

**T.C.**

**İSTANBUL AYDIN ÜNİVERSİTESİ**

**UYGULAMALI BİLİMLER YÜKSEKOKULU**

**YÖNETİM BİLİŞİM SİSTEMLERİ**

**Lisans Tezi**

**Eğlence İçerikli Oyunu**

Bilal Örgen

Kerem Tiryaki

Haziran, 2024



**T.C.**

**İSTANBUL AYDIN ÜNİVERSİTESİ**

**UYGULAMALI BİLİMLER YÜKSEKOKULU**

**YÖNETİM BİLİŞİM SİSTEMLERİ**

**Eğlence İçerikli Oyunu**

Lisans Tezi

Bitirme Projesi

Bilal Örgen B2180.010084

Kerem Tiryaki B2080.010065

DANIŞMAN ÖĞR. GÖR. HAKAN BURAK EMEKLİ

# Onay Sayfası

İstanbul Aydın Üniversitesi Yönetim Bilişim Sistemleri B2180.010084 ve B2080.010065 numaralı Lisans Öğrencileri Bilal Örgen ve Kerem Tiryaki ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “Eğlence İçerikli Oyunu” başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı: Prof. Dr. Adı SOYADI ÖĞR. GÖR. HAKAN BURAK EMEKLİ

İstanbul Aydın Üniversitesi

Jüri Üyeleri: Prof. Dr. Adı SOYADI ………………………

İstanbul Aydın Üniversitesi

Jüri Üyeleri: Prof. Dr. Adı SOYADI ………………………

İstanbul Aydın Üniversitesi

14/06/2024

# Onur Sözü

Lisans tezi olarak sunduğum “Eğlence İçerikli Oyun” çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Kaynakça ’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim.

Bilal Örgen

Kerem Tiryaki

# Teşekkür Sayfası

Tezimizin hazırlanmasında bizlere yardımcı olan Sayın Dr. Öğr. Üyesi Hakan Burak Emekli hocamıza teşekkürlerimi sunarım.

Bilal Örgen & Kerem Tiryaki

# Özet

Projemizde geliştirdiğimiz oyunlar, kullanıcıların oynarken sıkılmayacakları, istedikleri zaman kapatabilecekleri, rahat, kolay ve oynarken düşündürecek bir zaman geçirme amacı ile tasarlanmıştır. Oyunlar hem düşündüren hem de rahat vakit geçirebilecek bir oyun bütünüdür. İlk olarak, kullanıcıların bilgilerini ölçmek ve şansa bağlı olarak iyi puanlar kazanabilecekleri bir oyun olan çarkıfeleği tasarladık. Çarkıfelek, en iyi puan veren oyun olması yanında en çok kaybettiren oyun olarak ödül risk sistemini oturtmaya çalıştık. Adam asmacada ise nostaljik duygularla yaptık. Kelimeler her türden olup tahmini zorlaştırdık, böylece kelime bilgisinin düzeyi belirleniyor. Yılan oyunları ise basit ve eğlenceli olmasına ve yanı sıra güzel puan kazandırarak sürekliliğini arttırdık. Renk hafıza oyununda ise oyuncuların hafızalarının ne kadar güçlü olduğunu ve bu hafızayı pekiştirmesini sağladık. Oyunun bitmesi için belirli bir puana sahip olup mağazadaki ilgili butona tıklanması gerekmektedir. Projede güncel teknolojiler kullanmayı da göz ardı edilmedi. Yapay Zeka tabanlı görselleri belirlediğimiz etiketlerden oluşturarak projemize katkı sunduk.

**ANAHTAR KELİMELER:** Çarkıfelek, PyQt5, Python, Adam Asmaca, Hafıza İçerikli, Ödül, Risk, Puan, Qt Designer, Framework, Visual Studio Code, DB Browser Lite, SqL, Sorgu, Veri, Veri tabanı, Görseller, Müzik dosyası,

# Abstract

The games we have developed in our project are designed with the aim of spending a comfortable, easy and thought-provoking time that users will not get bored while playing, and that they can turn off whenever they want. The games are a set of games that can both make you think and have a comfortable time. First of all, we designed the wheel of fortune, a game where users can measure their knowledge and earn good points depending on luck. Wheel of fortune is the game that gives the best points as well as being the most losing game, we tried to establish the reward risk system. In Hangman, we did it with nostalgic feelings. The words are of all kinds and we made it difficult to guess, thus determining the level of vocabulary knowledge. Snake games are simple and fun, and we have increased their continuity by earning good points. In the color memory game, we made sure that the players know how strong their memory is and reinforce this memory. In order for the game to end, it is necessary to have a certain number of points and click on the relevant button in the store. The use of current technologies was not ignored in the project. We contributed to our project by creating Artificial Intelligence-based visuals from the tags we determined.

**KEY WORDS:** Wheel of Fortune, PyQt5, Python, Hangman, Memory Content, Reward, Risk, Score, Qt Designer, Framework, Visual Studio Code, DB Browser Lite, SqL, Query, Data, Database, Visuals, Music file,

İçindekiler

[Onay Sayfası ii](#_Toc168356461)

[Onur Sözü iii](#_Toc168356462)

[Teşekkür Sayfası iv](#_Toc168356463)

[Özet v](#_Toc168356464)

[Abstract vi](#_Toc168356465)

[Kısaltmalar ix](#_Toc168356466)

[Şekiller 1](#_Toc168356467)

[GİRİŞ 2](#_Toc168356468)

[1.VİDEO OYUNLARININ TARİHİ 3](#_Toc168356469)

[2.PROJE 4](#_Toc168356470)

[2.1 Proje Yapılırken Kullanılan Araçların Tanıtımı 4](#_Toc168356471)

[2.2 sayfabir.py - Yükleme Ekranı (Load UI Screen) 7](#_Toc168356472)

[2.3 yeniduzen.py (Menü Ekranı) 8](#_Toc168356473)

[2.4 yeniduzen2.py (Oyun Menüsü) 10](#_Toc168356474)

[2.5 adamasmaca.py 15](#_Toc168356475)

[2.6 snake1.py 22](#_Toc168356476)

[2.7 carkifelek.py 26](#_Toc168356477)

[2.8 cf2.py 29](#_Toc168356478)

[2.9 renkli.py 31](#_Toc168356479)

[3.0 convert.py 42](#_Toc168356480)

[3.1 imgler\_rc.py 42](#_Toc168356481)

[3.2 main.py 43](#_Toc168356482)

[3.3 carkifeleksoru.db 56](#_Toc168356483)

[3.4 Kullanılan Görseller (img, gif, jpeg, png) 56](#_Toc168356484)

[3.5 music.mp3 57](#_Toc168356485)

[3.6 .ui dosyaları 57](#_Toc168356486)

[Kaynakça 58](#_Toc168356487)

# Kısaltmalar

Vsc: Visual Studio Code

Fonk. : Fonksiyon

İmg : Resim dosyası.

Png: Daha küçük resim dosyası türü.

.py: Python dosyası uzantısı

Qrc : QT Kaynak dosyası.

Css: Cascading Style Sheets (Basamaklı Stil Şablonları)

Qss: QT Style Sheets

.ui: Arayüz Dosyası

# Şekiller

|  |  |  |
| --- | --- | --- |
| Şekil 1.1 VSC  Şekil 1.2 Qt Designer  Şekil 1.3 Db Browser  Şekil 1.4 Adobe Photoshop  Şekil 1.5 Stable Diffusion  Şekil 2.1 Kod  Şekil 2.2 Kod  Şekil 2.3 Kod  Şekil 2.4 Kod  Şekil 2.5 Kod  Şekil 2.6 Kod  Şekil 2.7 Kod  Şekil 2.8 Kod  Şekil 2.9 Kod  Şekil 3.0 Kod  Şekil 3.1 Kod  Şekil 3.2 Kod  Şekil 3.3 Kod  Şekil 3.4 Kod  Şekil 3.5 Kod  Şekil 3.6 Kod  Şekil 3.7 Kod  Şekil 3.8 Kod  Şekil 4.9 Kod  Şekil 5.0 Kod  Şekil 5.1 Kod  Şekil 5.2 Kod  Şekil 5.3 Kod | Şekil 5.4 Kod  Şekil 5.5 Kod  Şekil 5.6 Kod  Şekil 5.7 Kod  Şekil 5.8 Kod  Şekil 5.9 Kod  Şekil 6.0 Kod  Şekil 6.1 Kod  Şekil 6.2 Kod  Şekil 6.3 Kod  Şekil 6.4 Kod  Şekil 6.5 Kod  Şekil 6.6 Kod  Şekil 6.7 Kod  Şekil 6.8 Kod  Şekil 6.9 Kod  Şekil 7.0 Kod  Şekil 7.1 Kod  Şekil 7.2 Kod  Şekil 7.3 Kod  Şekil 7.4 Kod  Şekil 7.5 Kod  Şekil 7.6 Kod  Şekil 7.7 Kod  Şekil 7.8 Kod  Şekil 7.9 Kod  Şekil 8.0 Kod  Şekil 8.1 Kod | Şekil 8.2 Kod  Şekil 8.3 Kod  Şekil 8.4 Kod  Şekil 8.5 Kod  Şekil 8.6 Kod  Şekil 8.7 Kod  Şekil 8.8 Kod  Şekil 8.9 Kod  Şekil 9.0 Kod  Şekil 9.1 Kod  Şekil 9.2 Kod  Şekil 9.3 Kod  Şekil 9.4 Kod  Şekil 9.5 Kod  Şekil 9.6 Kod  Şekil 9.7 Kod  Şekil 9.8 Kod  Şekil 9.9 Kod  Şekil 10.0 Kod  Şekil 10.1 Kod  Şekil 10.2 Kod  Şekil 10.3 Veritabanı |

# GİRİŞ

Bu projede geliştirdiğimiz oyunlar, kullanıcıların eğlenceli, rahat bir zaman geçirmelerini sağlamak amacıyla tasarlanmıştır. Kullanıcıların oyun sırasında sıkılmamaları ve istedikleri zaman oyunu kapatma isteğine sahip olmaları bizim için gerekli olmuştur. Oyunlarımız, hem düşündürten hem de rahat vakit geçirme imkanı sunan bir ortam oluşturur.

İlk olarak, kullanıcıların bilgilerini ölçmek ve şansa bağlı olarak iyi puanlar kazanabilecekleri bir oyun olan Çarkıfelek’i tasarladık. Çarkıfelek, hem en yüksek puanları veren hem de en çok kaybettiren oyun olarak ödül-risk sonuçlarını tamamen şansa dayandırmıştır. Bu oyun, kullanıcıların şanslarını denemeleri ve kültürel bilgilerini test etmeyi amaçlar.

Bir diğer oyunumuz olan Adam Asmaca, nostaljik bir oyun olması sebebiyle tasarlanmıştır. Bu oyunda, her türden kelimeler kullanarak tahmin etmeyi zorlaştırdık ve böylece kullanıcıların kelime bilgisini ölçmeyi hedefledik. Kullanıcılar, kelimeleri tahmin ederek bilgi seviyelerini artırabilirler. Kelimeleri tahmin edemezse bile kaybettiğinden fazla harf bilmeleri kullanıcılarımızın yararına olacaktır.

Yılan oyunları, basit ve eğlenceli yapısıyla dikkat çekmektedir. Bu oyunlar, kullanıcıların keyifli vakit geçirirken yüksek puanlar kazanmalarını sağlamakta ve bu sayede oyunun sürekliliğini artırmaktadır.

Renk Hafıza oyunumuz ise, oyuncuların hafıza yeteneklerini test etmelerini ve pekiştirmelerini sağlamaktadır. Tamamen kullanıcının kişisel hafızasının yeterliliğine bağlı kalacaktır. Bu mekanizma, kullanıcıları hatırlama konusunda geliştirici bir unsurdur.

Projemizde, güncel teknolojilere de yer verilmiştir. Yapay Zeka tabanlı görseller, belirlediğimiz etiketlerden oluşturularak oyunlarımızın görsel kalitesini ve çeşitliliğini artırmıştır. Bu sayede, kullanıcı deneyimini daha zengin ve çekici hale getirmeyi amaçladık.

Kullanıcı dostu arayüze sahip olup ve ödül-risk dengesi gibi unsurlarla, oyunlarımızın ilgi çekici ve sürdürülebilir olmasını sağladık. Projemizi bir sonraki sayfalarda gelin ele alalım.

# 1.VİDEO OYUNLARININ TARİHİ

Bilinen tarihte ilk video oyun 1952 yılında Cambridge Üniversitesinde Alexander S. Douglas tarafından geliştirip OXO adlı oyun üç taş oyununu simüle ediyordu.

60’lı yıllarda Spacewar! Oyununu geliştirildi. 70’li yıllarda da Altın Çağ başladı. Arcade oyunlar video oyun endüstrinin temellerini oluşturmasında rol aldı. 1977 yılında Atari, Atari 2600 adlı video oyun konsolunu piyasaya sürdüler. Atari 2600 çok geniş yelpazeli bir oyun kütüphanesi içeriyordu.

80’li yıllarda Pac-Man gibi dünya çapında fenomen bir oyun piyasaya sürüldü. Ama bu yıllarda Video Oyun Krizi ya da seferberliği yaşandı diyebiliriz. Düşük kaliteli oyunlar, fazlaca kısa doyumluk oyunlar çıkınca önemli şekilde tüketicinin güveninin düşürmesine sebep oldu. Atari’ nin E.T the Extra-Terrestrial oyunu krizin önemli bir görünümü oldu. Beklenen başarıyı yakalayamamasının sebebi kalite sorunlarından başladı. Fazla üretimler nedeniyle Ataride stoklar birikti ve mali kayıplar yaşandı.

85’li yıllarda Nintendo Super Mario Bros. Oyununu piyasaya sürdü ve endüstri tekrardan yeniden canlanmaya başladı.

90’lı yıllar 3D platform oyunları, CD-Rom oyunları, PlayStation tabanlı oyunlar oynanmasını mümkün kıldı. Nintendo 64 adlı konsol da bu yıllarda piyasaya sürüldü.

2000 lerin yılların başlamasıyla yeni konsollar ve dönemler başladı.

Yılın başında Sony, PlayStation 2’yi piyasaya sürdü. DVD oynatma özelliği ve geniş video oyunu kütüphanesiyle en çok satan oyun konsolu oldu. 2001 yılında Microsoft, XBOX adlı konsolu piyasaya sürdü. 2005 yılında, Microsoft, Xbox 360 adlı konsolu piyasaya sürdü ve bu, online oyunların ve dijitalin popülerleşmesini sağladı. 2006 yılında, Sony, PlayStation 3 ve Nintendo, Wii adlı konsolları piyasaya sürdü. Wii, hareket sensörlü kontrolleri ile dikkat çekti ve geniş bir oyuncu kitlesine elde etti.

2010 yıllara gelirken Mobil Oyunlar dijitalde yükseliş yaşadı. Akıllı telefonların yaygınlaşması ile mobil oyunlar büyük bir popülerlik kazandı. Angry Birds, Candy Crush Saga gibi oyunlar milyonlarca oyuncuya ulaştı.

