

## پروژه Password Generator App

شاگرد : علیرضا اردانی

منتور : امیرحسین قربان حسینی

تابستان ۱۴۰۴

## نیازمندی های پروژه

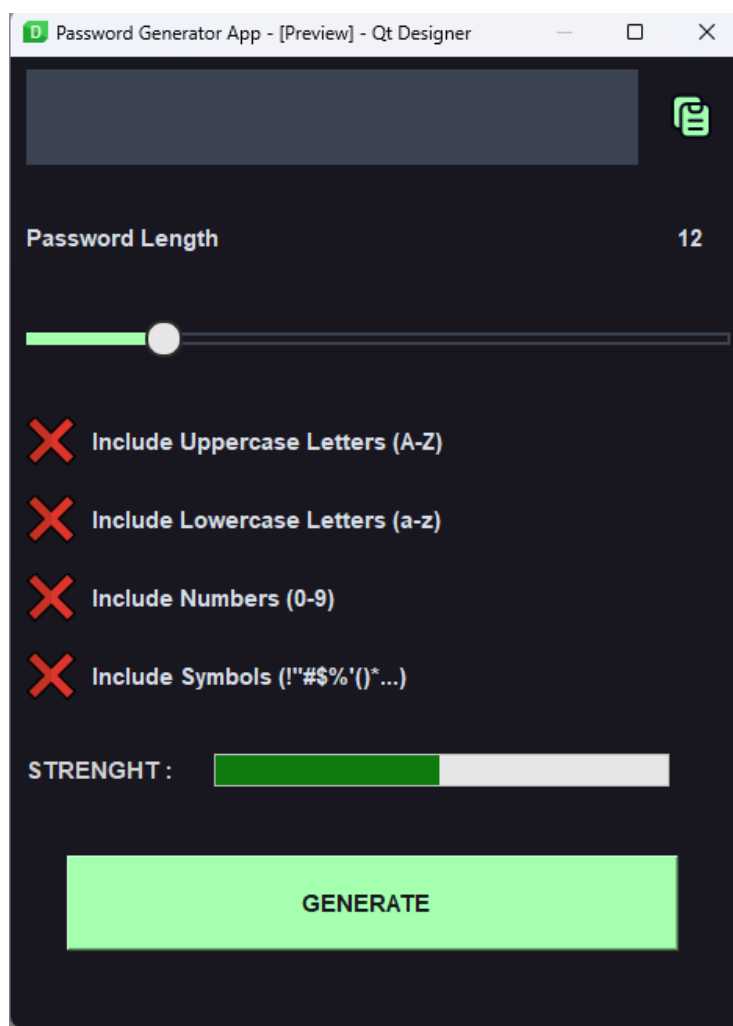
برای اجرای برنامه، نیاز به نصب PyQt5 می باشد. همچنین برای استفاده از Qt Designer باید آن را نصب کنیم.

دستورات نصب

```
pip install PyQt5
pip install PyQt5Designer
```

در این پروژه از کتابخانه secrets، کتابخانه string، کتابخانه sys و کتابخانه PyQt5 استفاده شده است.

خروجی نهایی این پروژه در تصویر زیر نشان داده شده است.

