯이 하나의 셀보다 넓거나 커집니다.

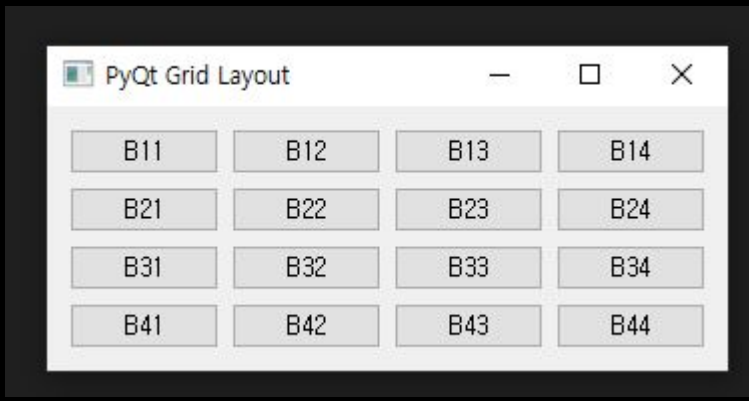
- addWidget() 활용

Sr.No.	Methods & Description
1	**addWidget(QWidget, int r, int c)** Adds a widget at specified row and column
2	**addWidget(QWidget, int r, int c, int rowspan, int columnspan)** Adds a widget at specified row and column and having specified width and/or height

A child layout object can also be added at any cell in the grid.

Sr.No.	Methods & Description
1	**addLayout(QLayout, int r, int c)** Adds a layout object at specified row and column

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

def window():
   app = QApplication(sys.argv)
   win = QWidget()
   grid = QGridLayout()

   for i in range(1,5):
      for j in range(1,5):
         grid.addWidget(QPushButton("B"+str(i)+str(j)),i,j)

   win.setLayout(grid)
   win.setGeometry(100,100,200,100)
   win.setWindowTitle("PyQt Grid Layout")
   win.show()
   sys.exit(app.exec_())

if __name__ == '__main__':
   window()
```

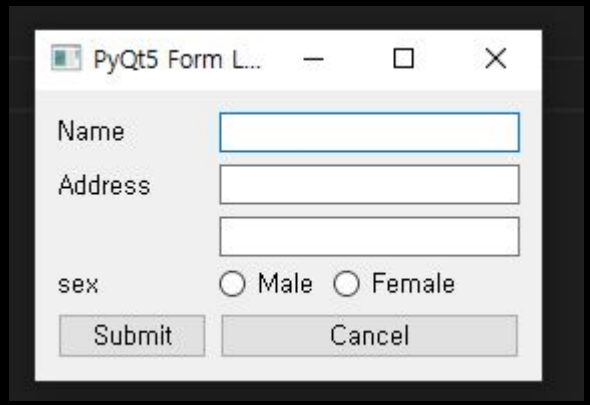


QFormLayout Class

- QFormLayout은 각 행이 구성된 두 개의 열 양식을 만드는 편리한 방법입니다
- 레이블과 연관된 입력 필드의 경우. 관례적으로 왼쪽 열은 레이블을 포함합니다
- 그리고 오른쪽 열은 입력 필드를 포함합니다.

Sr.No.	Methods & Description
1	**addRow(QLabel, QWidget)** Adds a row containing label and input field
2	**addRow(QLabel, QLayout)** Adds a child layout in the second column
3	**addRow(QWidget)** Adds a widget spanning both columns

```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
    app = QApplication(sys.argv)
    win = QWidget()
    l1 = QLabel("Name")
    nm = QLineEdit()

    l2 = QLabel("Address")
    add1 = QLineEdit()
    add2 = QLineEdit()
    fbox = QFormLayout()
    fbox.addRow(l1,nm)
    vbox = QVBoxLayout()

    vbox.addWidget(add1)
    vbox.addWidget(add2)
    fbox.addRow(l2,vbox)
    hbox = QHBoxLayout()

    r1 = QRadioButton("Male")
    r2 = QRadioButton("Female")
    hbox.addWidget(r1)
    hbox.addWidget(r2)
    hbox.addStretch()
    fbox.addRow(QLabel("sex"),hbox)
    fbox.addRow(QPushButton("Submit"),QPushButton("Cancel"))