2013 yılında, Sony, PlayStation 4 ve Microsoft, Xbox One adlı konsolları piyasaya sürdü. Bu konsollar, güçlü donanımları ve online oyun hizmetleri ile dikkat çekti. Sanal gerçeklik (VR) teknolojileri, Oculus Rift ve PlayStation VR gibi cihazlarla popüler hale geldi. Bu teknolojiler, oyun deneyimini tamamen yeni bir boyuta taşıdı.

2020’li yıllarda da yeni nesil konsollar ve bulut içerikli oyunlar gelişmeye başladı. 2020 yılında, Sony, PlayStation 5 ve Microsoft, Xbox Series X ve Series S adlı konsolları piyasaya sürdü. Bu konsollar, yüksek performansları ve hızlı yükleme süreleri ile dikkat çekti.

Bulut oyunları, Google Stadia, NVIDIA GeForce Now ve Xbox Cloud Gaming gibi hizmetlerle yaygınlaşmaya başladı. Bu teknolojiler, güçlü donanım gereksinimlerini ortadan kaldırarak oyunlara her yerden erişim imkânı sağladı.

# 2.PROJE

Kullanıcıların eğlenceli, rahat bir zaman geçirmelerini sağlamak amacıyla “Eğlence İçerikli Oyunu” muzu sunmaya karar verdik. Oyunumuz Python dilinde Windows platformunda oynanabilecek vakit geçirebileceğiniz bir deneyim sunar.

# 2.1 Proje Yapılırken Kullanılan Araçların Tanıtımı

**Microsoft Visual Studio Code**

VSC, açık kaynaklı olması sayesinde geniş bir kullanıcı kitlesine sahip ve sürekli olarak gelişiyor. Kod düzenleme deneyimini iyileştiren birçok özelliği var, mesela kod otomatik tamamlama ve kesme gibi. Hatta kodlarımızda karşılaştığımız hatalarımızı detaylı olarak düzenleme fırsatı veriyor. Ayrıca, birçok programlama dilini destekliyor, bu da farklı projelerde çalışmak isteyen geliştiriciler için kolaylık sağlıyor. Eklentileri oldukça geniş ve kullanıcılar ihtiyaçlarına göre özelleştirebiliyor.

**Şekil 1.1 VSC**

Hem Windows'ta hem de diğer işletim sistemlerinde kullanılabilmesi büyük bir avantaj. Entegrasyonlar, geliştirme sürecini hızlandırıyor ve GitHub gibi popüler araçlarla uyumlu. Hafif bir kod editörü olmasına rağmen güçlü bir geliştirme ortamı sunuyor, bu da performans açısından verimli bir deneyim sağlıyor. Ayrıca, aktif bir topluluğa sahip olması da sorunlarımızı çözmemiz ve yeni şeyler öğrenmemiz için bize destek sağlıyor. Bu nedenlerle, biz geliştiriciler arasında popüler bir tercih haline geliyor.

**Qt Designer**

****

**Şekil 1.2 QT Designer**

Qt Designer, kullanıcı arayüzü tasarımı için kullanılan bir grafik arayüz editörüdür. Kullanıcılar, sürükle ve bırak yöntemiyle arayüz öğelerini düzenleyebilir ve özelleştirebilirler. Qt Designer, Qt framework'üyle entegre çalışır ve PyQt veya PySide gibi Python kütüphaneleriyle birlikte kullanılabilir. Bu sayede, program geliştiricileri, kullanıcı dostu ve estetik arayüzler oluşturabilirler. Basit ve kullanıcı dostu arayüzü sayesinde, geliştirme sürecini hızlandırır ve kolaylaştırır. Projemizde PyQt 5 kütüphanelerini kullandık.

**DB Browser SQLite**



**Şekil 1.3 DB Browser SQLite**

Db Browser for SQLite, veritabanı oluşturmak, düzenlemek ve sorgulamak için kullanılan bir araçtır. Bu program, kullanıcı dostu bir arayüz sunar ve SQLite veritabanı dosyalarıyla uyumludur. Veritabanı şemalarını görüntüleyebilir, tablolar arasında ilişkiler kurabilir ve veri girişlerini düzenleyebilirsiniz. SQL sorgularını doğrudan çalıştırabilir ve sonuçları anında görüntüleyebilirsiniz. Db Browser for SQLite, veritabanı yönetimi için pratik bir çözüm sunar ve geliştiricilerin veritabanı işlemlerini kolayca gerçekleştirmesine yardımcı olur. Ayrıca, veritabanı dosyalarını farklı formatlarda içe ve dışa aktarma seçenekleri sunar. Kullanımı kolay arayüzü ve işlevselliği ile, SQLite tabanlı projeler için ideal bir araçtır.

**Adobe Photoshop**



**Şekil 1.4 Adobe Photoshop**

Adobe Photoshop, profesyonel fotoğraf düzenleme ve grafik tasarım yazılımıdır. Kullanıcılar, fotoğrafları düzenleyebilir, grafikleri oluşturabilir ve dijital sanat eserleri yaratabilirler. Kapsamlı olmasıyla renk düzeltme, kesme, kopyalama ve yapıştırma gibi işlemleri kolaylaştırır. Katmanlar, efektler ve filtreler gibi gelişmiş özelliklerle, kullanıcılar hayal güçlerini sınırlamadan yaratıcılıklarını ifade edebilirler. Adobe Photoshop'un arayüzü ve öğrenme kaynakları hem deneyimli kullanıcılar hem de yeni başlayanlar için uygundur. Oyunumuzda birkaç label ve oyun isminin görsellerini ve yazılarını bu programı kullanarak meydana getirdik.

**Stable Diffusion**

Stable Diffusion, fotoğrafları metin olmadan sentezlemek için gizli difüzyon adı verilen bir teknik kullanarak çarpıcı görseller oluşturur. Yapay zeka, resimleri gizli bir alana sıkıştırır ve ardından istenen sonuç elde edilene kadar tekrar tekrar gürültü ekler ve çıkarır. elde edilir.

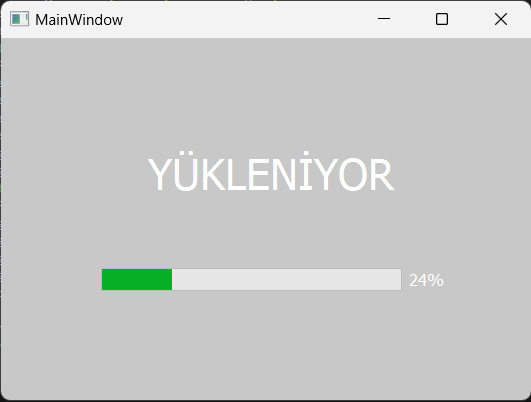
**Şekil 1.5 Stable Diffusion**

# 2.2 sayfabir.py - Yükleme Ekranı (Load UI Screen)

 Şekil 2.1 deki kodlar, PyQt5 kütüphanesini kullanarak bir kullanıcı arayüzü tasarlar ve oluşturur. Bu dosyanın adı "sayfabir.py" olarak belirtilmiştir. Kodlar, PyQt5 modüllerini içe aktarır ve "Ui\_MainWindow" adında bir sınıf tanımlar. Bu sınıf, bir ana pencere oluşturmak için "setupUi" ve "retranslateUi" adında iki metot içerir. "setupUi" metodu, ana pencerenin özelliklerini ve içeriğini tanımlar, örneğin boyutu, arka plan rengi, etiket ve ilerleme çubuğu gibi bileşenlerin yerleşimi ve stilini ayarlar. "retranslateUi" metodu ise, arayüzdeki metinlerin çevirisini yapar. Ana kod bloğu, PyQt5 uygulamasını başlatır."Ui\_MainWindow" sınıfından bir örnek oluşturarak ana pencereyi oluşturur.

**Şekil 2.1 Kod**

Son olarak, uygulamayı çalıştırır ve PyQt5 event loop'unu başlatır. Bu kodlar, kullanıcı arayüzü tasarlamak için PyQt5 kütüphanesinin temel özelliklerini ve kullanımını gösterir. Şekil 2.2 deki de kod dosyamızın görünümüdür.

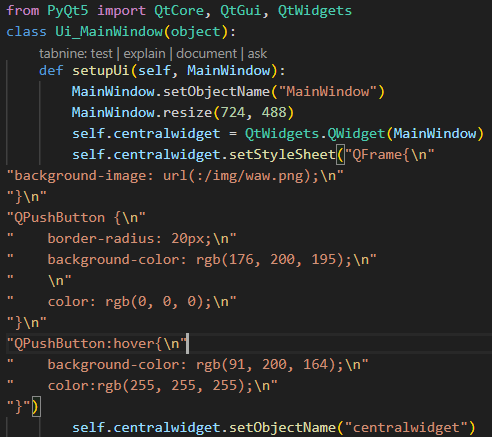
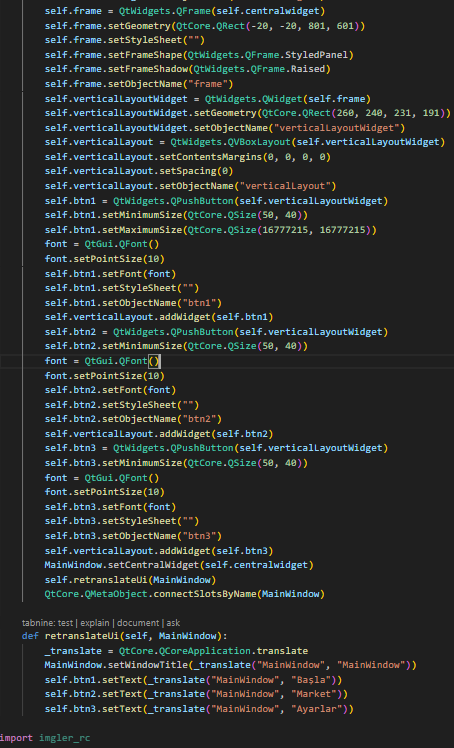


**Şekil 2.2 Çıktı**

# 2.3 yeniduzen.py (Menü Ekranı)

Şekil 2.3 Kod görseli, PyQt5 kütüphanesini kullanarak bir kullanıcı arayüzü oluşturan bir Python sınıfı içerir. "Ui\_MainWindow" sınıfı, bir ana pencerenin görünümünü ve davranışlarını tanımlar. "setupUi" metodu, ana pencerenin boyutunu ve içeriğini ayarlar. Ana pencerenin boyutu "724x488" piksel olarak belirlenmiştir.

Ana pencerenin içeriği "centralwidget" adında bir QWidget nesnesi olarak tanımlanır. Bu nesne, ana pencerenin içinde bulunan tüm diğer arayüz öğelerini içerir.

Stil sayfası, PyQt5'nin içinde yer alan CSS benzeri bir dil olan "QSS" (Qt Style Sheets) kullanılarak tanımlanmıştır. Bu stil sayfası, arka plan görüntüsü ve düğme stilleri gibi öğelerin görünümünü belirler. Arkaplan görüntüsü "waw.png" adlı bir dosyadan alınır. Düğmelerin stili, normal durumda "rgb(176, 200, 195)" arka plan rengi ve "20px" yuvarlak köşelere sahip olacak şekilde tanımlanmıştır.

**Şekil 2.3 Kod**

Fare üzerine geldiğinde ise arka plan rengi "rgb(91, 200, 164)" ve yazı rengi "rgb(255, 255, 255)" olarak değişir.Bu kodlar, PyQt5 kullanarak özelleştirilmiş bir kullanıcı arayüzü tasarlamanın temel bir örneğini sunar.

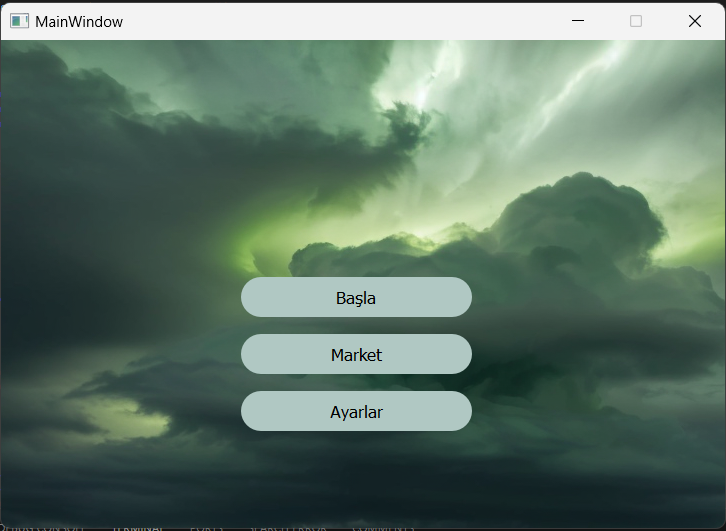
Şekil 2.4 teki kodlarım, bir PyQt5 uygulamasının kullanıcı arayüzünü oluşturan bir sınıf içerir. İlk olarak, bir QFrame nesnesi oluşturulur ve bu çerçeve ana pencerenin içinde yer alacak şekilde boyutlandırılır.

Çerçeve, özel bir şekilde çizilmiş bir panel olarak stilize edilmiştir ve yükseltilmiş bir gölge efekti ile vurgulanmıştır.

**Şekil 2.4 Kod**

Sonra, çerçeve içine bir dikey düzen nesnesi (QVBoxLayout) yerleştirilir ve bu düzen içine üç adet QPushButton eklenir.

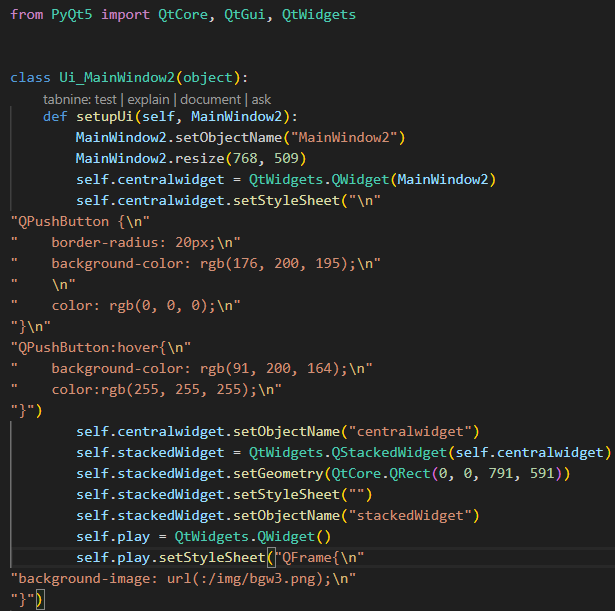
Her düğme, belirli bir işlevi temsil eder: "Başla", "Market" ve "Ayarlar".Ana kod bloğunda, QMainWindow sınıfından bir örnek oluşturulur ve Ui\_MainWindow sınıfı kullanılarak arayüz tasarımı yapılır. Son olarak, retranslateUi() metodu aracılığıyla metin çevirileri yapılır. Bu kodlar, PyQt5 kullanarak özelleştirilmiş bir ana pencere tasarlamanın bir örneğini gösterir ve "imgler\_rc" adında bir dosyaya dışa aktarılan görüntü kaynaklarını kullanır. Şekil 2.5 de kodlarımızın önizlemesidir.

****

**Şekil 2.5 Çıktı**

# 2.4 yeniduzen2.py (Oyun Menüsü)

MainWindow2.setObjectName("MainWindow2"): Pencerenin ismini MainWindow2 olarak ayarlar.