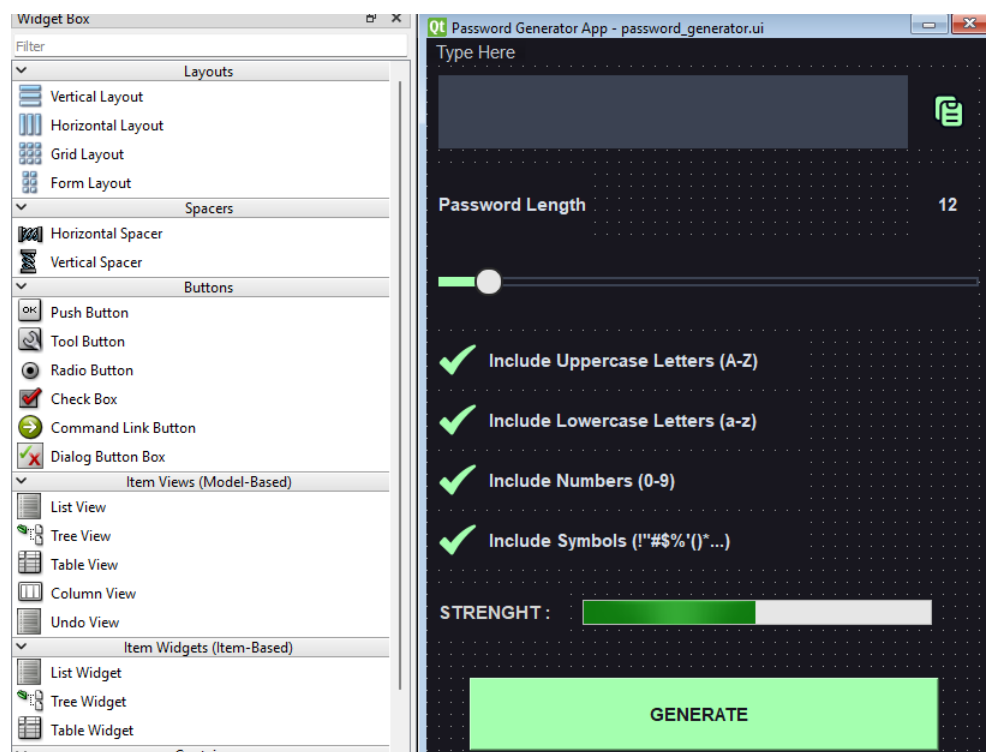


در این پروژه، کاربر ابتدا طول رمز را مشخص نموده و سپس گزینه های مد نظر خود را جهت استفاده در تولید رمز انتخاب می کند. در نهایت با کلیک بر روی دکمه GENERATE، رمز تولید می شود. کاربر می تواند با زدن دکمه کناری باکسی که در آن رمز نمایش داده شده است، آن را کپی کند.

## طراحی رابط کاربری

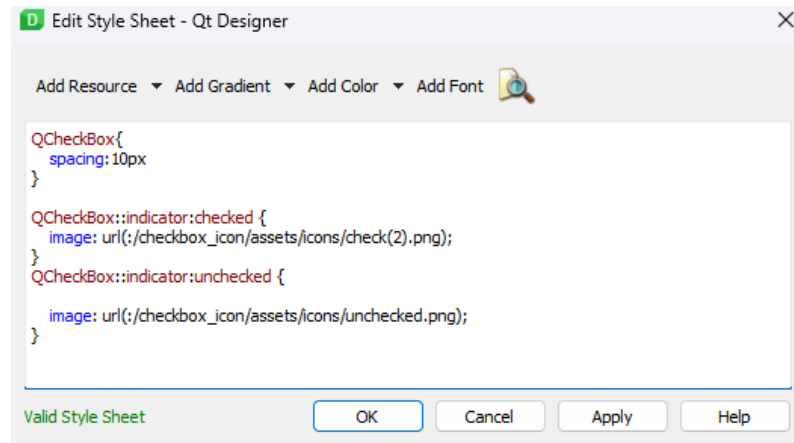
با استفاده از برنامه Qt Designer، و به کمک ویجت های موجود در آن و با استایل دهی به آن ها طراحی پروژه انجام شده است.

. ویجت های برنامه



. استایل دهی : با راست کلیک بر روی ویجت مد نظر و انتخاب `changeStyleSheet` می توانیم

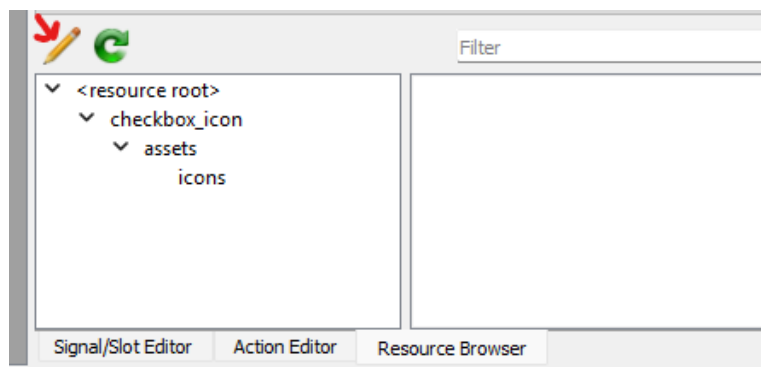
استایل مدنظر خود را اعمال کنیم.

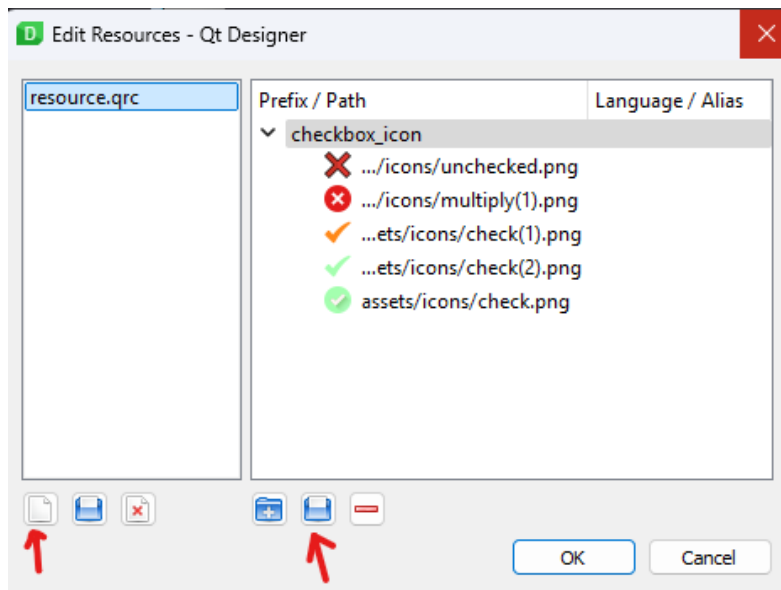


. اضافه کردن resource به رابط کاربری : در منوی Resource Browser با ایجاد resource