    win.setLayout(fbox)

    win.setWindowTitle("PyQt5 Form Layout")
    win.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
   window()
```



Sr.No	Widgets & Description
1	**QLabel** A QLabel object acts as a placeholder to display non-editable text or image, or a movie of animated GIF. It can also be used as a mnemonic key for other widgets.
2	**QLineEdit** QLineEdit object is the most commonly used input field. It provides a box in which one line of text can be entered. In order to enter multi-line text, QTextEdit object is required.
3	**QPushButton** In PyQt API, the QPushButton class object presents a button which when clicked can be programmed to invoke a certain function.
4	**QRadioButton** A QRadioButton class object presents a selectable button with a text label. The user can select one of many options presented on the form. This class is derived from QAbstractButton class.
5	**QCheckBox** A rectangular box before the text label appears when a QCheckBox object is added to the parent window. Just as QRadioButton, it is also a selectable button.
6	**QComboBox** A QComboBox object presents a dropdown list of items to select from. It takes minimum screen space on the form required to display only the currently selected item.

QLabel Widget

A **QLabel** object acts as a placeholder to display non-editable text or image, or a movie of animated GIF. It can also be used as a mnemonic key for other widgets. Plain text, hyperlink or rich text can be displayed on the label.

The following table lists the important methods defined in QLabel class:

Sr.No.	Methods & Description
1	**setAlignment()** Aligns the text as per alignment constants Qt.AlignLeft Qt.AlignRight Qt.AlignCenter Qt.AlignJustify
2	**setIndent()** Sets the labels text indent
3	**setPixmap()** Displays an image
4	**Text()** Displays the caption of the label
5	**setText()** Programmatically sets the caption
6	**selectedText()** Displays the selected text from the label (The textInteractionFlag must be set to TextSelectableByMouse)
7	**setBuddy()**

```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
   app = QApplication(sys.argv)
   win = QWidget()

   l1 = QLabel()
   l2 = QLabel()
   l3 = QLabel()
   l4 = QLabel()

   l1.setText("Hello World")
   l4.setText("TutorialsPoint")
   l2.setText("welcome to Python GUI Programming")

   l1.setAlignment(Qt.AlignCenter)
   l3.setAlignment(Qt.AlignCenter)
   l4.setAlignment(Qt.AlignRight)
   l3.setPixmap(QPixmap("python.png"))

   vbox = QVBoxLayout()
   vbox.addWidget(l1)
   vbox.addStretch()
   vbox.addWidget(l2)
   vbox.addStretch()
   vbox.addWidget(l3)
   vbox.addStretch()
   vbox.addWidget(l4)
   win.setLayout(vbox)

   win.setWindowTitle("QLabel Demo")
   win.show()
   sys.exit(app.exec_())

def hovered():
   print ("hovering")
def clicked():
   print ("clicked")