MainWindow2.resize(768, 509): Pencerenin boyutlarını genişlik olarak 768 piksel, yükseklik olarak 509 piksel olarak ayarlar.

self.centralwidget.setStyleSheet(...): Merkez widget için stil tanımları yapar: QPushButton {...}: Tüm butonlar için köşe yuvarlama, arka plan rengi ve yazı rengi ayarları. QPushButton:hover {...}: Fareyle butonun üzerine gelindiğinde uygulanacak stil (arka plan ve yazı rengi değişikliği).

**Şekil 2.6 Kod**

self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget): Merkez widget içerisinde bir QStackedWidget oluşturur. QStackedWidget, birden fazla widget arasında geçiş yapmaya olanak tanır.

self.stackedWidget.setGeometry(QtCore.QRect(0, 0, 791, 591)): stackedWidgetin boyutlarını ve konumunu ayarlar.

self.stackedWidget.setStyleSheet(""): stackedWidget için herhangi bir stil tanımlamaz (boş bırakılmış).

self.play = QtWidgets.QWidget(): Yeni bir widget oluşturur ve bunu play olarak adlandırır.

self.play.setStyleSheet("QFrame{\nbackground-image: url(:/img/bgw3.png);\n}"): play widget'ı için bir stil tanımlar:

QFrame{background-image: url(:/img/bgw3.png);}: Bu, QFrame sınıfına ait bir widget'in arka plan resmini bgw3.png dosyası ile değiştirir.

self.play widget'ı içerisine bir QFrame ekleniyor ve ardından bu çerçeve (frame) içerisine dört tane buton (obt1, obt2, obt3, obt4) yerleştiriliyor.

self.play.setObjectName("play"): play widget'ına "play" ismini verir.

self.frame = QtWidgets.QFrame(self.play): play widget'ı içerisine bir çerçeve (frame) oluşturur.

self.frame.setGeometry(QtCore.QRect(0, 0, 791, 591)): Çerçevenin boyutlarını ve konumunu ayarlar.

font = QtGui.QFont(): Yeni bir font objesi oluşturur.

font.setFamily("Lucida Fax"): Font ailesini "Lucida Fax" olarak ayarlar.

font.setPointSize(13): Font boyutunu 13 yapar.

**Şekil 2.7 Kod**

font.setBold(True): Fontu kalın yapar. font.setWeight(75): Font ağırlığını 75 olarak ayarlar.

self.frame.setFont(font): Çerçevenin fontunu ayarlar. self.frame.setStyleSheet(""): Çerçeve için herhangi bir stil ayarlamaz. self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel):

Çerçeve şekli olarak StyledPanel belirler.

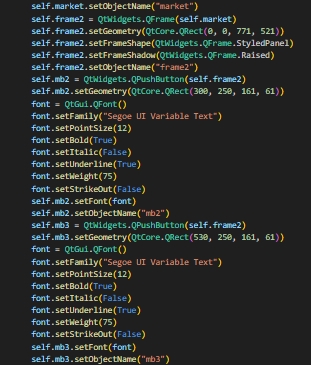
self.frame.setFrameShadow(QtWidgets.QFrame.Raised): Çerçeve gölgesini Raised olarak ayarlar.

Obt1 – Obt2 – Obt3 – Obt4 Butonları:

self.frame.setObjectName("frame"): Çerçeveye "frame" ismini verir.

self.frame = QtWidgets.QFrame(self.play): play widget'ı içerisine bir çerçeve (frame) oluşturur.

self.frame.setGeometry(QtCore.QRect(0, 0, 791, 591)): Çerçevenin boyutlarını ve konumunu ayarlar.



**Şekil 2.8 Kod Şekil 2.9 Kod**

self.labelpuan: Çerçeve içinde bir etiket oluşturur. self.labelpuan.setGeometry(...): Etiketin konumunu ve boyutlarını ayarlar.

self.labelpuan.setStyleSheet("color: white;"): Etiketin yazı rengini beyaz yapar.

self.labelpuanyazi: Çerçeve içinde başka bir etiket oluşturur.

self.labelpuanyazi.setGeometry(...): Etiketin konumunu ve boyutlarını ayarlar.

self.labelpuanyazi.setStyleSheet("background-image: url(:/img/puan.png);"): Etiketin arka planını bir resimle değiştirir.

self.labelpuanyazi.setText(""): Etiketin metnini boş bırakır.

self.labelpuanyazi.setObjectName("labelpuanyazi"): Etikete "labelpuanyazi" ismini verir.

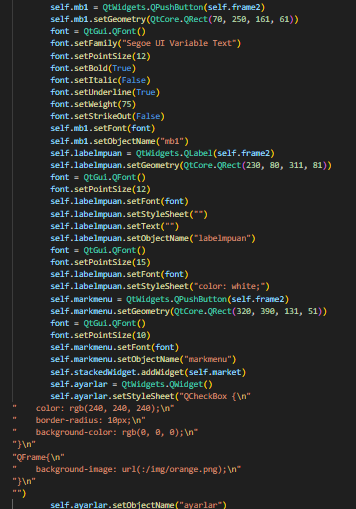
self.playmenu: Çerçeve içinde bir buton oluşturur.

self.playmenu.setObjectName("playmenu"): Butona "playmenu" ismini verir.

self.market: Yeni bir market widget'ı oluşturur.

self.frame2 = QtWidgets.QFrame(self.market): Market widget'ı içinde bir çerçeve (frame2)

self.frame2.setFrameShadow(QtWidgets.QFrame.Raised): Çerçeve gölgesini Raised olarak ayarlar. self.frame2.setObjectName("frame2"): Çerçeveye "frame2" ismini verir.



**Şekil 3.1 Kod**

**Şekil 3.0 Kod** Mb1 – Mb2 – Mb3 Butonları genel görünümü:

self.mb2: Çerçeve içinde ikinci bir buton oluşturur. self.mb2.setGeometry(...): Butonun konumunu ve boyutlarını ayarlar. font = QtGui.QFont(): Yeni bir font objesi oluşturur. font.setFamily("Segoe UI Variable Text"): Font ailesini "Segoe UI Variable Text" olarak ayarlar. font.setPointSize(12): Font boyutunu 12 yapar. font.setBold(True): Fontu kalın yapar. font.setUnderline(True): Fontun altını çizer. font.setWeight(75): Font ağırlığını 75 yapar. self.mb2.setFont(font): Butonun fontunu ayarlar. self.mb2.setObjectName("mb2"): Butona "mb2" ismini verir.

self.labelmpuan: Çerçeve içinde bir etiket oluşturur. self.labelmpuan.setGeometry(...): Etiketin konumunu ve boyutlarını ayarlar. font = QtGui.QFont(): Yeni bir font objesi oluşturur. font.setPointSize(12): Font boyutunu 12 yapar. self.labelmpuan.setFont(font): Etiketin fontunu ayarlar. self.labelmpuan.setObjectName("labelmpuan"): Etikete "labelmpuan" ismini verir. font.setPointSize(15): Font boyutunu 15 yapar. self.labelmpuan.setStyleSheet("color: white;")

Market’ten Menüye Buton:

self.markmenu: Çerçeve içinde bir buton oluşturur. self.markmenu.setGeometry(...): Butonun konumunu ve boyutlarını ayarlar. font = QtGui.QFont(): Yeni bir font objesi oluşturur. font.setPointSize(10): Font boyutunu 10 yapar. self.markmenu.setFont(font): Butonun fontunu ayarlar. self.markmenu.setObjectName("markmenu"): Butona "markmenu" ismini verir. self.stackedWidget.addWidget(self.market): market widget'ını stackedWidgete ekler.

self.ayarlar: Yeni bir ayarlar widget'ı oluşturur. self.ayarlar.setStyleSheet(...): Ayarlar widget'ının ve içindeki bileşenlerin stil ayarlarını yapar. self.ayarlar.setObjectName("ayarlar"): Ayarlar widget'ına "ayarlar" ismini verir. self.frame3 = QtWidgets.QFrame(self.ayarlar): Ayarlar widget'ı içinde bir çerçeve (frame3) oluşturur. self.frame3.setGeometry(...): Çerçevenin boyutlarınıve konumunuayarlar.

self.frame3.setFrameShape(QtWidgets.QFrame.StyledPanel): Çerçeve şekli olarak StyledPanel belirler. self.frame3.setFrameShadow(QtWidgets.QFrame.Raised): Çerçeve gölgesini Raised olarak ayarlar. self.frame3.setObjectName("frame3"): Çerçeveye "frame3" ismini verir.

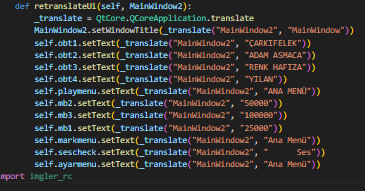
CheckBox:

self.sescheck: Çerçeve içinde bir check box oluşturur. self.sescheck.setGeometry(...): Check box'ın konumunu ve boyutlarını ayarlar. font = QtGui.QFont(): Yeni bir font objesi oluşturur. font.setPointSize(11): Font boyutunu 11 yapar. font.setBold(True): Fontu kalın yapar. font.setWeight(75): Font ağırlığını 75 yapar. self.sescheck.setFont(font): Check box'ın fontunu ayarlar. self.sescheck.setObjectName("sescheck"): Check box'a "sescheck" ismini verir.

Ayarlardan Menüye Buton:

self.ayarmenu: Çerçeve içinde bir buton oluşturur. self.ayarmenu.setGeometry(...): Butonun konumunu ve boyutlarını ayarlar. font = QtGui.QFont(): Yeni bir font objesi oluşturur. font.setPointSize(10): Font boyutunu 10 yapar. self.ayarmenu.setFont(font): Butonun fontunu ayarlar. self.ayarmenu.setObjectName("ayarmenu"): Butona "ayarmenu" ismini verir.

self.stackedWidget.addWidget(self.ayarlar): ayarlar widget'ını stackedWidgete ekler. retranslateUi Fonksiyonu: Bu fonksiyon, PyQt5 ile oluşturulan kullanıcı arayüzünde yer alan metinlerin belirli bir dile çevrilmesini sağlar. Genellikle uygulamanın başlangıcında veya dil değişikliği olduğunda çağrılır.

İşlevi, arayüzdeki metinlerin doğru ve güncel bir şekilde görüntülenmesini sağlamaktır.  
 **Şekil 3.2 Kod**

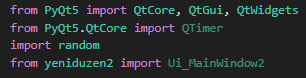
\_translate, QtCore.QCoreApplication.translate fonksiyonunu kullanarak metinlerin çevrilmiş halini alır. Pencerenin başlığını "MainWindow" olarak ayarlar.

self.obt1, self.obt2, self.obt3, self.obt4 ve self.playmenu butonlarının metinlerini sırasıyla "ÇARKIFELEK", "ADAM ASMACA", "RENK HAFIZA", "YILAN" ve "ANA MENÜ" olarak ayarlar.

self.mb2, self.mb3 ve self.mb1 butonlarının metinlerini sırasıyla "50000", "100000" ve "25000" olarak ayarlar.

self.markmenu, self.sescheck ve self.ayarmenu bileşenlerinin metinlerini sırasıyla "Ana Menü", "Ses" ve "Ana Menü" olarak ayarlar.

# 2.5 adamasmaca.py

random modülünü içe aktarır. Bu modül, rastgele sayı üretme işlemleri için kullanılır.

**Şekil 3.3 Kod**

yeniduzen2 isimli bir Python dosyasından Ui\_MainWindow2 sınıfını içe aktarır. Bu sınıf, PyQt5 ile tasarlanmış bir kullanıcı arayüzünü temsil eder.

puan isimli global bir değişken oluşturur ve başlangıç değerini 0 olarak ayarlar. Bu değişken, programın herhangi bir yerinde puanı takip etmek için kullanılabilir.

Ui\_MainWindow7 adında bir sınıf tanımlar. Bu sınıf, MainWindow7 adlı bir ana pencereyi yapılandırmak için kullanılan arayüz bileşenlerini içerir.

setupUi metodu, MainWindow7 penceresini yapılandırmak için kullanılır

**Şekil 3.4 Kod**

global puan ifadesi, fonksiyon içinde puan değişkenini global olarak kullanmanıza izin verir. MainWindow7 penceresinin adını "MainWindow7" olarak ayarlar ve pencerenin boyutlarını 800x600 piksel olarak belirler. MainWindow7 penceresine bir merkezi widget ekler ve bu widget'a "centralwidget" adını verir. self.centralwidget içinde bir QLabel oluşturur ve bu etiketin konumunu ve boyutlarını (0, 0, 800, 600) olarak ayarlar. self.background\_label etiketine "bck.png" adlı resmi yükler. Bu resim, arka plan olarak kullanılacaktır. Resmi, etiketin boyutlarına sığacak şekilde ölçeklendirir. Bu, resmin etiketin tamamını kaplamasını sağlar.

centralwidget içine QFrame nesnesi ekler QFrame'in konumunu ve boyutlarını (0, 0, 801, 601) olarak ayarlar. Çerçevenin şekli StyledPanel olarak ayarlanır. Çerçevenin gölge tipi Raised olarak belirlenir. Çerçevenin adını "frame" olarak ayarlar.

QLabellar:

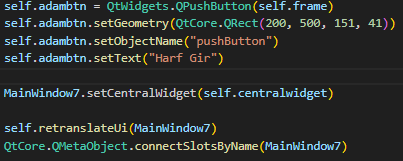
frame içinde bir QLabel oluşturur ve bu etiketin konumunu ve boyutlarını (260, 430, 321, 41) olarak ayarlar. Etiketin adını "cizgilabel" olarak belirler. Etiketin yazı tipi boyutunu 14 olarak ayarlar. Etiketin metin rengini beyaz olarak ayarlar.

**Şekil 3.5 Kod** frame içinde bir QLabel oluşturur ve bu etiketin konumunu ve boyutlarını (400, 500, 151, 41) olarak ayarlar. Etiketin yazı tipi ailesini "Tahoma" olarak ayarlar.

Etiketin yazı tipi boyutunu 12 olarak ayarlar. Etiketin adını "pnlabel" olarak belirler. Etiketin metin rengini beyaz olarak ayarlar.

frame içinde bir QPushButton oluşturur ve bu butonun konumunu ve boyutlarını (580, 450, 111, 41) olarak ayarlar. Butonun adını "menubut" olarak belirler.

MainWindow7 penceresinin merkezi widget'ını centralwidget olarak ayarlar. Bu, pencerenin ana içeriği centralwidget ve onun içindeki bileşenler olacaktır.



**Şekil 3.6 Kod**

`self.adambtn = QtWidgets.QPushButton(self.frame)`: Bir QPushButton öğesi oluşturuluyor ve `self.frame` widget'ına ekleniyor. `self.frame`, düğmenin yerleştirileceği widget'ın bir örneğidir. Bu widget bir çerçeve olabilir veya başka bir arayüz öğesi.