جدید، ابتدا باید resource با پسوند qrc ایجاد کنیم و سپس فایل های مد نظر خود را به آن اضافه

کنیم.





## پیاده سازی پروژه با زبان برنامه نویسی پایتون

۱- ابتدا باید تمام نیازها اعم از ویجت ها مورد استفاده، کتابخانه ها، فایل مربوط به `resource` را `import` کنیم.

```
1 from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QLineEdit, QLabel, QSlider, QCheckBox, QProgressBar
2 from PyQt5 import uic
3 import sys
4 import string
5 from secrets import choice, SystemRandom
6 import resources
7 from PyQt5.QtCore import QTimer
```

در این پروژه از ویجت های QApplication, QMainWindow, QPushButton

QLineEdit(readonly), QLabel, QSlider, QCheckBox, QProgressBar استفاده شده است.

برای لود کردن فایل طراحی از uic، برای اجرای برنامه از sys، برای دسترسی به انواع کاراکتر ها قابل استفاده

در رمز از string، در فرایند تولید رمز از secrets، برای شناساندن resource به پایتون، از فایل تبدیل

شده به پایتون resource.py و همچنین از QTimer برای تایمر استفاده شده است.

۲- کلاس PasswordGenerator، کلاس اصلی برنامه ما است که از QMainWindow ارث بری کرده و

تمام ویجت ها و عملکرد آنها در این کلاس تعریف شده اند.

ابتدا متد \_\_init\_\_ آن را پیاده سازی می کنیم و self را به عنوان پارامتر، پاس می دهیم. درون این متد، متد

\_\_init\_\_ والد را صدا زده و کلاس و self را به آن پاس می دهیم تا دسترسی به ویژگی ها را داشته باشد.

سپس حروف کوچک، حروف بزرگ، اعداد و نماد های مختلف را تعریف می کنیم و سپس تابع initUi را صدا

می زنیم.

```

1  def __init__(self):
2      super(PasswordGenerator, self).__init__()
3      '''
4      --- Password Characters Option ---
5      uppercase : ABCDEFGHIJKLMNOPQRSTUVWXYZ
6      lowercase : abcdefghijklmnopqrstuvwxyz
7      digits : 0123456789
8      symbols : !"#%&'()*+,-./:;<=>?@[\\]^_`{|}~
9      '''
10     self.UPPERCASE = string.ascii_uppercase
11     self.LOWERCASE = string.ascii_lowercase
12     self.SYMBOLS = string.punctuation
13     self.NUMBERS = string.digits
14
15     self.initUi()
16

```

### ۳- تابع initUi

داخل این تابع ابتدا، فایل طراحی را لود می‌کنیم و سپس کلیه ویجت‌ها خود را تعریف کرده و در انتها سیگنال‌های clicked دکمه‌ها و valueChanged اسلایدر را به توابع مدنظر متصل می‌کنیم.




```
1 uic.loadUi('password_generator.ui', self)
```



```
1 # Define and Modify Widgets
2 self.passwordOutput = self.findChild(QLineEdit, 'generatedPasswordLineEdit')
3 self.copyToClipboardButton = self.findChild(QPushButton, 'copyPushButton')
4 self.passwordLengthLabel = self.findChild(QLabel, 'passwordLengthLabel')
5 self.sliderValueLabel = self.findChild(QLabel, 'sliderValueLabel')
6 self.passwordLengthSlider = self.findChild(QSlider, 'passwordLengthSlider')
7 # Check Boxes
8 self.uppercaseCheckBox = self.findChild(QCheckBox, 'uppercaseCheckBox')
9 self.lowercaseCheckBox = self.findChild(QCheckBox, 'lowercaseCheckBox')
10 self.numbersCheckBox = self.findChild(QCheckBox, 'numbersCheckBox')
11 self.symbolCheckBox = self.findChild(QCheckBox, 'symbolCheckBox')
12
13 # Strength
14 self.strengthLabel = self.findChild(QLabel, 'passwordStrengthLabel')
15 self.strengthProgressBar = self.findChild(QProgressBar, 'passwordStrengthProgressBar')
16 self.strengthProgressBar.setRange(0, 100)
17 self.strengthProgressBar.setValue(0)
18 # Generate Password Button
19 self.generateButton = self.findChild(QPushButton, 'generatePushButton')
20
21
```






```

1 # connect signals to buttons
2 self.copyToClipboardButton.clicked.connect(self.copy_to_clipboard)
3 self.passwordLengthSlider.valueChanged.connect(self.update_slider)
4 self.generateButton.clicked.connect(self.generate_password)
5

