هدف دوره آشنایی با مفاهیم کاربردی پایتون و ایجاد برنامه های کارآمد و کاربردی

آشنایی با مفاهیم مدیریت استثنا و کنترل خطاها

آشنایی با سوکت پروگرمینگ

آشنایی با ماژول رکوئست و عبارات با قاعده

آشنایی با پردازش موازی

ثبت وقایع

طراحی برنامه گرافیکی حسابداری و انبارداری و و ارتباط با دیتابیس و شبکه محلی کارمندان و سطوح دسترسی متفاوت

همچنین در این دوره آشنایی با مفاهیم برنامه نویسی شی گرا ، کاربا دیتابیس و کار با گیت از الزامات دوره میباشد

قدم اول ساخت ریپازیتوری جدید در گیت و ساخت ماشین مجازی جهت ماژول های مورد نیاز دوره و دلیل نیاز به ماشین مجازی (داستان خانم سرور )

.env و فایل Python-dotenv , Pyinstaller , pyqt 5, mysql-connector-python و نصب ماژول های Requirements.txt ساخت فایل

برای هر جلسه پروژه گیت هاب را میسازیم تا بدانیم چه چیزی قرار است درس بدهیم

و ساخت کلاس بیسیکQT شروع به کار با ماژول

استفاده از ماژول های بیسیک

from PyQt5.QtWidgets import QWidget, QApplication

import sys

import os

تعریف کلاسی که میتواند کانستراکتور داشته باشد و میتواند نداشته باشد . میتواند متود نمایش را در اجرای کد داشته باشد و یا در خود کد

همچنین تعیین ابعاد یک برنامه بسیار مهم در کیفیت اجرای برنامه است

عموما چون در کانستراکتور مقدار دهی جدیدی نیست میتوان کانستراکتور را حذف کرد و متود ساخت فرم را صدا زد . همچنین متود نمایش را میتوان در بدنه کد اجرایی یا بدنه تابع نوشت

میتوان آیکون برنامه Os میتوان فولدر فایل اجرایی را پیدا کرد و بدین ترتیب با قرار دادن یک آیکون در کنار فایل اجرایی و استفاده از Os با استفاده از ماژول

را نیز بصورت داینامیک تغییر داد

from PyQt5.QtWidgets import QWidget, QApplication

from PyQt5.QtGui import QIcon

import os

import sys

class Window(QWidget):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setUI()

    def setUI(self):

        # -----------> Basic Window

        self.setWindowTitle('Test Window')

        self.setGeometry(300, 300, 300, 300)

        self.setMinimumSize(300, 300)

        self.setMaximumSize(300, 300)

        self.setWindowIcon(QIcon("icon.ico"))

        self.setStyleSheet("background-color: red;")

        #self.setFixedSize(300, 300)

        #------------> QLable

        self.show()

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    obj1 = Window()

    sys.exit(app.exec\_())

میتوان نسبت به تغییر رنگ یک آبجکت اقدام کرد paint همچنین با تابع استایل شیت و کمک گرفتن از ابزار کالر باکس در نرم افزاری مانند

همچنین میتوان رنگ های اصلی را با نام به صورت مستقیم استفاده کرد

مد نظر باشد تابع نمایش همیشه باید در خط آخر باشد

تعریف لیبل ها و دسترسی به مقادیر داخلی یک لیبل و ایجاد یک دکمه و اتصال به تابع و اعمال تغییرات روی تابع

تعریف اکتیو و دی اکتیو کردن یک آبجکت

from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QPushButton

from PyQt5.QtGui import QIcon, QFont

from button import Button

import os

import sys

*class* Window(*QWidget*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_()

        self.setUI()

*def* setUI(*self*):

        # -----------> Basic Window

        self.setWindowTitle('Test Window')

        self.setGeometry(300, 300, 600, 600)

        self.setMinimumSize(300, 300)

        self.setMaximumSize(300, 300)

        self.setWindowIcon(QIcon("icon.ico"))

        #self.setStyleSheet("background-color: red;")

        #self.setFixedSize(300, 300)

        #------------> QLable

        self.lable1 = QLabel("hamid",self)

        self.lable1.move(10,10)

        self.lable1.setStyleSheet("color: red;")

        #------------> Pushbutton

        self.btn1 = QPushButton("Click Me", self)

        self.btn1.move(100, 100)

        self.btn1.resize(50,50)

        self.btn1.clicked.connect(self.clicked)

        #--------------------------------

        self.btn2 = QPushButton("Click Me2", self)

        self.btn2.move(200,200)

        self.btn2.resize(50,50)

        #--------------------------------

        self.show()

*def* clicked(*self*):

        self.lable1.setFont(QFont("Times New Roman", 20))

        self.lable1.setText("Im changed")

        self.lable1.setStyleSheet("color:blue;")

        self.lable1.resize(130,80)

        self.btn2.setDisabled(True)

        print(self.lable1.text())

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    obj1 = Window()

    sys.exit(app.exec\_())

تعاریف دکمه و لاین ادیت و تکست ادیت و کمبوباکس و رادیو باتن و چک باکس

from PyQt5.QtWidgets import (QWidget, QApplication, QLabel, QPushButton,

                            QLineEdit, QTextEdit, QCheckBox, QRadioButton,

                            QComboBox)

from PyQt5.QtGui import QIcon, QFont

import os

import sys

*class* Window(*QWidget*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_()

        self.setUI()

*def* setUI(*self*):