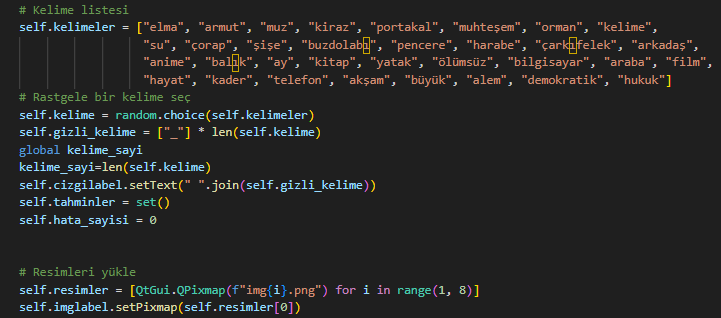
`self.adambtn.setGeometry(QtCore.QRect(200, 500, 151, 41))`: Düğmenin geometrisi (konumu ve boyutu) ayarlanıyor. QRectF (Rectangle) sınıfı kullanılarak (x, y, width, height) koordinatları belirtiliyor. Bu kod, (200, 500) konumunda, 151 piksel genişliğinde ve 41 piksel yüksekliğinde bir düğme oluşturur.

`self.adambtn.setObjectName("pushButton")`: Düğmeye bir nesne adı atanıyor. Bu, düğmeyi tanımlamak için kullanılabilir, örneğin stil yapısı dosyasında.

`self.adambtn.setText("Harf Gir")`: Düğmenin metni ayarlanıyor. Bu durumda, düğmenin üzerinde "Harf Gir" metni görünecek.

`self.retranslateUi(MainWindow7)`: Arayüz öğelerinin metinlerini yeniden çevirir. Bu, dil değiştirildiğinde veya yerelleştirme yapıldığında kullanılır.

`QtCore.QMetaObject.connectSlotsByName(MainWindow7)`: PyQt5'nin otomatik bağlama mekanizmasını kullanarak, belirli bir nesnenin sinyallerini belirli slotlara bağlar. Bu kod, belirli sinyal-slot bağlantılarını otomatik olarak yapar. Bu genellikle Qt Designer ile oluşturulan arayüzlerde kullanılır.



**Şekil 3.7 Kod**

Şekil 3.7 Kod Parçasında: self.kelimeler : Bir kelime listesi oluşturuluyor. Bu liste içinde çeşitli kelimeler bulunuyor. self.kelime = random.choice(self.kelimeler): Kelime listesinden rastgele bir kelime seçiliyor. `random.choice()` fonksiyonu kullanılarak rastgele bir kelime belirleniyor. self.gizli\_kelime = ["\_"] \* len(self.kelime)`: Seçilen kelimenin harfleri yerine '\_' karakterinden oluşan bir gizli kelime listesi oluşturuluyor. Bu, kullanıcının doğru harfleri tahmin etmesi gereken gizli kelimeyi temsil eder. global kelime\_sayi`: `kelime\_sayi` adında bir global değişken oluşturuluyor ve seçilen kelimenin uzunluğu bu değişkene atanıyor.

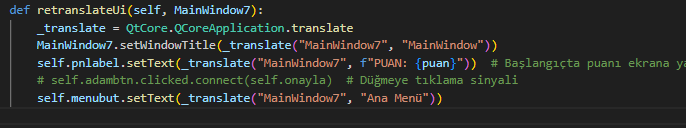
self.cizgilabel.setText(" ".join(self.gizli\_kelime))`: Gizli kelimenin her harfinin arasına boşluk eklenerek bir metin oluşturuluyor ve bu metin cizgilabel adlı bir etikete (label) ayarlanıyor. Bu etiket, kullanıcıya gizli kelimeyi göstermek için kullanılır.

self.tahminler = set()`: Kullanıcının yaptığı tahminlerin tutulacağı bir küme (set) oluşturuluyor. Bu küme, kullanıcının aynı harfi birden fazla kez tahmin etmesini önler. self.hata\_sayisi = 0`: Başlangıçta kullanıcının hata sayısı sıfırlanıyor.

self.resimler = [QtGui.QPixmap(f"img{i}.png") for i in range(1, 8)]`: 1'den 7'ye kadar olan sayılarla adlandırılmış PNG formatındaki resim dosyaları yükleniyor.

`QtGui.QPixmap()` ile resimler QPixmap nesnelerine dönüştürülüyor ve bir liste içinde saklanıyor.

self.imglabel.setPixmap(self.resimler[0])`: İlk resim (`img1.png`) imglabel adlı bir etikete atanıyor. Bu etiket, oyun sırasında kullanıcının yaptığı hatalara göre değiştirilerek oyunun ilerleyişini gösterir.



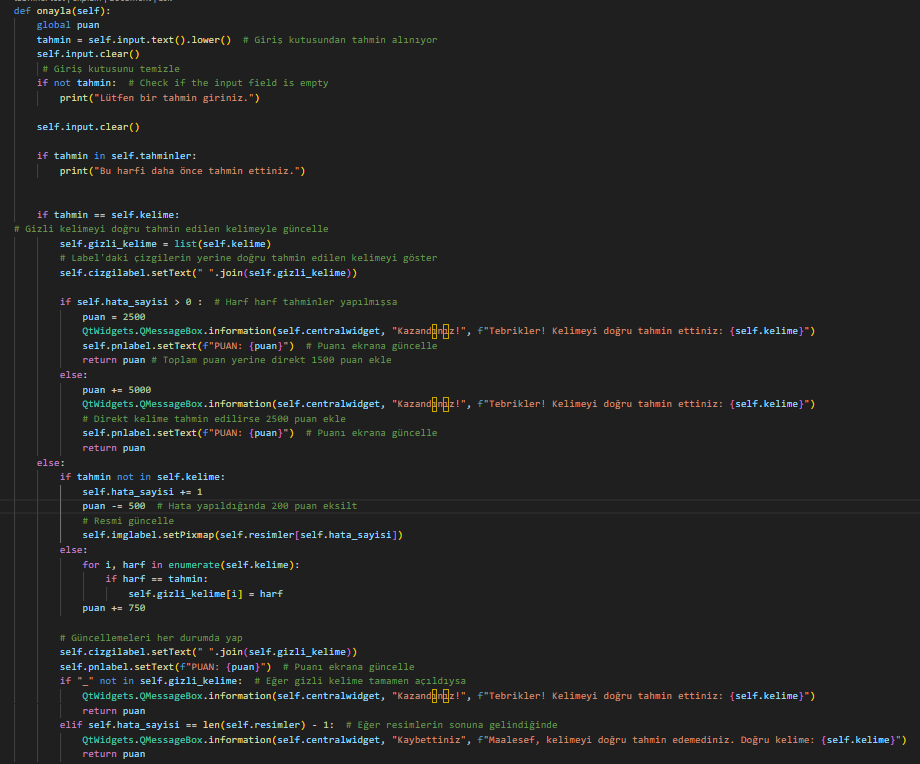
**Şekil 3.8 Kod**

\_translate = QtCore.QCoreApplication.translate: Çeviri işlemleri için bir `\_translate` işlevi oluşturuluyor. Bu işlev, metinlerin çevirilmesi için PyQt5'in sağladığı çeviri işlevini kullanır.

`MainWindow7.setWindowTitle(\_translate("MainWindow7", "MainWindow"))`: Ana pencerenin başlık metni, `\_translate` işlevi kullanılarak "MainWindow" olarak ayarlanır. Bu, pencerenin başlık çubuğunda görünecek metni belirtir.

`self.pnlabel.setText(\_translate("MainWindow7", f"PUAN: {puan}"))`: "puan" değişkenini kullanarak puan etiketi (label) için metni ayarlar. Bu, başlangıçta puanın ekrana yazdırılmasını sağlar.

`self.menubut.setText(\_translate("MainWindow7", "Ana Menü"))`: Menü düğmesi (push button) için metni ayarlar. Bu, menü düğmesinin üzerinde "Ana Menü" metninin görüneceğini belirtir.



**Şekil 3.9 Kod**

Onayla Fonksiyonu:

`global puan`: Fonksiyon içinde kullanılacak olan global `puan` değişkenini tanımlar.

`tahmin = self.input.text().lower()`: Kullanıcının giriş kutusuna yazdığı tahmini alır ve küçük harfe dönüştürür. Bu, büyük-küçük harf duyarlılığını kaldırır.

`self.input.clear()`: Kullanıcının giriş kutusunu temizler. Böylece kullanıcı yeni bir tahmin yapabilir.

`if not tahmin:`: Kullanıcının giriş kutusuna bir şey yazıp yazmadığını kontrol eder. Eğer boşsa, kullanıcıya bir tahmin girmesi gerektiğini belirten bir mesaj gösterir.

`if tahmin in self.tahminler:`: Kullanıcının daha önce aynı harfi tahmin edip etmediğini kontrol eder. Eğer kullanıcı aynı harfi daha önce tahmin etmişse, kullanıcıya bir mesaj gösterir.

`if tahmin == self.kelime:`: Kullanıcının tahmini, gizli kelimeyle eşleşiyorsa:

- Gizli kelimeyi doğru tahmin edilen kelimeyle günceller.

- Harf harf tahminler yapılmışsa, 2500 puan ekler. Aksi halde, doğru tahmin edilen kelime için 5000 puan ekler.

- Kullanıcıya doğru tahmin ettiği kelimeyi bildiren bir mesaj gösterir.

- Kullanıcının puanını günceller ve bu puanı ekranda gösterir.

- Bu durumda fonksiyon puanı döndürür.

`else:`: Kullanıcının tahmini gizli kelimeyle eşleşmiyorsa:

- Kullanıcının hata sayısını artırır.

- Kullanıcıya hata yaptığında 500 puan eksileceğini belirtir.

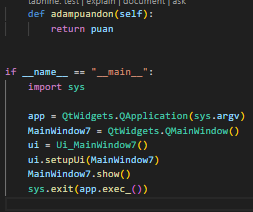
- Resmi günceller.

- Tahmin edilen harfi gizli kelimeye ekler.

- Kullanıcının puanını 750 puan artırır.

- Güncellemeleri ekrana yansıtır.

Son olarak, eğer gizli kelime tamamen açıldıysa veya kullanıcının hata sayısı resimlerin sonuna ulaştığında oyunun sonucunu gösterir ve puanı döndürür.

`def adampuandon(self): return puan`: Bu işlev, mevcut oyuncu puanını döndürür. `if \_\_name\_\_ == "\_\_main\_\_":`: Bu koşul ifadesi, bu dosyanın doğrudan çalıştırılıp çalıştırılmadığını kontrol eder. Eğer bu dosya doğrudan çalıştırılıyorsa (yani bir modül olarak başka bir dosyaya ithafen kullanılmıyorsa), içindeki kodlar çalıştırılır.

**Şekil 4.0 Kod**

`import sys`: `sys` modülü, Python yürütme ortamı hakkında bilgi sağlar.

`app = QtWidgets.QApplication(sys.argv)`: PyQt5 uygulamasını başlatır. `sys.argv` komut satırı argümanlarını alır ve uygulamaya iletilir.

`MainWindow7 = QtWidgets.QMainWindow()`: Ana pencere (QMainWindow) oluşturulur.

`ui = Ui\_MainWindow7()`: Ui\_MainWindow7 sınıfından bir örnek oluşturulur.

`ui.setupUi(MainWindow7)`: Ana pencereye (MainWindow7) arayüzü kurulumunu yapar.

`MainWindow7.show()`: Ana pencereyi gösterir.

`sys.exit(app.exec\_())`: Uygulamayı başlatır ve ana döngüye girer.

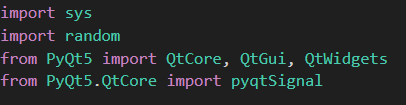
`sys.exit()` ile uygulamanın kapatılması için sinyal beklenir.

`app.exec\_()` uygulamanın döngüsünü başlatır ve kullanıcı etkileşimi bekler.

Kullanıcı uygulamayı kapattığında, `sys.exit()` çağrılır ve uygulama sonlandırılır.

# 2.6 snake1.py

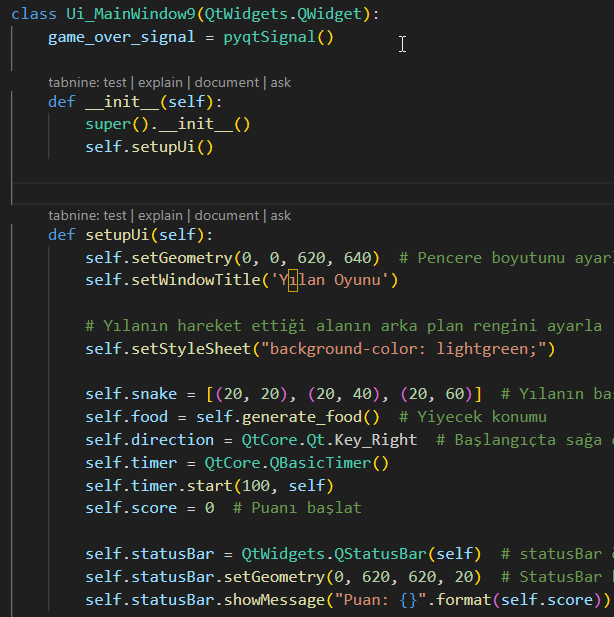
Yılan oyunu, basit ve nostaljik bir oyun oynama deneyimi sunmayı amaçlamaktadır. Bu oyun, hızlı ve eğlenceli bir şekilde puan kazandıran bir yapıya sahiptir. Oyunda yılan, ekranın dışına çıkamaz veya kendi vücudunun içinden geçemez. Her elma yendiğinde yılanın boyu uzar ve oyun zorlaşır. Yılan oyununun temel amacı, karmaşık bilgi veya hafıza gerektirmeyen bir puan kazanma sistemi üzerinden oyun oluşturulmasıdır.

Bu kod bloğunda dosyada gerekli olan kütüphaneler dosyaya tanıtılmıştır.

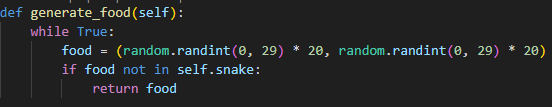
**Şekil 4.1 Kod**

pyqtSignal kütüphanesi dosyada bir yere ya da başka bir dosyaya sinyal yollayarak birbirleri arasında iletişim kurulur.

Random kütüphanesi ise bir değişkeni belirlenen sayılar arasından rastgele sayı üretmesi için kullanılan kütüphanedir.Sys kütüphanesi ise bazı python kodlarını çalıştırılmasında yardımcı olması için kullanılır.

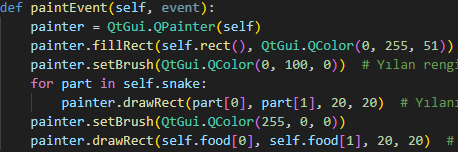


**Şekil 4.2 Kod**

Bu kod bloğunda UI\_MainWindow9 adında bir class tanımlandı ve QtWidgets.QWidget sınıfından miras alır ve bir pencere widget'ı olarak işlev görür. GameoverSignal ise oyun bittiğinde çalışacak sinyal olarak belirlenmiştir. İnit fonksyionu ise setupUI fonksiyonunu çalıştırmak için kullanılır. SetupUI dosyasında ise ekranda ne nerede nasıl gözükeceği belirlenir. Sonra ise yılanın başlangıç konumu belirlenir ve oyun başlandığında yılanı hangi yöne doğru hareket edileceği belirlenir. Koddaki generate\_food fonksiyonu çalıştırılır bu da oluşturulacak elmanın konumu belirler ve skoru 0 olarak başlatır. Ardından ise statusbar öğesinin içeriğini değiştirmek için kod bloğu oluşturulur. StatusBar da kullanıcının topladığı puan yazdırılır.