```

#### ۴- تابع `update_slider`

این تابع، مقدار اسلایدر را براساس تعامل کاربر بر روی لیبل مقدار آن نمایش می‌دهد.



```

1 def update_slider(self, slider_value):
2     self.sliderValueLabel.setText(str(slider_value))
3

```

#### ۵- تابع `copy_to_clipboard`

این تابع جهت پیاده سازی عمل کپی کردن رمز ایجاد شده است. در این تابع جهت راه اندازی `clipboard`، ابتدا تابع `clipboard` را از `QApplication` صدا می‌زنیم، سپس متن داخل ویجت `passwordOutput` را به عنوان متن `clipboard` قرار می‌دهیم. سپس برای مطلع کردن کاربر از کپی شدن رمز، می‌توانیم دو روند را پیش بگیریم

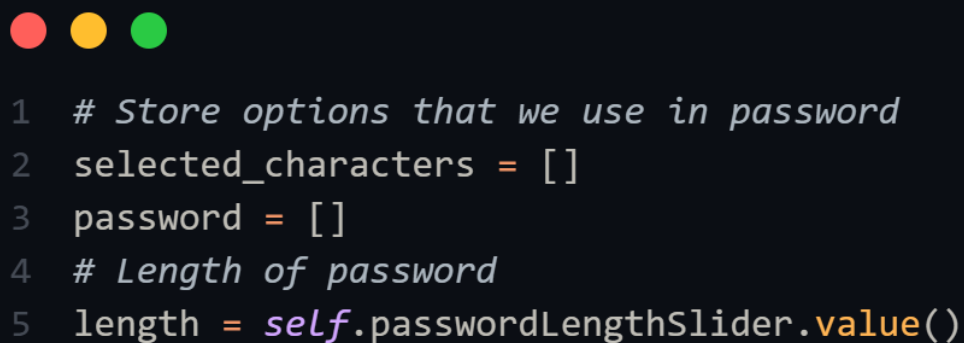
۱: استفاده از `QMessageBox` - ۲: تغییر متن داخل `passwordOutput` برای لحظاتی به `Copied!`

در این پروژه به دلیل این که QMessageBox در روند تعامل کاربر با برنامه اختلال ایجاد می‌کند، از روش دوم استفاده می‌کنیم. بدین ترتیب ابتدا رمز تولید شده را در متغیر org\_text نگه داری می‌کنیم، سپس استایل متن را تغییر داده و رنگ متن را سبز کرده و متن آن را به "Copied!" تغییر می‌دهیم. سپس با استفاده از QTimer، تایمر راه اندازی می‌کنیم و به مدت ۱.۵ ثانیه این متن را نگه داری کرده و سپس آن را به متن اصلی که رمز تولید شده است، تغییر می‌دهیم و همچنین استایل آن را به استایل قبلی برمی‌گردانیم. تابع singleShot، ابتدا تاخیری ایجاد می‌کند و سپس تابعی برای انجام کاری بعد از تاخیر به آن پاس می‌دهیم.

```
1
2 def copy_to_clipboard(self):
3     # Copy Password
4     clipboard = QApplication.clipboard()
5     clipboard.setText(self.passwordOutput.text())
6     # Show When Copied
7     # QMessageBox.about(self, 'Copy To Clipboard', 'Password Copied!')
8     # print('Password Copied! ---- {} ----'.format(clipboard.text()))
9     org_text = self.passwordOutput.text()
10    self.passwordOutput.setStyleSheet(
11        "background-color:#3B4252;color:green;border-color:#3B4252;")
12    self.passwordOutput.setText("Copied!")
13
14    QTimer.singleShot(1500, lambda: (
15        self.passwordOutput.setText(org_text),
16        self.passwordOutput.setStyleSheet(
17            "background-color:#3B4252;color:#E6E6E6;border-color:#3B4252;")
18    ))
19
```

## ۶ – تابع generate\_password

در این تابع لیست `selected_characters` جهت نگه داری همه کاراکتر های انتخابی کاربر و لیست `password` جهت نگهداری بخش های مختلف رمز که به صورت شانسی انتخاب شده اند، تعریف شده است. طول رمز با توجه مقدار اسلایدر، به متغیر `length` داده می شود.



```
1 # Store options that we use in password
2 selected_characters = []
3 password = []
4 # Length of password
5 length = self.passwordLengthSlider.value()
```