if __name__ == '__main__':
   window()
```

QLineEdit

- QLineEdit object는 가장 일반적으로 사용되는 입력 필드이고 입력 상자 제공

- 여러 줄 텍스트를 입력하기 위해서는 QTextEdit 개체가 필요합니다.

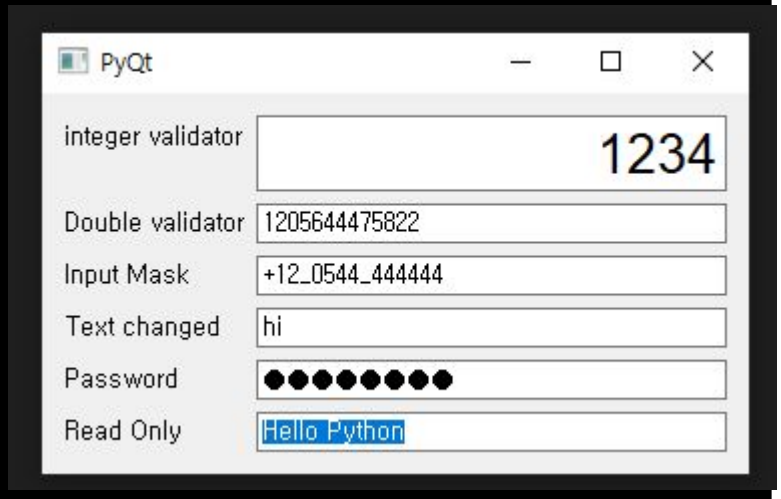
Sr.No.	Methods & Description
1	**setAlignment()** Aligns the text as per alignment constants Qt.AlignLeft Qt.AlignRight Qt.AlignCenter Qt.AlignJustify
2	**clear()** Erases the contents
3	**setEchoMode()** Controls the appearance of the text inside the box. Echomode values are − QLineEdit.Normal QLineEdit.NoEcho QLineEdit.Password QLineEdit.PasswordEchoOnEdit
4	**setMaxLength()** Sets the maximum number of characters for input
5	**setReadOnly()** Makes the text box non-editable
6	**setText()** Programmatically sets the text
7	**text()** Retrieves text in the field
8	**setValidator()** Sets the validation rules. Available validators are QIntValidator − Restricts input to integer QDoubleValidator − Fraction part of number limited to specified decimals QRegexpValidator − Checks input against a Regex expression
9	**setInputMask()** Applies mask of combination of characters for input
10	**setFont()** Displays the contents QFont object

QLineEdit

Sr.No.	Methods & Description
1	**cursorPositionChanged()** Whenever the cursor moves
2	**editingFinished()** When you press 'Enter' or the field loses focus
3	**returnPressed()** When you press 'Enter'
4	**selectionChanged()** Whenever the selected text changes
5	**textChanged()** As text in the box changes either by input or by programmatic means
6	**textEdited()** Whenever the text is edited

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
def window():
   app = QApplication(sys.argv)
   win = QWidget()

   e1 = QLineEdit()
   e1.setValidator(QIntValidator())
   e1.setMaxLength(4)
   e1.setAlignment(Qt.AlignRight)
   e1.setFont(QFont("Arial",20))

   e2 = QLineEdit()
   e2.setValidator(QDoubleValidator(0.99,99.99,2))

   flo = QFormLayout()
   flo.addRow("integer validator", e1)
   flo.addRow("Double validator",e2)

   e3 = QLineEdit()
   e3.setInputMask('+99_9999_999999')
   flo.addRow("Input Mask",e3)
```

```
 e4 = QLineEdit()
    e4.textChanged.connect(textchanged)
    flo.addRow("Text changed",e4)

    e5 = QLineEdit()
    e5.setEchoMode(QLineEdit.Password)
    flo.addRow("Password",e5)

    e6 = QLineEdit("Hello Python")
    e6.setReadOnly(True)
    flo.addRow("Read Only",e6)

    e5.editingFinished.connect(enterPress)
    win.setLayout(flo)
    win.setWindowTitle("PyQt")
    win.show()

    sys.exit(app.exec_())

def textchanged(text):
    print ("contents of text box: {}".format(text))

def enterPress():
    print ("editeing finished")

if __name__ == '__main__':
    window()
```



Ex

색상을 변경하며 별 그리기

Turtle 그래픽을 사용하여 각 꼭지점에서 색상이 변경되는 5각 별을 그리는 프로그램을 작성하세요.

```python
import turtle

def draw_star():
    screen = turtle.Screen()
    screen.bgcolor("black")
    star = turtle.Turtle()
    star.speed(1)
    colors = ['red', 'yellow', 'blue', 'green', 'purple']

    for i in range(5):
        star.color(colors[i])
        star.forward(100)
        star.right(144)

    star.hideturtle()
    screen.mainloop()

# 함수 호출
draw_star()
```

Ex

무지개 색 원 그리기

Turtle 그래픽을 사용하여 무지개 색상의 원을 겹쳐 그리는 프로그램을 작성하세요.

```python
import turtle

def draw_rainbow_circles():
    screen = turtle.Screen()
    screen.bgcolor("black")
    rainbow = turtle.Turtle()
    rainbow.speed(0)
    colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
    radius = 50

    for color in colors:
        rainbow.color(color)
        rainbow.circle(radius)
        radius += 10  # 원의 반지름을 점점 늘려서 그리기

    rainbow.hideturtle()
    screen.mainloop()


# 함수 호출
draw_rainbow_circles()
```