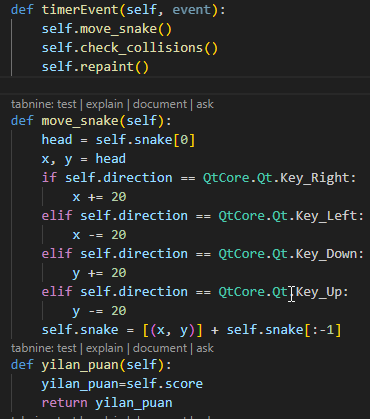
**Şekil 4.3 Kod**

Bu fonksiyonda while döngüsü açılır ve random fonksiyonuyla elmanın x ve y koordinatları ekrana bağlı olarak oluşturulur ve yılan elmayla temas etmiyorsa koordinatları fonksiyona geri döndürülür.



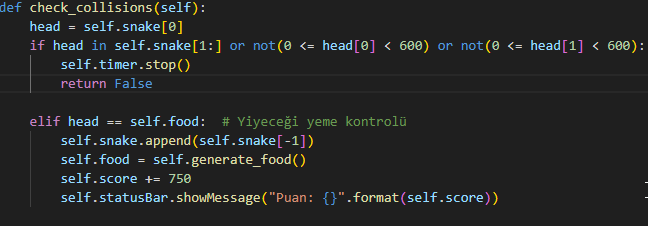
**Şekil 4.6 Kod**

Bu fonksiyon, painter adında bir değişken tanımlar ve bu değişkene bir QPainter nesnesi oluşturur. Bu nesne, elmanın ekranda gösterilmesini sağlar. Painter.fillRect, tüm ekranı belirlenen renge boyar. setBrush ise yılanın rengini belirlenen renge göre ayarlar. for döngüsü, yılanın her parçasının ne kadar büyük olduğunu belirlemek için kullanılır. Devamında, elmanın boyutunu ve rengini ayarlamak için kullanılan kodlar yer almaktadır.

timeEvent fonksiyonunda çağırıldığında move\_snake, check\_collisions ve repaint fonksiyonlarını çalıştıran fonksiyondur.

Mave\_snake fonksiyonun içinde yılanın başını hedef alan kod yazılır. Xve Y değişkeninlerin kafanın değerine atar. İf koşullarına göre hangi yön butonuna basıldığına göre yılanın gideceği yönü değiştiren kodlardır ve sonrasında ise yılanın yeni koordinatlarını güncelleyen kod satırı yazılır.

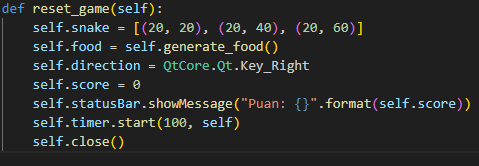
**Şekil 4.7 Kod**

Yılan\_puan fonksiyonu ise dosyada bulunan score değişkenini yılan\_puan değişkenine atayıp bu değişkeni main.py dosyasında kullanmak için değer döndüren fonksiyondur.

**Şekil 4.8 Kod**

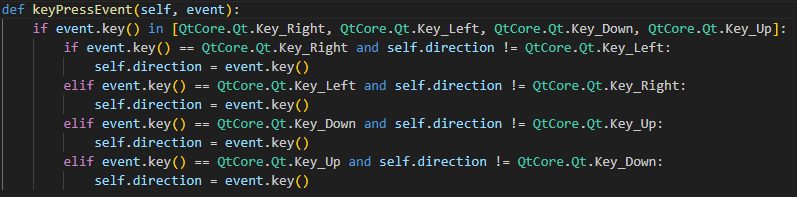
Bu fonksiyonda yılanın kafasını head adlı değişkene atanır ve eğer head değişkeni kendisiyle çarpışıyorsa ya da ekran sınırına değiyorsa zaman sayacını durdurur ve fonksiyona false değerini döndürür böylece oyun bitmış olur.

Bir diğer if koşulunda ise head değeri elma objesine değiyorsa yeni elma üretilen kodu çalıştırır,yılanın boyu uzatılır ve puan eklenen kod çalıştırılır kodun sonunda statusbar içeriğini güncellenir.



**Şekil 4.9 Kod**

Şekil 4.9 da fonksiyonda oyun kaybedilince çalışacak olan yılanın koordinatları başlangıçtaki konumu olacak şekilde yeniden belirlenir. Ardından, yeni bir yem üretme işlevi çağrılır. Yılanın ilk hareket ettiği yöne yönlendirilmesi sağlanır. Skor sıfırlanır, durum çubuğu içeriği temizlenir ve ekranda kısa bir süre sonra kapanma işlemi gerçekleştirilir.

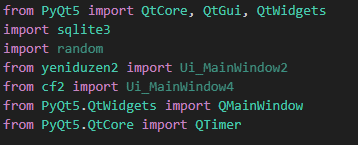


**Şekil 5.0 Kod**

Bu fonksiyonda yılanın gittiği yönün tam tersine gitmesini engelleyen ve bunu if koşuluyla sağlanan kod bloğudur. Ayrıca gitmek istediği yön tuşlarına basıldığında o yönlere doğru hareket etmesini sağlar.

# 2.7 carkifelek.py

Çarkıfelek dosyasında yapılan işlemler cf2.py dosyası ile birlikte çalışır. Kod rastgele gelen soruları, cf2.py dosyasından çekilen puanlarla birlikte kullanılır. cf2.py dosyasıyla gelen puanı veri tabanından gelen soruları doğru cevap verilerek kazanılır. Kullanıcının cf2.py dosyasında çark çevirerek kullanıcının kendisinin etki ettiği hissi oluşur.

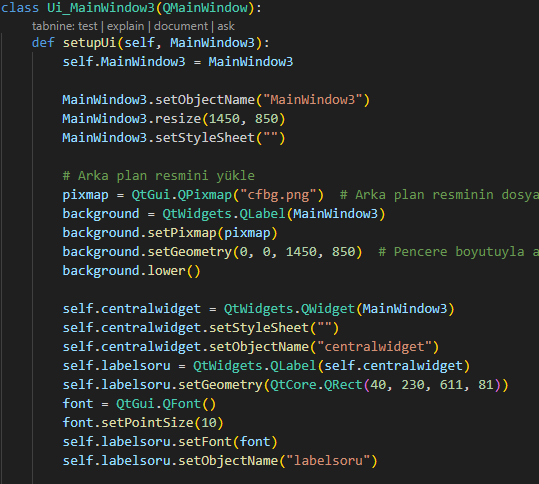


**Şekil 5.1 Kod**

Bu İmport kodları dosyada kullanılacak olan kodların kullanılması için gerekli kütüphaneleri kurulumu için kullanılır. sqlite3 import kodu sqlite veri tabanından çekilecek sorular için kullanılan kodların çalışması için kullanılır.

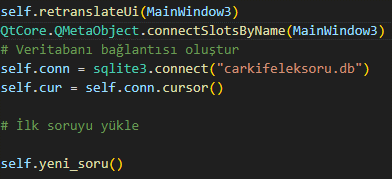
Random import kodu ise belirli değer aralığında rastgele sayı üreten kodlar için kullanılan kütüphanedir.

Qtimer import kodu ise kodların zamanlama ile çalışmasını sağlayan fonksiyonları içeren kütüphaneyi dosyaya tanıtan koddur.

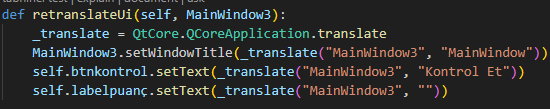
** Şekil 5.2 Kod Şekil 5.3 Kod**

Şekil 5.2 ve Şekil 5.3bloklarında UI\_MainWindow3 adında bir sınıf tanımlandı ve setupUI adında bir fonksiyon oluşturuldu. SetupUI fonksiyonu kullanıcının ekranda gördüğü kısmın özelliklerini belirlemek için kullanılan bölümdür.

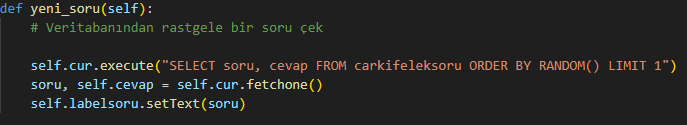
Dosyada bulunan label öğesinin pixmap özelliğini kullanarak arkaplan için resim koyulmasını sağlar ve ekranda gözüken diğer öğelerin adlarını, yazı stillerini, fontlarını, renklerini, nerede durması gerektiğini ve konumunu belirleyen kodlardır.

Bu fonksiyon bloğu ekranın güncellemesini sağlar. Sinyalleri birbirine bağlayan kodlar burada yazılır. Bu fonksiyonda carkifeleksoru veri tabanına bağlanmak için bu fonksiyonun içi kullanılır.

**Şekil 5.4 Kod**

Cursor ise veri tabanını düzenlemeyi sağlar ve sonrasında yeni soru fonksiyonunu çalıştırır.

**Şekil 5.5 Kod**

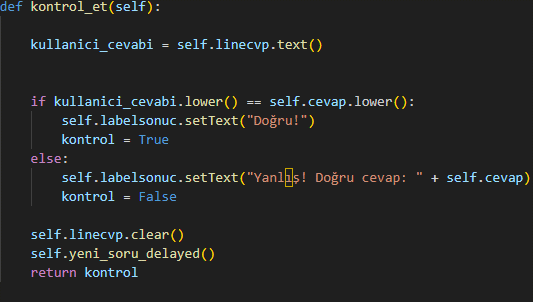
retranslateUI fonksiyonu ekranda gözüken öğelerin içeriğin yazıldığı bölümdür. Ekran adını belirleyen yapı bun fonksiyondadır.

**Şekil 5.6 Kod**

Veri tabanından rastgele soru ve onun cevabını çekilmesi için kod kullanılır. Devamında soru ve cevabı başka bir değişkene atılır. Label içeriğine soru yazdırılan kod kullanılır. Kısaca veri tabanından soru ve cevap çekilir ve bir değişkene atılır soru değişkeni ise label text özeliğine eklenir.

Bu fonksiyonda yeni soru fonksiyonu çalıştırılırken 5 saniye geciktirir.

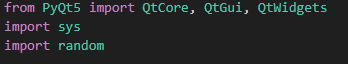
**Şekil 5.7 Kod**

Bu fonksiyonda ekranda kullanıcının verdiği cevap bir değişkene atılır eğer kullanıcının verdiği cevap veri tabanından çekilen cevapla aynı ise ekranda bulunan label öğesinin text kısmına “Doğru” yazısı yazdırılır.

**Şekil 5.8 Kod**

Değilse “Yanlış” yazdırılır ve doğru ya da yanlış olmasına göre True ya da False değeri fonksiyona döndürülür. Bu kodların çalışmasından sonra cevap yeri sıfırlanır ve soru değiştiren kod çalıştırılır.

# 2.8 cf2.py

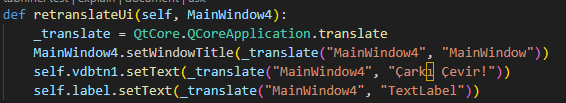


**Şekil 5.9 Kod**

Bu import kodları dosyada çalışacak kodların çalışması için gerekli olan kütüphaneleri dosyaya tanıtmak için kullanılır. Sys kütüphanesi Python kodlarının çalışması durması ve başka işlemler için kullanılan fonksiyonlar sys kütüphanesinden kullanılır. Random kütüphanesi belirli değerler arasında rastgele bir değer seçmek için kullanılır.

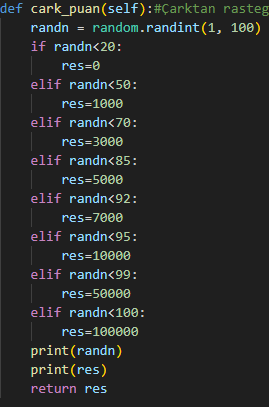
Bu kodda sınıf oluşturulmuş ve sınıfın adı UI\_MainWindow4 olarak tanımlanmıştır. Sınıfın içinde setupUI adında bir fonksiyon yazıldı. Burada ekrandaki öğelerin içeriğini değiştiren birtakım kodlar kullanılır. Fonksiyonun sonunda belirli bir butona tıklandığında çalışacak olan kod satırı kullanılır. Bu kod bloğunda gif belgesi dosyaya tanıtılır ve kullanılması için bir değişkene atanır ayrıca bu kod bloğunda retranslateUı fonksiyonu çalıştırılır.

**Şekil 6.0 Kod**



**Şekil 6.1 Kod**

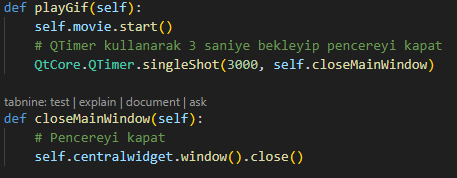
Bu kod bloğunda ekranda bulunan öğelerin içindeki yazıların ve özeliklerin belirlendiği kod alanıdır.

Bu fonksiyonda random kütüphanesini kullanarak 1 ile 100 arasında rastgele sayı üretiliyor,üretilen rastgele sayı bir değişkene atanır.

Değişkenin değerine göre hangi aralıktaysa ona göre bir puan belirlenir ve sonrasında değer return kodu iel fonksiyona geri gönderilir.

Eğer 20 altı değer döndürülürse 0 olarak belirlenir bu oyuncunun puanlarını sıfırlamak için kullanılır.

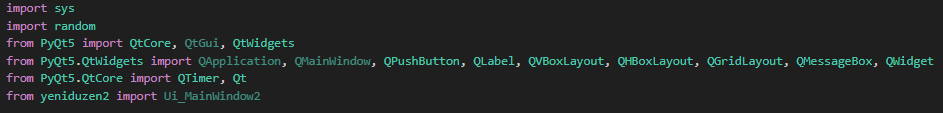
**Şekil 6.2 Kod**

Bu kod bloğunda playgif fonksiyonu setupUI fonksiyonunda tanıtılan GIF’ı 3 saniye boyunca çalıştıracak kod bloğudur.

**Şekil 6.4 Kod**

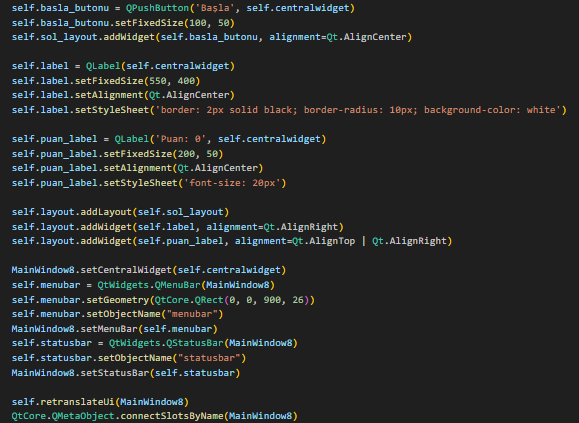
closeMainWindow fonksiyonu ise ekranda açık olan pencereyi kapatmak için kullanılır.

# 2.9 renkli.py



**Şekil 6.5 Kod**

Python'un sys modülü ile PyQt5 modüllerinden gerekli olanları içeri aktarıyor. Ardından, PyQt5 modülünden QtCore, QtGui ve QtWidgets alt modüllerini ve QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout, QHBoxLayout, QGridLayout, QMessageBox, QWidget gibi sınıfları içeri aktarıyor. Ayrıca, Qt alt modülünden QTimer ve Qt sabitlerini içeri aktarıyor.

