1. 기초(Basics)

(1) 창 띄우기

```
#창 띄우기

import sys
from PyQt5.QtWidgets import  QApplication, QWidget

class MyApp(QWidget):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
self.setWindowTitle('My First App')
self.move(300, 300)
self.resize(400, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- self는 MyApp의 객체, setWindowTitle()은 타이틀바에 나타나는 창의 제목 설정
- move()는 위젯을 스크린의 x=300px, y=300px의 위치로 이동
- resize()는 위젯의 크기를 너비 400px, 높이 200px로 조절
- show() 메소드는 위젯을 스크린에 보여줌
- __name__은 현재 모듈의 이름이 저장되는 내장 변수, 프로그램이 직접 실행되는지 혹은 모듈을 통해 실행되는지를 확인

(2) 어플리케이션 아이콘 넣기

```
#어플리케이션 아이콘 넣기

import sys
from PyQt5.QtWidgets import  QApplication, QWidget
from PyQt5.QtGui import QIcon

class MyApp(QWidget):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
self.setWindowTitle('Icon')
self.setWindowIcon(QIcon('이미지이름.확장자'))
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- setWindowIcon()은 어플리케이션 아이콘을 설정하도록 함.
- QIcon()에 보여질 이미지를 입력함.
- setGeometry()는 창의 위치와 크기를 설정함. 앞 두 매개변수는 창의 x, y 위치를 결정하고, 뒤의 두 매개변수는 각각 창의 너비와 높이를 결정함. move()와 resize()를 하나로 합쳐놓은 것과 같음.

(3) 창 닫기

```
#창 닫기

import sys
from PyQt5.QtWidgets import  QApplication, QWidget, QPushButton
from PyQt5.QtCore import QCoreApplication

class MyApp(QWidget):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
btn=QPushButton('Quit', self)
btn.move(50, 50)
btn.resize(btn.sizeHine())
btn.clicked.connect(QCoreApplication.instance().quit)

self.setWindowTitle('Quit Button')
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

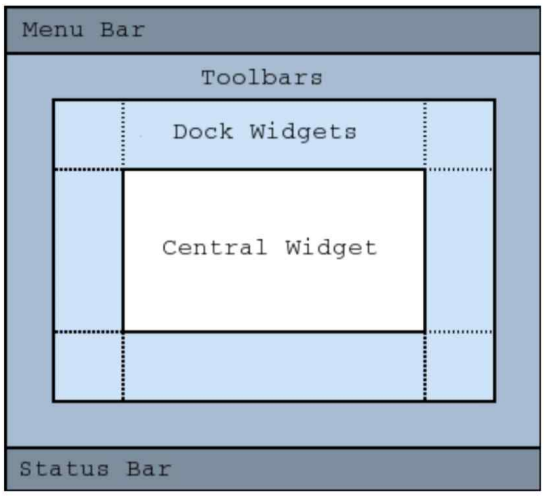
- btn은 QPushButton 클래스의 인스턴스임.
- QPushButton()의 첫번째 파라미터에는 버튼이 표시될 텍스트를 입력하고, 두번째 파라미터에는 버튼이 위치할 부모 위젯을 입력함.
- PyQt5에서의 이벤트 처리는 시그널과 슬롯 메커니즘으로 이루어짐.
- btn(버튼)을 클릭하면 'clicked' 시그널이 만들어짐
- instance()는 현재 인스턴스를 반환함
- 'clicked' 시그널은 어플리케이션을 종료하는 quit()에 연결됨

(4) 툴팁 나타내기(툴팁: 어떤 위젯의 기능을 설명하는 말풍선 형태의 도움말)