حال داخل بلاک `try..except` روند تولید رمز پیگیری می شود، در صورتی که هر یک از چک باکس ها، چک شده باشند، ابتدا به لیست تبدیل شده و سپس به `selected_characters` اضافه می شوند و سپس در صورتی که چک باکسی انتخاب شده باشد، یک آیتم از لیست کاراکتر مرتبط با چک باکس به صورت شانسی به کمک تابع `choice` از کتابخانه `secrets` انتخاب می شود. دلیل انتخاب کتابخانه `secrets` به جای `random`، قابل پیشبینی بودن کتابخانه `random` در عملیات هایی مانند `shuffle` و `choice` می باشد. تابع `choice` به صورت تصادفی یک آیتم را از یک لیست انتخاب می کند.

```

1 # Check which options will be included in password, make sure to have at least one of these options characters
2 if self.uppercaseCheckBox.isChecked():
3     selected_characters.extend(list(self.UPPERCASE))
4     password.append(choice(list(self.UPPERCASE)))
5
6 if self.lowercaseCheckBox.isChecked():
7     selected_characters.extend(list(self.LOWERCASE))
8     password.append(choice(list(self.LOWERCASE)))
9
10 if self.numbersCheckBox.isChecked():
11     selected_characters.extend(list(self.NUMBERS))
12     password.append(choice(list(self.NUMBERS)))
13
14 if self.symbolCheckBox.isChecked():
15     selected_characters.extend(list(self.SYMBOLS))
16     password.append(choice(list(self.SYMBOLS)))
17 print(password)

```

در ادامه در صورتی که selected\_characters خالی باشد، درون ویجت passwordOutput، به کاربر پیام “Please select at least one character set.” داده می‌شود و مقدار progressbar نیز ۰ در نظر گرفته می‌شود.

```

1 # Check which options will be included in password, make sure to have at least one of these options characters
2 if self.uppercaseCheckBox.isChecked():
3     selected_characters.extend(list(self.UPPERCASE))
4     password.append(choice(list(self.UPPERCASE)))
5
6 if self.lowercaseCheckBox.isChecked():
7     selected_characters.extend(list(self.LOWERCASE))
8     password.append(choice(list(self.LOWERCASE)))
9
10 if self.numbersCheckBox.isChecked():
11     selected_characters.extend(list(self.NUMBERS))
12     password.append(choice(list(self.NUMBERS)))
13
14 if self.symbolCheckBox.isChecked():
15     selected_characters.extend(list(self.SYMBOLS))
16     password.append(choice(list(self.SYMBOLS)))
17 print(password)

```

در صورتی که طول رمزی که توسط کاربر تعیین شده از طول لیست همه کارکترهایی که کاربر برای تولید رمز انتخاب کرده بیشتر باشد، این پیام نمایش داده می‌شود. "Password is longer than characters set length." و در غیر صورت وارد روند تولید رمز می‌شود.

```
1 if length > len(selected_characters):
2     self.passwordOutput.setText("Password is longer than characters set length.")
3
4 else:
```

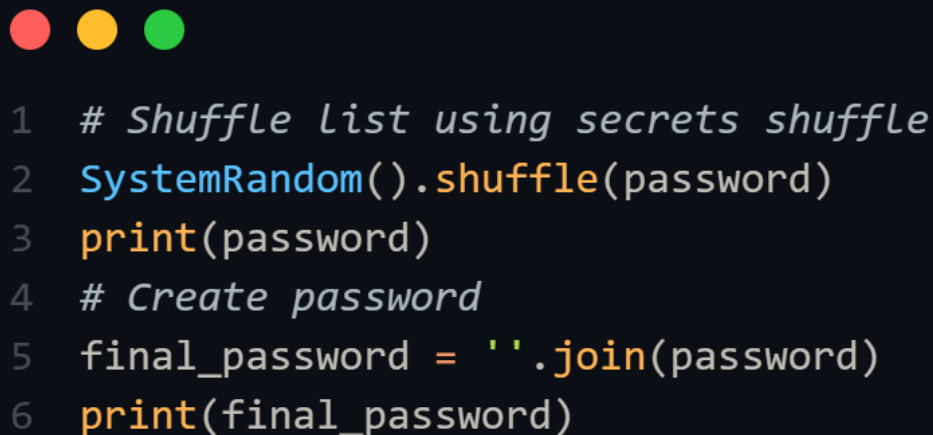
در روند تولید رمز، در زمانی که چک باکس ها را چک می‌کردیم، اگر چک باکسی انتخاب شده بود از لیست کاراکترهای مرتبط به آن، آیتمی به صورت تصادفی انتخاب می‌شد و اکنون باید باقی رمز را تولید کنیم. پس طول باقیمانده از رمز را باید در متغیر remaining length نگهداری کنیم. سپس درون یک حلقه for، باقی رمز را به کمک تابع choice انتخاب می‌کنیم.

```
1 remaining_length = length - len(password)
2     # use secrets.choice() instead in Loop
3     # temp = sample(selected_characters, remaining_length)
4     for _ in range(remaining_length):
5         password.append(choice(selected_characters))
6
7     print(password)
```

حال لیست رمز ما با توجه به طول انتخاب شده توسط کاربر، تکمیل شده است. با استفاده SystemRandom از کتابخانه secrets و صدا زدن تابع shuffle آن، این لیست را در هم ریخته می‌کنیم. و در نهایت به کمک

تابع `join`، آیتم های داخل لیست `password` را بافضای خالی بهم الصاق می کنیم.