****

**Şekil 6.6 Kod**

`MainWindow8.setObjectName("MainWindow8")` ve `MainWindow8.resize(900, 600)`: Ana pencerenin adını belirler ve boyutunu ayarlar.

`self.centralwidget = QtWidgets.QWidget(MainWindow8)`: Ana pencerenin merkezi widget'ını oluşturur. Bu, diğer arayüz öğelerinin yerleştirileceği alanı temsil eder.

`self.layout = QHBoxLayout(self.centralwidget)`: Ana pencerenin merkezi widget'ı üzerinde bir yatay düzen oluşturur. Bu, diğer düzenleri yatay olarak düzenlemek için kullanılacak bir konteynerdir.

`self.sol\_layout = QVBoxLayout()`: Sol taraftaki düğmeler için dikey bir düzen oluşturur. Bu, sol taraftaki düğmelerin birbirinin altında düzenlenmesini sağlar.

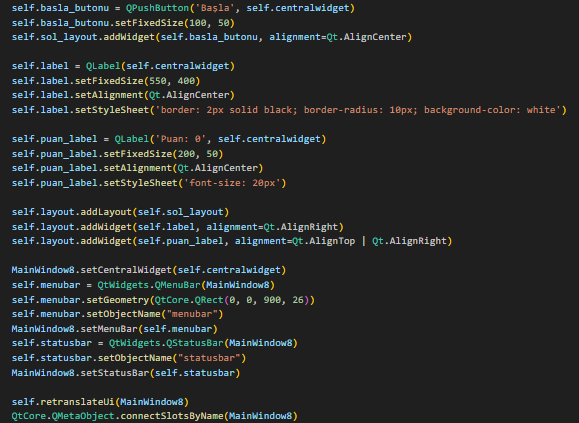
`self.buton\_layout = QGridLayout()`: Dört renk düğmesini içeren bir ızgara düzeni oluşturur. Bu, düğmelerin bir ızgara şeklinde düzenlenmesini sağlar.

`spacer = QWidget(self.centralwidget)`: Düğmelerin sol üst köşesindeki boşluğu eklemek için bir boş widget oluşturur.

`spacer.setFixedSize(200, 150)`: Boşluğun boyutunu ayarlar.

Dört renk düğmesini oluşturur ve düğmelere arka plan rengi verir.

`self.sol\_layout.addLayout(self.buton\_layout)`: Sol taraftaki düğmelerin ızgara düzenini ana düzene ekler. Bu, sol taraftaki düğmelerin ana pencerenin merkezi widget'ı içindeki düzene yerleştirilmesini sağlar. Bu kodların hepsinde PyQt5 kullanarak bir pencere oluşturur ve bu pencerede dört renk düğmesi içeren bir arayüz düzeni oluşturur.



**Şekil 6.7 Kod**

`self.basla\_butonu = QPushButton('Başla', self.centralwidget)`: "Başla" metniyle bir başlatma düğmesi oluşturur.

`self.basla\_butonu.setFixedSize(100, 50)`: Başlatma düğmesinin boyutunu ayarlar.

`self.sol\_layout.addWidget(self.basla\_butonu, alignment=Qt.AlignCenter)`: Başlatma düğmesini sol taraftaki düğme düzenine ekler ve bu düğmeyi düzenin ortasına hizalar.

`self.label = QLabel(self.centralwidget)`: Etiket (label) oluşturur. Bu etiket, oyun alanını temsil eder.

`self.label.setFixedSize(550, 400)`: Etiketin boyutunu ayarlar.

`self.label.setAlignment(Qt.AlignCenter)`: Etiketin metnini ortalamak için hizalama ayarı yapar.

`self.label.setStyleSheet('border: 2px solid black; border-radius: 10px; background-color: white')`: Etiketin görünümünü şekillendirir. Kenarlık rengi, köşelerin yuvarlatılması ve arka plan rengi gibi özellikler belirlenir.

`self.puan\_label = QLabel('Puan: 0', self.centralwidget)`: Puanı göstermek için bir etiket oluşturur ve başlangıçta 0 puanı gösterir.

`self.puan\_label.setFixedSize(200, 50)`: Puan etiketinin boyutunu ayarlar.

`self.puan\_label.setAlignment(Qt.AlignCenter)`: Puan etiketinin metnini ortalamak için hizalama ayarı yapar.

`self.puan\_label.setStyleSheet('font-size: 20px')`: Puan etiketinin metin boyutunu ayarlar.

`self.layout.addLayout(self.sol\_layout)`: Sol taraftaki düğme düzenini ana düzene ekler.

`self.layout.addWidget(self.label, alignment=Qt.AlignRight)`: Oyun alanı etiketini ana düzene ekler ve sağa hizalar.

`self.layout.addWidget(self.puan\_label, alignment=Qt.AlignTop | Qt.AlignRight)`: Puan etiketini ana düzene ekler ve yukarı sağ köşeye hizalar.

`MainWindow8.setCentralWidget(self.centralwidget)`: Ana pencerenin merkezi widget'ını ayarlar.

`self.menubar = QtWidgets.QMenuBar(MainWindow8)`: Menü çubuğunu oluşturur.

`self.menubar.setGeometry(QtCore.QRect(0, 0, 900, 26))`: Menü çubuğunun boyutunu ayarlar.

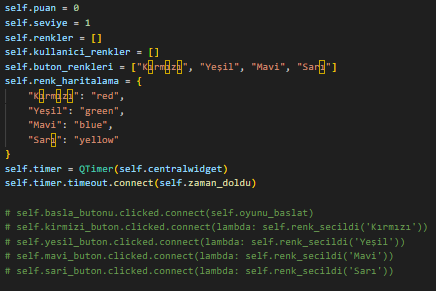
`MainWindow8.setMenuBar(self.menubar)`: Ana pencerenin menü çubuğunu ayarlar.

`self.statusbar = QtWidgets.QStatusBar(MainWindow8)`: Durum çubuğunu oluşturur.

`MainWindow8.setStatusBar(self.statusbar)`: Ana pencerenin durum çubuğunu ayarlar.

`self.retranslateUi(MainWindow8)`: Arayüz öğelerinin metinlerini yeniden çevirir.

`QtCore.QMetaObject.connectSlotsByName(MainWindow8)`: PyQt5'in otomatik bağlama mekanizmasını kullanarak, belirli bir nesnenin sinyallerini belirli slotlara bağlar. Bu kod, belirli sinyal-slot bağlantılarını otomatik olarak yapar. Bu genellikle Qt Designer ile oluşturulan arayüzlerde kullanılır.



**Şekil 6.8 Kod**

Şekil 6.8 renklerin yönetimi ve oyun mekanizmasını tanımlar.

`self.puan = 0`: Oyuncunun puanını saklar ve başlangıçta 0 olarak ayarlanır.

`self.seviye = 1`: Oyun seviyesini saklar ve başlangıçta 1 olarak ayarlanır.

`self.renkler = []`: Oyunun oluşturduğu renk dizisini saklar.

`self.kullanici\_renkler = []`: Oyuncunun seçtiği renkleri saklar.

`self.buton\_renkleri = ["Kırmızı", "Yeşil", "Mavi", "Sarı"]`: Oyun düğmelerinin renklerini saklar.

`self.renk\_haritalama`: Renk adları ile gerçek renkler arasındaki eşleştirmeyi sağlar.

`self.timer = QTimer(self.centralwidget)`: Bir QTimer nesnesi oluşturur. Bu, bir zamanlayıcı kullanarak belirli aralıklarla belirli işlevleri çağırmak için kullanılır.

`self.timer.timeout.connect(self.zaman\_doldu)`: Zamanlayıcı zaman aşımına ulaştığında çağrılacak işlevi belirler.

Yorum satırlarına alınmış kodlar, düğmelerin tıklanma olaylarına bağlanmış işlevleri tanımlar. Bu işlevler, oyunu başlatmayı ve renk seçimlerini işlemeyi sağlar. Yorum satırına alınmalarının sebebi main.py dosyasından başlatma gereği duyulduğundan kaynaklıdır.

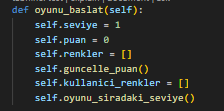


**Şekil 6.9 Kod**

`def retranslateUi(self, MainWindow8):`: Arayüzdeki metinlerin çevirisini gerçekleştirecek olan işlevi tanımlar. Bu işlev, ana pencere nesnesini ve çeviri işlemleri için kullanılacak çeviri işlevini alır.

`\_translate = QtCore.QCoreApplication.translate`: Çeviri işlemleri için bir `\_translate` işlevi oluşturur. Bu işlev, metinlerin çevirilmesi için PyQt5'in sağladığı çeviri işlevini kullanır.

`MainWindow8.setWindowTitle(\_translate("MainWindow8", "Renk Hafıza Oyunu"))`: Ana pencerenin başlık metnini çevirir. "Renk Hafıza Oyunu" metnini çevirir ve pencerenin başlığı olarak ayarlar.

PyQt5 arayüzündeki metinlerin çevirisini gerçekleştirir ve ana pencerenin başlığını "Renk Hafıza Oyunu" olarak ayarlar.

`self.seviye = 1`: Oyun seviyesini sıfırlar ve başlangıç seviyesini belirler.

`self.puan = 0`: Oyuncunun puanını sıfırlar ve başlangıç puanını belirler.

**Şekil 7.0 Kod**

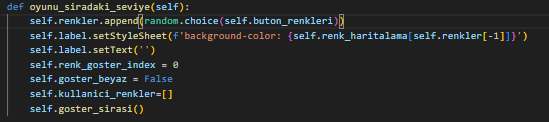
`self.renkler = []`: Oyunun oluşturduğu renk dizisini temizler.

`self.guncelle\_puan()`: Oyuncu puanını güncellemek için `guncelle\_puan()` işlevini çağırır.

`self.kullanici\_renkler = []`: Oyuncunun seçtiği renkleri saklayan listeyi temizler.

`self.oyunu\_siradaki\_seviye()`: Oyunu bir sonraki seviyeye geçirmek için `oyunu\_siradaki\_seviye()` işlevini çağırır.

Oyunun başlatılması için gerekli ayarları yapar ve bir sonraki seviyeye geçer.



**Şekil 7.1 Kod**

Bu fonk. oyunun bir sonraki seviyesine geçilmesi için gerekli ayarları yapar ve oyuncuya yeni renkleri gösterir.

`self.renkler.append(random.choice(self.buton\_renkleri))`: Bir sonraki seviyede gösterilecek renkleri belirlemek için `buton\_renkleri` listesinden rasgele bir renk seçer ve bu rengi `renkler` listesine ekler.

`self.label.setStyleSheet(f'background-color: {self.renk\_haritalama[self.renkler[-1]]}')`: Oyuncuya gösterilecek olan rengi, seçilen renge uygun bir arka plan rengi olarak etiketin stil sayfasına uygular.

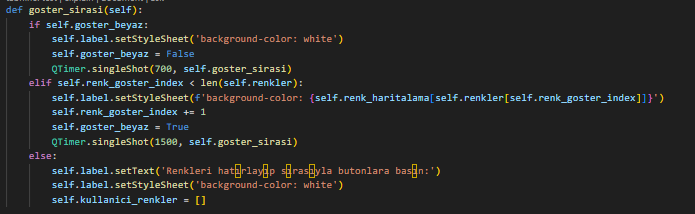
`self.label.setText('')`: Etiketin metnini temizler. Bu, önceki seviyede gösterilen rengin etikette kalan metnini kaldırır.

`self.renk\_goster\_index = 0`: Gösterilen renklerin indeksini sıfırlar. Bu, oyuncunun gösterilen renkleri doğru bir şekilde hatırlamak için hangi renk gösterildiğini takip etmesine yardımcı olur.

`self.goster\_beyaz = False`: Etiketin beyaz arka plan rengini gösterip göstermeyeceğini belirler. Başlangıçta bu değer False olarak ayarlanır.

`self.kullanici\_renkler=[]`: Oyuncunun seçtiği renkleri saklayan listeyi temizler. Bu, bir sonraki seviyeye geçildiğinde oyuncunun yeni renkleri doğru bir şekilde hatırlamasını sağlar.

`self.goster\_sirasi()`: Oyuncuya gösterilen renkleri ekranda göstermek için `goster\_sirasi()` işlevini çağırır.



**Şekil 7.2 Kod**

Bu işlev, oyuncuya gösterilen renkleri belirli bir sırayla ekranda gösterir. İşte bu kodun ne yaptığını adım adım açıklayalım:

a) `if self.goster\_beyaz:`: Eğer `goster\_beyaz` değeri True ise:

- `self.label.setStyleSheet('background-color: white')`: Etiketin arka plan rengini beyaza ayarlar.

- `self.goster\_beyaz = False`: `goster\_beyaz` değerini False olarak ayarlar.

- `QTimer.singleShot(700, self.goster\_sirasi)`: 700 milisaniye sonra `goster\_sirasi()` işlevini tekrar çağırır. Bu, belirli bir süre sonra renklerin gösterilmesine devam etmek için bir zamanlayıcı oluşturur.

b) `elif self.renk\_goster\_index < len(self.renkler)):`: Eğer gösterilmesi gereken renklerin indeksi, renkler listesinin uzunluğundan küçükse:

-`self.label.setStyleSheet(f'background-color:

{self.renk\_haritalama[self.renkler[self.renk\_goster\_index]]}')`: Etiketin arka plan rengini, gösterilen renge uygun olarak ayarlar.

- `self.renk\_goster\_index += 1`: Renk gösterme indeksini bir artırır.

- `self.goster\_beyaz = True`: `goster\_beyaz` değerini True olarak ayarlar. Bu, bir sonraki renk gösteriminde beyaz arka plan renginin gösterilmesi için gereklidir.

- `QTimer.singleShot(1500, self.goster\_sirasi)`: 1500 milisaniye sonra `goster\_sirasi()` işlevini tekrar çağırır. Bu, belirli bir süre sonra bir sonraki renk gösterimine geçmek için bir zamanlayıcı oluşturur.

c) `else:`:

- `self.label.setText('Renkleri hatırlayıp sırasıyla butonlara basın:')`: Etiketin metnini, oyuncuya renkleri hatırlaması ve sırasıyla butonlara basması gerektiğini belirten bir mesajla günceller.

- `self.label.setStyleSheet('background-color: white')`: Etiketin arka plan rengini beyaza ayarlar.

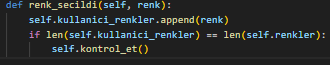
- `self.kullanici\_renkler = []`: Kullanıcının seçtiği renkleri saklayan listeyi temizler. Bu, oyuncunun yeni renkleri doğru bir şekilde hatırlamasını sağlar.

Oyuncuya gösterilen renkleri belirli bir sırayla ekranda gösterir ve ardından oyuncuya renkleri hatırlaması için bir mesaj gösterir.

self.timer.stop(): Zamanlayıcıyı durdurur. Bu, renklerin gösterilmesi veya oyuncunun seçim yapması sırasında zamanlayıcının devam etmesini engeller.

**Şekil 7.3 Kod**

self.kontrol\_et(): Oyuncunun renkleri hatırlayıp doğru sırayla butonlara basıp basmadığını kontrol etmek için kontrol\_et() işlevini çağırır. Bu işlev, oyuncunun seçimlerini kontrol eder ve uygun işlemi gerçekleştirir.