        # -----------> Basic Window

        self.setWindowTitle('Test Window')

        self.setGeometry(300, 300, 600, 600)

        self.setMinimumSize(300, 300)

        self.setMaximumSize(300, 300)

        self.setWindowIcon(QIcon("icon.ico"))

        #self.setStyleSheet("background-color: red;")

        #self.setFixedSize(300, 300)

        #------------> QLable

        self.lable1 = QLabel("hamid",self)

        self.lable1.move(10,10)

        self.lable1.setStyleSheet("color: red;")

        #------------> Pushbutton

        self.btn1 = QPushButton("Click Me", self)

        self.btn1.move(100, 100)

        self.btn1.resize(50,50)

        self.btn1.clicked.connect(self.clicked)

        #--------------------------------

        self.btn2 = QPushButton("Click Me2", self)

        self.btn2.move(200,200)

        self.btn2.resize(50,50)

        self.btn2.clicked.connect(self.clicked2)

        #------------> Lineedit

        self.lineedit1 = QLineEdit(self)

        self.lineedit1.move(150, 100)

        self.lineedit1.textChanged.connect(self.text)

        #------------> Textedit

        self.textedit1 = QTextEdit(self)

        self.textedit1.move(250, 100)

        self.textedit1.textChanged.connect(self.text)

        #------------> Checkbox

        self.checkbox1 = QCheckBox("Check Me", self)

        self.checkbox1.move(100, 250)

        self.checkbox1.stateChanged.connect(self.check)

        #------------> Radiobutton

        self.radiobutton1 = QRadioButton("Radio Me", self)

        self.radiobutton1.move(50, 100)

        self.radiobutton1.clicked.connect(self.radio)

        #------------> Combobox

        self.combobox1 = QComboBox(self)

        self.combobox1.move(100, 100)

        #self.combobox1.addItem("Hamid")

        self.combobox1.addItems(["hamid", "reza", "mehrabi"])

        #self.combobox1.currentIndexChanged.connect(self.combo)

        self.combobox1.currentTextChanged.connect(self.combo)

        self.show()

*def* clicked(*self*):

        self.lable1.setFont(QFont("Times New Roman", 20))

        self.lable1.setText("Im changed")

        self.lable1.setStyleSheet("color:blue;")

        self.lable1.resize(130,80)

        #self.btn2.setDisabled(True)

        self.lineedit1.setDisabled(False)

        #self.lineedit1.setEnabled(True)

        print(self.lable1.text())

*def* clicked2(*self*):

        self.lineedit1.setText("im here")

        self.lineedit1.setDisabled(True)

*def* text(*self*):

        print(self.textedit1.toPlainText())

*def* check(*self*):

        print(self.checkbox1.isChecked())

*def* radio(*self*):

        print(self.radiobutton1.isChecked())

*def* combo(*self*,*name*):

        print(self.combobox1.currentText())

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    obj1 = Window()

    sys.exit(app.exec\_())

و استفاده از آن و ایجاد فرم لاگین برای یک ماشین حساب گرافیکی تعریف فرم لاگین و معرفی متغیر های محیطی

pyinstaller --noconsole --icon=icon.ico --add-data="icon.ico;." login.py

تعریف مسیج باکس جهت کنترل ارور ها و یا ایجاد ارور های دلخواه

انواع دکمه های استاندارد و سلف کلوز و نمایش بصورت دیفالت برای مسیج باکس

from PyQt5.QtWidgets import  QMessageBox, QWidget

*class* Message(*QWidget*):

*def* Error(*self*):

        #msg = QMessageBox()

        #msg.setIcon(QMessageBox.Critical)

        #msg.setWindowTitle("Warning")

        #msg.setText("username or password is incorrect")

        #msg.setStandardButtons(QMessageBox.Ok)

        #msg.exec\_()

        QMessageBox.critical(self, "Error", "username or password is incorrect", QMessageBox.Ok)

from PyQt5.QtWidgets import (QWidget, QApplication, QLabel, QPushButton,

                            QLineEdit, QTextEdit, QCheckBox, QRadioButton,

                            QComboBox, QMessageBox)

from PyQt5.QtGui import QIcon, QFont

import training

import message

import os

import sys

*class* Window(*QWidget*):

*def* \_\_init\_\_(*self*):

*super*().\_\_init\_\_()

        self.setUI()

*def* setUI(*self*):

        # -----------> Basic Window

        self.setWindowTitle('Login form')

        self.setGeometry(300, 300, 300, 100)

        self.setFixedSize(300,100)

        self.setWindowIcon(QIcon("icon.ico"))

        #------------> QLable

        self.lable1 = QLabel("username",self)

        self.lable1.move(10,10)

        self.lable2 = QLabel("password",self)

        self.lable2.move(10,40)

        #------------> Lineedit

        self.line1 = QLineEdit(self)

        self.line1.resize(230,20)

        self.line1.move(60,10)

        self.line2 = QLineEdit(self)

        self.line2.resize(230,20)

        self.line2.move(60,40)

        self.line2.setEchoMode(QLineEdit.Password)

        #------------> Button

        self.btn1 = QPushButton("Login", self)

        self.btn1.move(215,70)

        self.btn1.clicked.connect(self.login)

        self.btn2 = QPushButton("Cancel", self)

        self.btn2.move(60,70)

        self.show()

*def* login(*self*):

        #if self.line1.text() == "admin" and self.line2.text() == "admin":

        #    self.close()

        #    train.show()

        if self.line1.text() == os.getenv("username") and self.line2.text() == os.getenv("pwd"):

            self.close()

            train.show()

        else :

            message.Error()

تعریف ماشین حساب گرافیکی

تعریف گوگل ترنسلیت گرافیکی