```
#툴팁 나타내기

import sys
from PyQt5.QtWidgets import  QApplication, QWidget, QPushButton
from PyQt5.QtGui import QFont

class MyApp(QWidget):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
QToolTip.setFont(QFont('SansSerif', 10))
self.setToolTip('This is a <b>QWidget</b> widget')

btn=QPushButton('Button', self)
btn.setToolTip('This is a <b>QPushButton</b> widget')
btn.move(50, 50)
btn.resize(btn.sizeHint())

self.setWindowTitle('Tooltips')
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- 여기서는 10px 크기의 'SansSerif' 폰트를 사용
- 툴팁을 만들기 위해서는 setToolTip()을 사용해서, 표시될 텍스트를 입력해줌
- sizeHint()는 버튼을 적정한 크기로 설정하도록 도와줌

(5) 상태바 만들기



< 메인창의 레이아웃

```
#상태바 만들기

import sys
from PyQt5.QtWidgets import  QApplication, QMainWindow
class MyApp(QMainWindow):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
self.statusBar().showMessage('Ready')

self.setWindowTitle('Statusbar')
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- 상태바는 QMainWindow 클래스의 statusBar()를 이용해 만드는데, 이 메소드를 최초로 호출함으로써 만들어짐
- 그 다음 호출부터는 상태바 객체를 반환함
- showMessage()를 통해 상태바에 보여질 메세지를 설정할 수 있음
- 텍스트가 사라지게 하고 싶으면, clearMessage()를 이용하거나, showMessage()에 텍스트가 표시되는 시간을 설정할 수 있음

(6) 메뉴바 만들기

```
#메뉴바 만들기

import sys
from PyQt5.QtWidgets import  QApplication, QMainWindow, QAction, qApp
from PyQt5.QtGui import QIcon

class MyApp(QMainWindow):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
exitAction=QAction(QIcon('이미지이름.확장자'), 'Exit', self)
exitAction.setShortcut('Ctrl+Q')
exitAction.setStatusTip('Exit application')
exitAction.triggered.connect(qApp.quit)

self.statusBar()

menubar=self.menuBar()
menubar.setNativeMenuBar(False)
filemenu=menubar.addMenu('&File')
filemenu.addAction(exitAction)

self.setWindowTitle('Menubar')
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- 아이콘(QIcon)과 'Exit' 라벨을 갖는 하나의 동작(action)을 만들고, 이 동작에 대해 단축키(shortcut)를 정의함
- 메뉴에 마우스를 올렸을 때, 상태바에 나타날 상태팁을 setStatusTip()을 사용해 설정
- qApp.quit에 관한 동작을 선택했을 때, 생성된 시그널(triggered)이 QApplication 위젯의 quit()에 연결되고, 어플을 종료함
- menuBar()은 메뉴바를 생성함. 이어서 File 메뉴를 하나 만들고, 거기에 exitAction 동작을 추가함
- &File의 &은 간편하게 단축키를 설정하도록 해줌, F 앞에 &가 있으므로, Alt+F가 File 메뉴의 단축키가 됨

(7) 툴바 만들기

```
#툴바 만들기

import sys
from PyQt5.QtWidgets import  QApplication, QMainWindow, QAction, qApp
from PyQt5.QtGui import QIcon
class MyApp(QMainWindow):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
exitAction=QAction(QIcon('이미지이름.확장자'), 'Exit', self)
exitAction.setShortcut('Ctrl+Q')
exitAction.setStatusTip('Exit application')
exitAction.triggered.connect(qApp.quit)

self.statusBar()

self.toolbar=self.addToolBar('Exit')
self.toolbar.addAction(exitAction)

self.setWindowTitle('Toolbar')
self.setGeometry(300, 300, 300, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- 메뉴바와 같이 QAction 객체를 생성함. 이 객체는 아이콘(이미지), 라벨('Exit')을 포함하고, 단축기(Ctrl+Q)를 통해 실행 가능함
- 상태바에 메세지('Exit application')를 보여주고, 클릭시 생성되는 시그널은 quit()에 연결되어 있음
- addToolbar()를 이용해 툴바를 만들고, addAction()을 이용해 툴바에 exitAction 동작을추가함

(8) 창을 화면 가운데로 띄우기