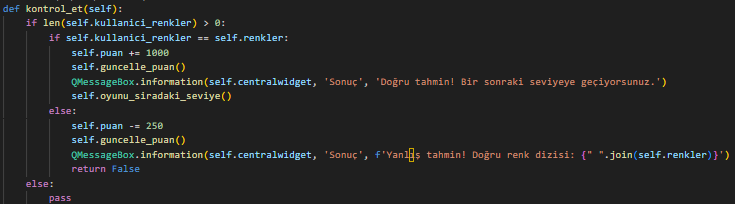
****

**Şekil 7.4 Kod**

Oyuncunun bir renk düğmesine tıkladığında çağrılır ve seçilen rengi kaydeder. `self.kullanici\_renkler.append(renk)`: Oyuncunun seçtiği rengi, `kullanici\_renkler` listesine ekler. Bu, oyuncunun seçimlerini takip etmek için kullanılır.

`if len(self.kullanici\_renkler) == len(self.renkler):`: Eğer oyuncu tüm renkleri seçmişse:

- `self.kontrol\_et()`: Oyuncunun seçimlerini kontrol etmek için `kontrol\_et()` işlevini çağırır.



**Şekil 7.5 Kod**

Bu fonksiyon oyuncunun renkleri doğru sırayla seçip seçmediğini kontrol eder.

a)`if len(self.kullanici\_renkler) > 0:`: Oyuncunun en az bir renk seçtiğini kontrol eder. Eğer oyuncu en az bir renk seçtiyse:

- `if self.kullanici\_renkler == self.renkler:`: Oyuncunun seçtiği renklerin, gösterilen renklerle aynı olup olmadığını kontrol eder. Eğer oyuncunun seçtiği renkler gösterilen renklerle aynı ise: - `self.puan += 1000`: Oyuncunun puanını 1000 artırır.

- `self.guncelle\_puan()`: Puanı güncellemek için `guncelle\_puan()` işlevini çağırır.

- `QMessageBox.information(self.centralwidget, 'Sonuç', 'Doğru tahmin! Bir sonraki seviyeye geçiyorsunuz.')`: Oyuncuya bilgi mesajı gösterir, doğru tahmin yaptığına dair bilgi verir ve bir sonraki seviyeye geçeceğini belirtir.

- `self.oyunu\_siradaki\_seviye()`: Bir sonraki seviyeye geçmek için `oyunu\_siradaki\_seviye()` işlevini çağırır.

- `else:`: Eğer oyuncunun seçtiği renkler gösterilen renklerle aynı değilse:

- `self.puan -= 250`: Oyuncunun puanını 250 azaltır.

- `self.guncelle\_puan()`: Puanı güncellemek için `guncelle\_puan()` işlevini çağırır.

- `QMessageBox.information(self.centralwidget, 'Sonuç', f'Yanlış tahmin! Doğru renk dizisi: {" ".join(self.renkler)}')`: Oyuncuya bilgi mesajı gösterir, yanlış tahmin yaptığına dair bilgi verir ve doğru renk dizisini gösterir.

- `return False`: Yanlış tahmin yapıldığını belirtmek için False değeri döndürür.

b) `else:`: Eğer oyuncu henüz herhangi bir renk seçmemişse:

- `pass`: İşlevi geçer. Herhangi bir işlem yapmaz.

Oyunun sonunda oyuncunun elde ettiği puanı döndürür.

**Şekil 7.6 Kod**

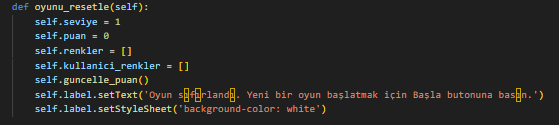
`sondeger = self.puan`: Oyunun sonunda elde edilen puanı `sondeger` adlı bir değişkene atar.

`return sondeger`: `sondeger` değişkenini döndürür. Bu, oyunun sonunda elde edilen puanın dışarıya aktarılmasını sağlar.

Oyuncunun puanını güncellemek için label ı günceller.

**Şekil 7.7 Kod**

`self.puan\_label.setText(f'Puan: {self.puan}')`: Arayüzdeki puan etiketinin metnini, oyuncunun güncel puanını içerecek şekilde ayarlar. Bu, oyuncunun puanının ekranda güncel olarak gösterilmesini sağlar.



**Şekil 7.8 Kod**

Oyunun sıfırlanması için gereken ayarları yapar.

`self.seviye = 1`: Oyun seviyesini sıfırlar ve başlangıç seviyesini belirler.

`self.puan = 0`: Oyuncunun puanını sıfırlar ve başlangıç puanını belirler.

`self.renkler = []`: Oyunun oluşturduğu renk dizisini temizler.

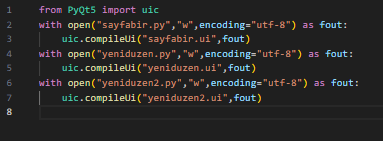
`self.kullanici\_renkler = []`: Oyuncunun seçtiği renkleri saklayan listeyi temizler.

`self.guncelle\_puan()`: Puanı güncellemek için `guncelle\_puan()` işlevini çağırır.

`self.label.setText('Oyun sıfırlandı. Yeni bir oyun başlatmak için Başla butonuna basın.')`: Etiketin metnini, oyunun sıfırlandığına dair bir mesajla günceller.

`self.label.setStyleSheet('background-color: white')`: Etiketin arka plan rengini beyaza ayarlar. Bu, mesajın daha iyi görünmesini sağlar.

# 3.0 convert.py



**Şekil 7.9 Kod**

Şekil 6.5 te PyQt5 kütüphanesinin uic modülünü kullanarak üç farklı .ui dosyasını (sayfabir.ui, yeniduzen.ui, yeniduzen2.ui) Python .py dosyalarına (sayfabir.py, yeniduzen.py, yeniduzen2.py) derlemek için kullanılır. Her bir with open bloğu, ilgili .ui dosyasını açarak, Python koduna dönüştürülmüş halini belirlenen .py dosyasına yazar ve bu sayede Qt Designer ile oluşturulan arayüzler PyQt5 ile kullanılabilir hale gelir.

# 3.1 imgler\_rc.py

PyQt5 uygulamalarında kullanılan kaynak dosyalarını (resource files) yönetmek için kullanılır. Genellikle bir Python betiğinin içine gömülü olarak gelir ve PyQt5 uygulamasının kaynak dosyalarının nasıl kullanılacağını tanımlar.

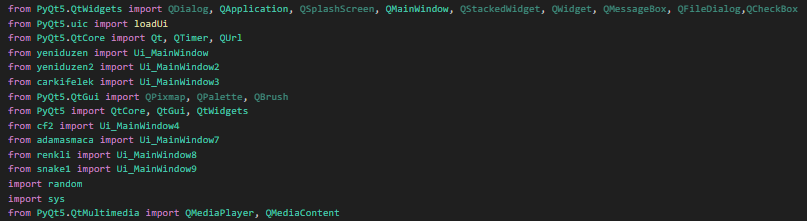
Öncelikle bir qt\_resource\_struct\_v2 bayt dizesi içerir. Bu dize, Qt uygulamasının kaynaklarını tanımlar ve içinde kullanılan veri yapılarını barındırır.

**Şekil 8.0 Kod**

Ardından, qt\_version kontrol edilir ve PyQt5'nin sürümüne göre uygun bir kaynak yapılandırması seçilir (rcc\_version ve qt\_resource\_struct).

Son olarak, qInitResources() fonksiyonu, kaynak verilerini PyQt5'ün QtCore.qRegisterResourceData() işlevine kaydeder, böylece PyQt5 uygulaması bu kaynaklara erişebilir hale gelir. Bu dosya PyQt5 uygulamalarının daha karmaşık kaynak dosyalarını yönetmesine yardımcı olur ve bu kaynaklara kolayca erişim sağlar. Bu, genellikle uygulamanın kullanıcı arayüzüne görsel stiller, simgeler, metinler ve diğer gömülü kaynaklar dahildir.

# 3.2 main.py



**Şekil 8.1 Kod**

Bu kodlar PyQt5 kütüphanesinden çeşitli modülleri ve sınıfları içe aktarıyor

`QDialog, QApplication, QSplashScreen, QMainWindow, QStackedWidget, QWidget, QMessageBox, QFileDialog, QCheckBox`: PyQt5'ün farklı widget'larına ve iletişim kutularına (dialog kutuları) erişim sağlayan sınıfları içe aktarır.

`loadUi`: PyQt5.uic modülünden loadUi işlevini içe aktarır. Bu işlev, UI dosyalarını yüklemek ve uygulamaya entegre etmek için kullanılır.

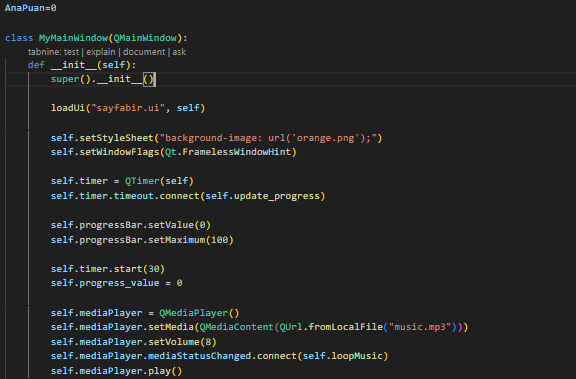
`Qt, QTimer, QUrl`: PyQt5.QtCore modülünden Qt, QTimer ve QUrl gibi çeşitli çekirdek sınıfları içe aktarır. Qt, PyQt5 uygulamalarının temelini oluşturan çeşitli sabitler içerir. QTimer, zamanlayıcılar oluşturmak için kullanılır. QUrl, URL'leri işlemek için kullanılır.

`Ui\_MainWindow, Ui\_MainWindow2, Ui\_MainWindow3, Ui\_MainWindow4, Ui\_MainWindow7, Ui\_MainWindow8, Ui\_MainWindow9`: PyQt5 uygulamasında kullanılacak olan farklı kullanıcı arayüzü (UI) sınıflarını içe aktarır. Bu sınıflar, Qt Designer veya Qt Creator gibi araçlarla tasarlanan UI dosyalarının Python kodlarına dönüştürülmesiyle oluşturulur.

`QPixmap, QPalette, QBrush`: PyQt5.QtGui modülünden QPixmap, QPalette ve QBrush gibi çeşitli grafik sınıflarını içe aktarır. QPixmap, bir görüntüyü temsil etmek için kullanılır. QPalette, arayüz öğelerinin renklerini ve arka planlarını ayarlamak için kullanılır. QBrush, fırça desenleri ve arka plan renkleri oluşturmak için kullanılır.

`random`: Python'ın random modülünü içe aktarır. Bu, rastgele sayılar üretmek ve liste öğelerini karıştırmak için kullanılır.

`sys`: Python'ın sys modülünü içe aktarır. Bu, sistemle ilgili işlevler sağlar ve özellikle komut satırı argümanlarına ve Python yürütme ortamına erişim sağlar.

`QMediaPlayer, QMediaContent`: PyQt5.QtMultimedia modülünden QMediaPlayer ve QMediaContent sınıflarını içe aktarır. Bu sınıflar, medya oynatma işlevselliğini sağlar ve ses ve video dosyalarını oynatmak için kullanılır.

**Şekil 8.2 Kod**

`AnaPuan=0`: `AnaPuan` adlı bir değişken oluşturur ve ona 0 değerini atar. Bu, oyunun ana puanını saklamak için kullanılır.

`class MyMainWindow(QMainWindow):`: `MyMainWindow` adında bir sınıf tanımlar ve bu sınıfı QMainWindow sınıfından türetir. Bu, PyQt5'de bir ana pencere oluşturur.

`def \_\_init\_\_(self):`: Sınıfın yapıcı metodunu tanımlar. Bu metod, sınıf bir örneği oluşturulduğunda otomatik olarak çağrılır.

`super().\_\_init\_\_()`: Miras alınan sınıfın yapıcı metodunu çağırır. Bu durumda, QMainWindow sınıfının yapıcı metodunu çağırır ve uygun şekilde başlatır.

`loadUi("sayfabir.ui", self)`: PyQt5.uic modülünden loadUi fonksiyonunu kullanarak "sayfabir.ui" adlı bir UI dosyasını yükler ve ana pencereye entegre eder. Bu, tasarım dosyasını kullanıcı arayüzüne dönüştürür ve uygun şekilde görüntüler.

`self.setStyleSheet("background-image: url('orange.png');")`: Ana pencerenin arka planını "orange.png" adlı bir görüntü ile ayarlar.

`self.setWindowFlags(Qt.FramelessWindowHint)`: Pencerenin çerçevesiz olmasını sağlar. Bu, pencerenin kenarlıksız ve başlıksız olmasını sağlar.

`self.timer = QTimer(self)`: Bir QTimer nesnesi oluşturur ve bu nesneyi ana pencereye bağlar.

`self.timer.timeout.connect(self.update\_progress)`: Zamanlayıcı zaman aşımı olayı (timeout) gerçekleştiğinde `update\_progress` adlı bir metodun çağrılmasını sağlar.

`self.progressBar.setValue(0)`: İlerleme çubuğunun başlangıç değerini 0 olarak ayarlar.

`self.progressBar.setMaximum(100)`: İlerleme çubuğunun maksimum değerini 100 olarak ayarlar.

`self.timer.start(30)`: Zamanlayıcıyı başlatır ve her 30 milisaniyede bir timeout olayını tetikler. Bu, ilerleme çubuğunun güncellenmesini sağlar.

`self.progress\_value = 0`: İlerleme çubuğunun değerini saklamak için bir değişken oluşturur ve bu değişkeni başlangıçta 0 olarak ayarlar.

`self.mediaPlayer = QMediaPlayer()`: Bir QMediaPlayer nesnesi oluşturur. Bu nesne, medya dosyalarını oynatmak için kullanılır.

`self.mediaPlayer.setMedia(QMediaContent(QUrl.fromLocalFile("music.mp3")))`: MediaPlayer'a "music.mp3" adlı bir medya dosyasını yükler. Bu, müziği oynatmak için kullanılır.

`self.mediaPlayer.setVolume(8)`: MediaPlayer'ın ses seviyesini %8 olarak ayarlar.

`self.mediaPlayer.mediaStatusChanged.connect(self.loopMusic)`: Medya durumu değiştiğinde `loopMusic` adlı bir metodun çağrılmasını sağlar.`self.mediaPlayer.play()`: MediaPlayer'ı başlatır ve müziği çalmaya başlar.

Bu kod parçası müziğin döngü içinde sürekli olarak çalınmasını sağlayan bir metod tanımlar.

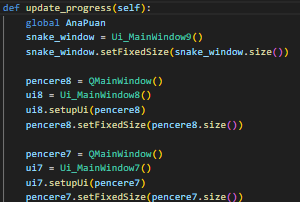
**Şekil 8.3 Kod**

`def loopMusic(self, status):`: `loopMusic` adında bir metod tanımlar ve bir durum parametresi alır. Bu metod, MediaPlayer'daki medyanın durumunu izler ve medya dosyası tamamlandığında müziği tekrar çalar.