بدین صورت : `[1,2,3,4]-->"1234"`



```
1 # Shuffle List using secrets shuffle
2 SystemRandom().shuffle(password)
3 print(password)
4 # Create password
5 final_password = ''.join(password)
6 print(final_password)
```

در پایان متن ویجت `passwordOutput` را به رمز تولید شده تغییر می دهیم و تابع

`update_password_strength` را جهت تغییر میزان قوی بودن رمز بر روی `progressbar` صدا


می زنیم.

۷- تابع `update_password_strength`

در این تابع بر اساس تعداد گزینه های که کاربر برای حضور در تولید رمز آن را انتخاب کرده و طول رمز به قوی


بودن رمز، امتیازی اختصاص می دهیم.

متغیر `used_char` برای نگهداری تعداد مجموعه کاراکترهایی که کاربر انتخاب کرده، `score` برای امتیازدهی رمز تولید شده و `length` برای نگهداری طول رمز، تعریف شده اند.



```
1 used_char = 0
2 score = 0
3 length = len(self.passwordOutput.text())
```

در صورتی طول رمز بزرگتر از ۸، ۱۶ و ۳۰ کاراکتر باشد، ۲۰ امتیاز به رمز اضافه می شود.



```
1 # score base on Length
2 if length >= 8:
3     score += 20
4 if length >= 16:
5     score += 20
6 if length >= 30:
7     score += 20
```

بر اساس تعداد لیست کاراکترهای انتخاب شده،

- در صورتی رمز شامل ۲ لیست از کاراکترهای موجود باشد، ۲۰ امتیاز
- در صورتی رمز شامل ۳ لیست از کاراکترهای موجود باشد، ۳۰ امتیاز
- در صورتی رمز شامل ۴ لیست از کاراکترهای موجود باشد، ۴۰ امتیاز

به رمز اضافه می‌شود.

```
1  # score if any of characters used
2  if self.uppercaseCheckBox.isChecked():
3      used_char += 1
4  if self.lowercaseCheckBox.isChecked():
5      used_char += 1
6  if self.numbersCheckBox.isChecked():
7      used_char += 1
8  if self.symbolCheckBox.isChecked():
9      used_char += 1
10 # score base on used characters
11 if used_char == 2:
12     score += 20
13 if used_char == 3:
14     score += 30
15 if used_char == 4:
16     score += 40
17 print(score)
```



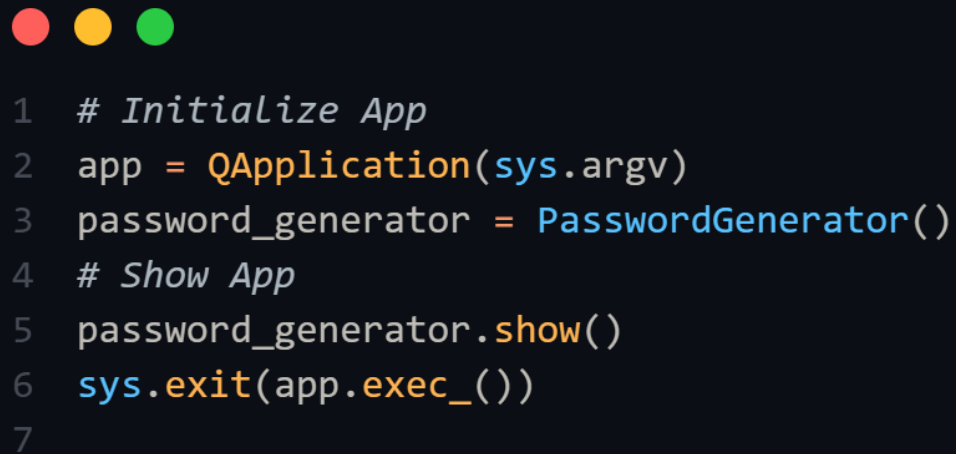
در نهایت رمز مقدار امتیاز به عنوان مقدار progressBar در نظر گرفته می‌شود و براساس میزان قوی بودن آن، با رنگ های مختلفی نمایش داده می‌شود.

```
if score <= 40:
    color = "■"■"#E3CF10"
elif score <= 60:
    color = "■"■"#E36110"
elif 75 <= score < 90:
    color = "■"■"#10E337"
elif score >= 90:
    color = "■"■"#1E5E1A"
```

```
1 self.strengthProgressBar.setValue(score)
2
3 if score <= 40:
4     color = "#E3CF10"
5 elif score <= 60:
6     color = "#E36110"
7 elif 75 <= score < 90:
8     color = "#10E337"
9 elif score >= 90:
10    color = "#1E5E1A"
11
12 self.strengthProgressBar.setStyleSheet(f"QProgressBar::chunk{{ background-color:{color}; }}")
13
14
```