```
#창을 화면 가운데로 띄우기

import sys
from PyQt5.QtWidgets import  QApplication, QWidget, QDesktopWidget

class MyApp(QWidget):
def __init__(self):
super().__init__()
self.initUI()

def initUI(self):
self.setWindowTitle('Centering')
self.resize(500, 350)
self.center()
self.show()

def center(self):
qr=self.frameGeometry()
cp=QDesktopWidget().availableGeometry().center()
qr.moveCenter(cp)
self.move(qr.topLeft())

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- center()을 통해 창이 화면 가운데에 위치하게 됨
- frameGeometry()를 이용해 창의 위치와 크기 정보를 가져옴
- cp=~은 사용하는 모니터의 화면 가운데 위치를 파악함
- qr.movecenter()은 창의 직사각형 위치를 화면의 중심 위치로 이동함
- self.move(qr.topLeft())는 현재 창을 화면의 중심으로 이동했던 직사각형(qr)의 위치로 이동시킴

(9) 날짜와 시간 표시하기

① 현재 날짜 출력하기

```
from PyQt5.QtCore import QDate

now=QDate.currentDate()
print(now.toString())
```

- currentDate()는 현재 날짜를 반환함
- toString()을 통해 현재 날짜를 문자열로 출력할 수 있음

② 날짜 형식 설정하기

```
from PyQt5.QtCore import QDate, Qt

now=QDate.currentDate()
print(now.toString('d.M.yy'))
print(now.toString('dd.MM.yyyy'))
print(now.toString('ddd.MMMM.yyyy'))
print(now.toString(Qt.ISODate))
print(now.toString(Qt.DefaultLocaleLongDate))
```

- d(일), M(월), y(년)은 각 문자의 개수에 따라 날짜의 형식이 다르게 출력됨
- Qt.ISODate, Qt.DefalutLocaleLongDate를 입력함으로써 ISO 표준 형식 또는 어플리케이션의 기본 설정에 맞게 출력할 수 있음

Expression	Output
d	the day as number without a leading zero (1 to 31)
dd	the day as number with a leading zero (01 to 31)
ddd	the abbreviated localized day name (e.g. 'Mon' to 'Sun'). Uses the system locale to localize the name, i.e. QLocale::system().
dddd	the long localized day name (e.g. 'Monday' to 'Sunday'). Uses the system locale to localize the name, i.e. QLocale::system().
M	the month as number without a leading zero (1 to 12)
MM	the month as number with a leading zero (01 to 12)
MMM	the abbreviated localized month name (e.g. 'Jan' to 'Dec'). Uses the system locale to localize the name, i.e. QLocale::system().
MMMM	the long localized month name (e.g. 'January' to 'December'). Uses the system locale to localize the name, i.e. QLocale::system().
yy	the year as two digit number (00 to 99)
yyyy	the year as four digit number. If the year is negative, a minus sign is prepended in addition.

Constant	Value	Description
Qt::TextDate	0	The default Qt format, which includes the day and month name, the day number in the month, and the year in full. The day and month names will be short, localized names. This is basically equivalent to using the date format string, "ddd MMM d yyyy". See QDate::toString() for more information.
Qt::ISODate	1	ISO 8601 extended format: either yyyy-MM-dd for dates or yyyy-MM-ddTHH:mm:ss (e.g. 2017-07-24T15:46:29), or with a time-zone suffix (Z for UTC otherwise an offset as [+|-]HH:mm) where appropriate for combined dates and times.
Qt::ISODateWithMs	9	ISO 8601 extended format, including milliseconds if applicable.
Qt::SystemLocaleShortDate	4	The short format used by the operating system.
Qt::SystemLocaleLongDate	5	The long format used by the operating system.
Qt::DefaultLocaleShortDate	6	The short format specified by the application's locale.
Qt::DefaultLocaleLongDate	7	The long format used by the application's locale.
Qt::SystemLocaleDate	2	*This enum value is deprecated.* Use Qt::SystemLocaleShortDate instead (or Qt::SystemLocaleLongDate if you want long dates).
Qt::LocaleDate	3	*This enum value is deprecated.* Use Qt::DefaultLocaleShortDate instead (or Qt::DefaultLocaleLongDate if you want long dates).
Qt::LocalDate	SystemLocaleDate	*This enum value is deprecated.* Use Qt::SystemLocaleShortDate instead (or Qt::SystemLocaleLongDate if you want long dates).
Qt::RFC2822Date	8	RFC 2822, RFC 850 and RFC 1036 format: either [ddd,] dd MMM yyyy hh:mm[:ss] +/-TZ or ddd MMM dd yyyy hh:mm[:ss] +/-TZ for combined dates and times.

③ 현재 시간 출력하기

```
from PyQt5.QtCore import QTime

time=QTime.currentTime()
print(time.toString())
```

- currentTime()은 현재 시간을 반환함
- toString()은 현재 시간을 문자열로 반환함

④ 시간 형식 설정하기

```
from PyQt5.QtCore import QTime, Qt

time=QTime.currentTime()

print(time.toString('h.m.s'))
print(time.toString('hh.mm.ss'))
print(time.toString('hh.mm.ss.zzz'))
print(time.toString(Qt.DefaultLocaleLongDate))
print(time.toString(Qt.DefaultLocaleShortDate))
```

- 'z'는 1000분의 1초를 나타냄

Expression	Output
h	the hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display)
hh	the hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display)
H	the hour without a leading zero (0 to 23, even with AM/PM display)
HH	the hour with a leading zero (00 to 23, even with AM/PM display)
m	the minute without a leading zero (0 to 59)
mm	the minute with a leading zero (00 to 59)
s	the whole second, without any leading zero (0 to 59)
ss	the whole second, with a leading zero where applicable (00 to 59)
z	the fractional part of the second, to go after a decimal point, without trailing zeroes (0 to 999). Thus "s.z" reports the seconds to full available (millisecond) precision without trailing zeroes.
zzz	the fractional part of the second, to millisecond precision, including trailing zeroes where applicable (000 to 999).
AP or A	use AM/PM display. *A/AP* will be replaced by either QLocale::amText() or QLocale::pmText().
ap or a	use am/pm display. *a/ap* will be replaced by a lower-case version of QLocale::amText() or QLocale::pmText().
t	the timezone (for example "CEST")

⑤ 현재 날짜와 시간 출력하기

```
from PyQt5.QtCore import QDateTime

datetime=QDateTime.currentDateTime()
print(datetime.toString())
```

- currentDateTime()은 현재의 날짜와 시간을 반환함
- toString()은 날짜와 시간을 문자열 형태로 반환함

⑥ 날짜와 시간 형식 설정하기

```
from PyQt5.QtCore import QDateTime, Qt

datetime=QDateTime.currentDateTime()
print(datetime.toString('d.M.yy hh:mm:ss'))
print(datetime.toString('dd.MM.yyyy, hh:mm:ss'))
print(datetime.toString(Qt.DefaultLocaeLongDate))
print(datetime.toString(Qt.DefaultLocaeShortDate))
```

⑦ 상태표시줄에 날짜 표시하기

```
#날짜와 시간 표시하기

import sys
from PyQt5.QtWidgets import  QApplication, QMainWindow
from PyQt5.QtCore import QDate, Qt

class MyApp(QMainWindow):
def __init__(self):
super().__init__()
self.date=QDate.currentDate()
self.initUI()

def initUI(self):
self.statusBar().showMessage(self.date.toString(Qt.DefaultLocaleLongDate))

self.setWindowTitle('Date')
self.setGeometry(300, 300, 400, 200)
self.show()

if __name__=='__main__':
app=QApplication(sys.argv)
ex=MyApp()
sys.exit(app.exec_())
```

- currentDate()를 통해 현재 날짜를 얻고, showMessage()로 상태표시줄에 현재 날짜를 표시함