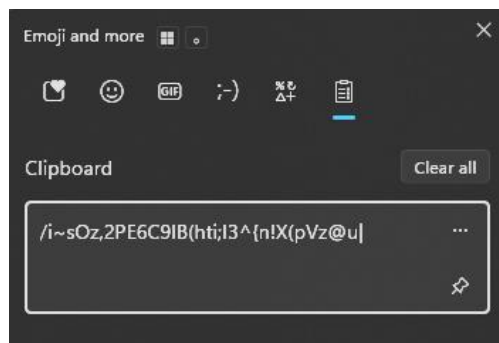
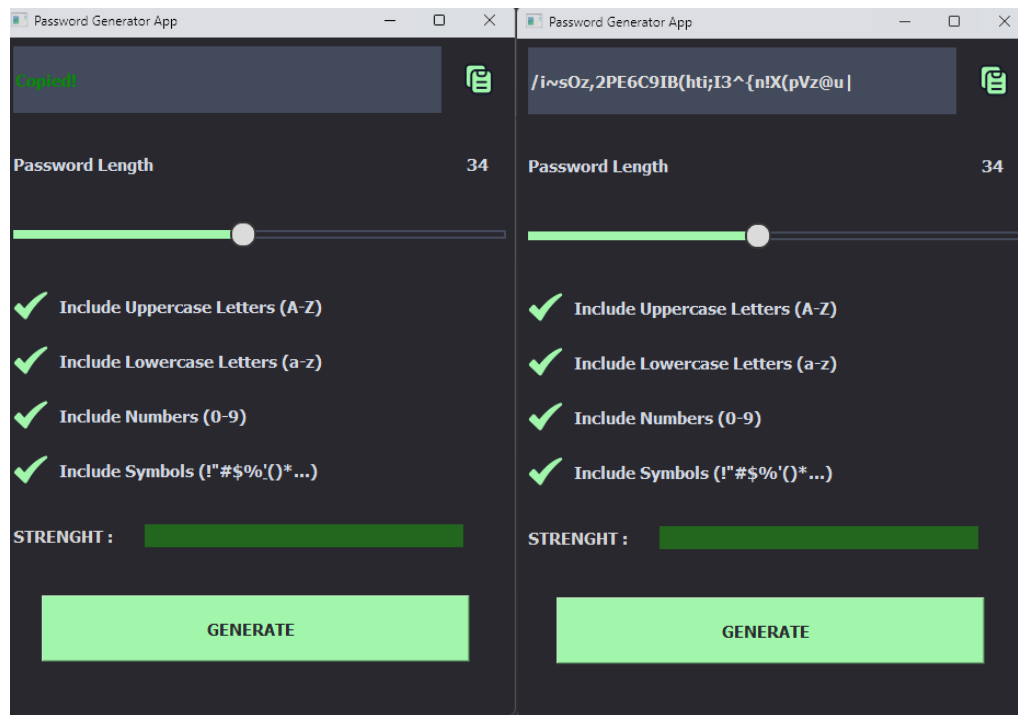
## ۸- اجرای برنامه

ابتدا نمونه از اپ و کلاس ساخته و سپس آن را نمایش می‌دهیم و در نهایت اجرا می‌کنیم.



```
1 # Initialize App
2 app = QApplication(sys.argv)
3 password_generator = PasswordGenerator()
4 # Show App
5 password_generator.show()
6 sys.exit(app.exec_())
7
```

## ۹- اجرای برنامه در حالت های مختلف



### Take the Password Test

**Tip:** It's often better to have longer passwords than shorter, more complex ones

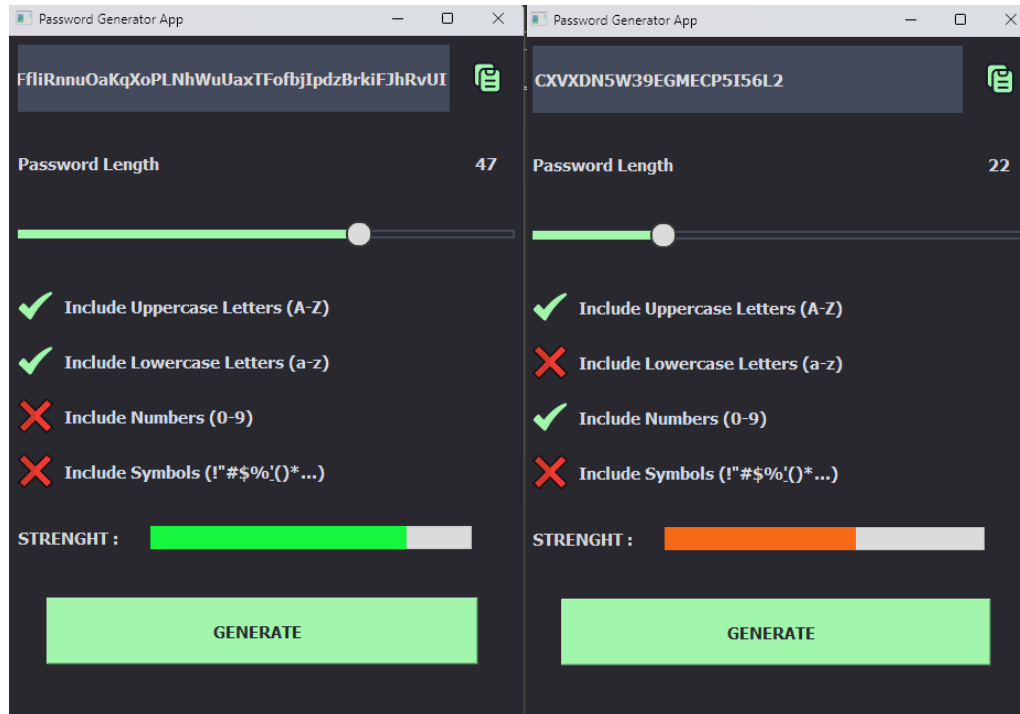
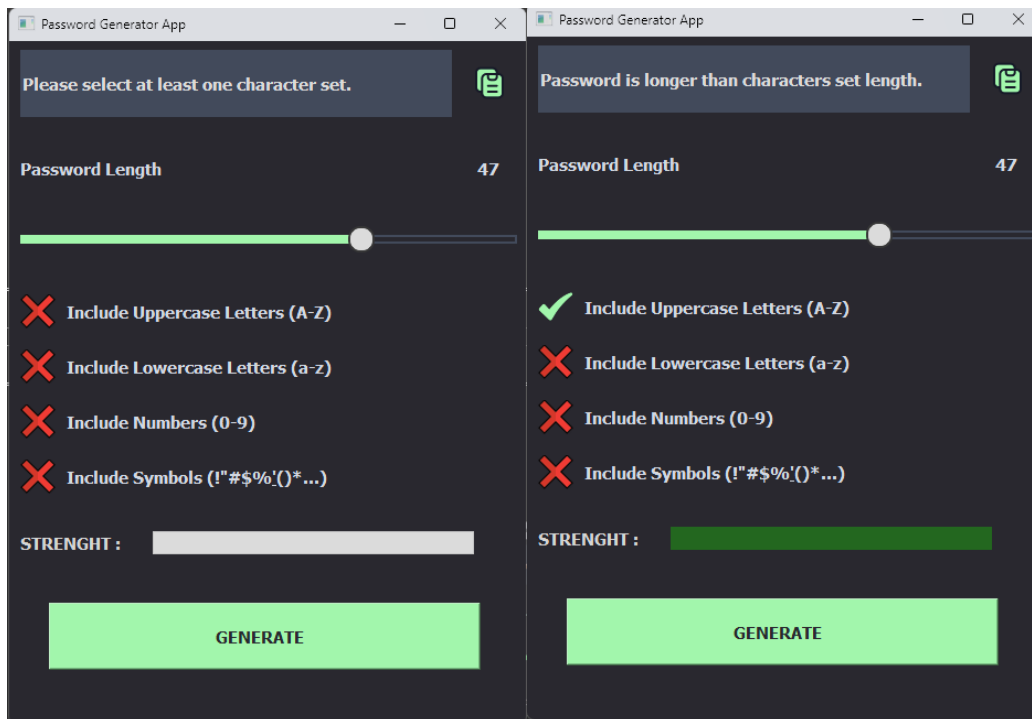
Show password: ☒

`/i~sOz,2PE6C9IB(hti;l3^{n!X(pVz@u|`

**Very Strong**

34 characters containing: Lower case Upper case Numbers Symbols

Time to crack your password:  
**3 thousand trillion trillion trillion years**



Password Generator App

!R}?GM"+

Password Length

8

✓

Include Uppercase Letters (A-Z)

✗

Include Lowercase Letters (a-z)

✗

Include Numbers (0-9)

✓

Include Symbols (!"#\$%&'()\*+,-./:;<=>?@[\]^\_`{|}~)

STRENGTH :

GENERATE