`if status == QMediaPlayer.EndOfMedia:`: Eğer medya durumu, QMediaPlayer.EndOfMedia durumuna eşitse:

- `self.mediaPlayer.setPosition(0)`: Müziğin çalma konumunu başlangıç konumuna (0) ayarlar. Böylece müzik dosyası tekrar başa döner.

- `self.mediaPlayer.play()`: Müziği tekrar çalmaya başlar. Bu, müziğin tekrar baştan çalmasını sağlar.

Şekil 8.4 te ilerleme çubuğunun güncellenmesi ve belirli bir ilerleme seviyesine ulaşıldığında diğer pencereye geçiş yapılmasını sağlar. `global AnaPuan`: Ana puan değişkenini global bir değişken olarak tanımlar. Böylece bu değişkenin içeriği, bu metodun dışında da erişilebilir olur.

**Şekil 8.4 Kod**

`snake\_window = Ui\_MainWindow9()`: `Ui\_MainWindow9` sınıfından bir `snake\_window` örneği oluşturur. Bu, yılan oyununun ana penceresini temsil eder.

`snake\_window.setFixedSize(snake\_window.size())`: Yılan oyunu penceresinin boyutunu sabitler. Bu, kullanıcıların pencere boyutunu değiştirmesini engeller.

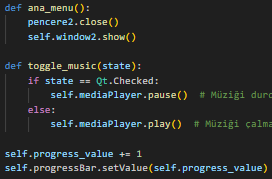
`pencere8 = QMainWindow()`: Yeni bir QMainWindow nesnesi oluşturur. Bu, renkli oyun penceresini temsil eder.

`ui8 = Ui\_MainWindow8()`: `Ui\_MainWindow8` sınıfından bir `ui8` örneği oluşturur. Bu, renkli oyun penceresinin kullanıcı arayüzünü temsil eder.

`ui8.setupUi(pencere8)`: `ui8` örneğini `pencere8` nesnesine bağlar. Bu, renkli oyun penceresinin kullanıcı arayüzünü pencereye entegre eder.

`pencere8.setFixedSize(pencere8.size())`: Renkli oyun penceresinin boyutunu sabitler. Bu, kullanıcıların pencere boyutunu değiştirmesini engeller.

Benzer adımları, adam asmaca oyunu (`pencere7`) için de gerçekleştirir.

Bu kodlar birkaç farklı işlevi içerir. Adım adım açıklayalım;

`ana\_menu()`: Bu metod, ana menüye dönme işlevini gerçekleştirir. Ana menüye dönüldüğünde, mevcut pencereyi kapatır ve ikinci pencereyi gösterir.

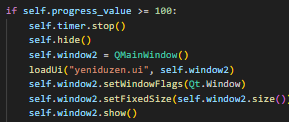
**Şekil 8.5 Kod**

`toggle\_music(state)`: Bu metod, müzik çalma durumunu değiştirir. CheckBox'ta bir değişiklik olduğunda çağrılır ve duruma göre müziği duraklatır veya çalmaya devam ettirir.

`self.progress\_value += 1`: İlerleme çubuğunun değerini artırır. Bu, ilerleme çubuğunun ilerlemesini simüle eder.

`self.progressBar.setValue(self.progress\_value)`: İlerleme çubuğunun değerini günceller ve görsel olarak ilerlemeyi gösterir.

Bu 2 fonksiyon ve sonraki kodlar ilerleme çubuğunun güncellenmesini ve müzik çalma durumunun yönetilmesini sağlar. Ayrıca ana menüye dönme işlevini de içerir.

İlerleme çubuğunun değerinin 100 veya daha büyük olması durumunda belirli işlemleri gerçekleştirir.

**Şekil 8.6 Kod**

`if self.progress\_value >= 100:`: İlerleme çubuğunun değeri 100 veya daha büyükse:

-`self.timer.stop()`: Zamanlayıcıyı durdurur. Bu, ilerleme çubuğunun daha fazla güncellenmemesini sağlar.

- `self.hide()`: Mevcut pencereyi gizler. Bu, mevcut pencerenin kullanıcının görüş alanından çıkmasını sağlar.

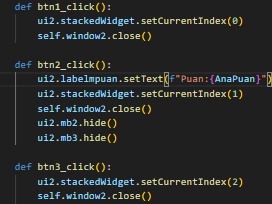
- `self.window2 = QMainWindow()`: Yeni bir QMainWindow nesnesi oluşturur. Bu, yeni pencereyi temsil eder.

- `loadUi("yeniduzen.ui", self.window2)`: "yeniduzen.ui" adlı bir tasarım dosyasını yükler ve bu dosyayı yeni pencereye entegre eder.

- `self.window2.setWindowFlags(Qt.Window)`: Yeni pencerenin bir pencere olmasını sağlar. Bu, pencerenin özelliklerini ayarlar ve çerçeve, başlık vb. öğeleri gösterir.

- `self.window2.setFixedSize(self.window2.size())`: Yeni pencerenin boyutunu sabitler. Bu, kullanıcıların pencere boyutunu değiştirmesini engeller.

- `self.window2.show()`: Yeni pencereyi gösterir. Bu, kullanıcıya yeni pencereyi görüntüler.

Üç farklı butona tıklama içerir. Her biri, farklı bir pencere indeksine geçişi ve mevcut pencerenin kapatılmasını sağlar.

`btn1\_click()`: Bu metod, birinci düğmeye tıklama işlevini gerçekleştirir. İkinci pencereye geçiş yapar ve `setCurrentIndex()` metoduyla belirtilen indekse (0) geçiş yapar. Sonra mevcut pencereyi kapatır. **Şekil 8.7 Kod**

`btn2\_click()`: Bu metod, ikinci düğmeye tıklama işlevini gerçekleştirir. İkinci pencereye geçiş yapar, `setCurrentIndex()` metoduyla belirtilen indekse (1) geçiş yapar ve AnaPuan'ı ekrana yazdırır. Ardından mevcut pencereyi kapatır ve ikinci penceredeki belirli öğeleri gizler.

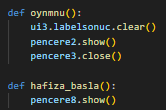
`btn3\_click()`: Bu metod, üçüncü düğmeye tıklama işlevini gerçekleştirir. İkinci pencereye geçiş yapar ve `setCurrentIndex()` metoduyla belirtilen indekse (2) geçiş yapar. Ardından mevcut pencereyi kapatır.



**Şekil 8.8 Kod Şekil 8.9 Kod**

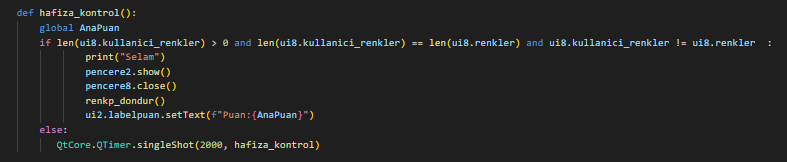
`cf\_click()`: Bu fonksiyon, "carkifelek" adındaki bir pencereyi gösterir. `QtCore.QTimer.singleShot(3000, pencere3.show)` kodu, 3 saniye gecikmeyle `pencere3.show()` işlevini çağırır ve bu da "carkifelek" penceresini gösterir.

`pencere2kapa()`: Bu fonksiyon, `pencere2` adındaki pencereyi kapatır.

`oynmnu()`: Bu fonksiyon, oyun menüsüne geri dönmek için kullanılır. İlk olarak, `ui3.labelsonuc` etiketini temizler. Daha sonra, `pencere2`'yi gösterir ve `pencere3`'ü kapatır.

`hafiza\_basla()`: Bu fonksiyon, hafıza oyununu başlatmak için kullanılır. `pencere8`'i gösterir.

**Şekil 9.0 Kod**



**Şekil 9.1 Kod**

Hafıza oyununun sonuçlarını kontrol etmek için bir fonksiyon içerir.

`global AnaPuan`: Ana puan değişkenini global bir değişken olarak tanımlar. Böylece bu değişkenin içeriği, bu fonksiyonun dışında da erişilebilir olur.

`if len(ui8.kullanici\_renkler) > 0 and len(ui8.kullanici\_renkler) == len(ui8.renkler) and ui8.kullanici\_renkler != ui8.renkler:`: Kullanıcı renklerinin uzunluğu 0'dan büyük olmalıdır ve kullanıcı renklerinin uzunluğu, oyunun renklerinin uzunluğuna eşit olmalıdır.

Ayrıca, kullanıcı renkleri, oyun renklerine eşit olmamalıdır. Bu koşullar sağlandığında:

- `"Selam"` yazdırılır.

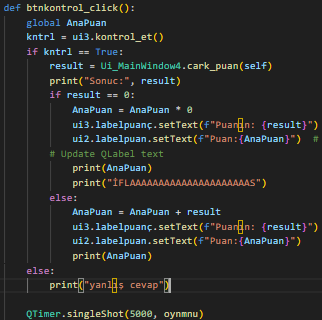
- `pencere2.show()`: İkinci pencereyi gösterir.

- `pencere8.close()`: Sekizinci pencereyi kapatır.

- `renkp\_dondur()`: `renkp\_dondur` adlı bir fonksiyon çağırılır.

- `ui2.labelpuan.setText(f"Puan:{AnaPuan}")`: Kullanıcının puanını içeren etiketi günceller.

Aksi durumda, `QtCore.QTimer.singleShot(2000, hafiza\_kontrol)` kodu, 2 saniyelik bir gecikme sonrasında `hafiza\_kontrol` fonksiyonunu tekrar çağırır. Bu, kullanıcının doğru renk dizisini tamamlamasını bekler.

Bu kod parçası, bir butona tıklama işlevini içerir.

`global AnaPuan`: Ana puan değişkenini global bir değişken olarak tanımlar. Böylece bu değişkenin içeriği, bu fonksiyonun dışında da erişilebilir olur.

`kntrl = ui3.kontrol\_et()`: `ui3` nesnesinin `kontrol\_et()` metodunu çağırır ve dönüş değerini `kntrl` değişkenine atar. Bu, kullanıcının verdiği cevabı kontrol eder.

**Şekil 9.2 Kod**

`if kntrl == True:`: Eğer kullanıcının verdiği cevap doğruysa:

- `result = Ui\_MainWindow4.cark\_puan(self)`: `Ui\_MainWindow4` sınıfının `cark\_puan` metodunu çağırır ve dönüş değerini `result` değişkenine atar. Bu, tekerleğin puanını alır.

- `if result == 0:`: Eğer tekerleğin döndüğü yere göre puan 0 ise:

- `AnaPuan = AnaPuan \* 0`: Ana puanı sıfırlar. - `ui3.labelpuanç.setText(f"Puanın: {result}")`: `ui3` nesnesinin `labelpuanç` etiketinin metnini günceller.

- `ui2.labelpuan.setText(f"Puan:{AnaPuan}")`: `ui2` nesnesinin `labelpuan` etiketinin metnini günceller.

Aksi durumda:

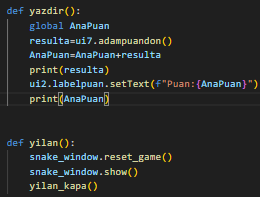
- `AnaPuan = AnaPuan + result`: Ana puanı tekerleğin puanıyla günceller.

- `ui3.labelpuanç.setText(f"Puanın: {result}")`: `ui3` nesnesinin `labelpuanç` etiketinin metnini günceller.

- `ui2.labelpuan.setText(f"Puan:{AnaPuan}")`: `ui2` nesnesinin `labelpuan` etiketinin metnini günceller.

`else:`: Eğer kullanıcının verdiği cevap yanlışsa, "yanlış cevap" yazdırır.

`QTimer.singleShot(5000, oynmnu)`: 5 saniye gecikme sonrasında `oynmnu` fonksiyonunu çağırır. Bu, oyun menüsüne geri döner.

`yazdir()`: Bu fonksiyon, bir değeri ekrana yazdırır.

`global AnaPuan`: Ana puan değişkenini global bir değişken olarak tanımlar. Böylece bu değişkenin içeriği, bu fonksiyonun dışında da erişilebilir olur.

**Şekil 9.3 Kod**

`resulta=ui7.adampuandon()`: `ui7` nesnesinin `adampuandon()` metodunu çağırır ve dönüş değerini `resulta` değişkenine atar. Bu, bir puan değeri alır.

`AnaPuan=AnaPuan+resulta`: Ana puanı, aldığı puan değeriyle günceller.

`print(resulta)`: Aldığı puanı konsola yazdırır.

- `ui2.labelpuan.setText(f"Puan:{AnaPuan}")`: `ui2` nesnesinin `labelpuan` etiketinin metnini günceller. Bu, kullanıcının puanını ekrana yazdırır.

- `print(AnaPuan)`: Güncellenmiş ana puanı konsola yazdırır.

`yilan()`: Bu fonksiyon, yılan oyununu başlatır ve mevcut pencereyi kapatır. İşlevi adım adım açıklayalım:

- `snake\_window.reset\_game()`: Yılan oyununu sıfırlar.

- `snake\_window.show()`: Yılan oyununu gösterir.

- `yilan\_kapa()`: `yilan\_kapa` adlı bir fonksiyonu çağırır.



**Şekil 9.4 Kod**

Bu kod parçası yılan oyununun sonuçlarını kontrol etmek için bir fonksiyon içerir.

global AnaPuan`: Ana puan değişkenini global bir değişken olarak tanımlar. Böylece bu değişkenin içeriği, bu fonksiyonun dışında da erişilebilir olur.

`if snake\_window.check\_collisions() == False:`: Eğer yılanın çarpışma kontrolü `False` döndürürse:

- `pencere2.show()`: İkinci pencereyi gösterir.

- `mesaj\_kutusu = QtWidgets.QMessageBox(self)`: Bir mesaj kutusu oluşturur.

- `mesaj\_kutusu.setWindowTitle('Oyun Bitti')`: Mesaj kutusunun başlığını ayarlar.

-`mesaj\_kutusu.setText(f'OYUNU KAYBETTİN !\nPuanınız: {snake\_window.score}')`: Mesaj kutusunun metnini ayarlar. Bu, oyunun sonucunu ve oyuncunun puanını içerir.

- `mesaj\_kutusu.setFont(font)`: Mesaj kutusunun yazı tipini ayarlar.

- `mesaj\_kutusu.setStyleSheet("color: white;")`: Mesaj kutusunun stilini ayarlar.

- `mesaj\_kutusu.exec\_()`: Mesaj kutusunu görüntüler.

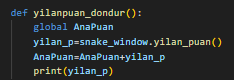
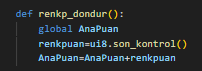
- `print("Gelio")`: "Gelio" metnini konsola yazdırır.

- `yilanpuan\_dondur()`: `yilanpuan\_dondur` adlı bir fonksiyonu çağırır.

- `snake\_window.close()`: Yılan oyununu kapatır.

- `ui2.labelpuan.setText(f"Puan:{AnaPuan}")`: `ui2` nesnesinin `labelpuan` etiketinin metnini günceller. Bu, oyuncunun puanını ekrana yazdırır.

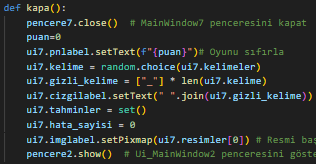
Aksi halde; `QTimer.singleShot(1000, yilan\_kapa)`: 1 saniyelik bir gecikme sonrasında `yilan\_kapa` fonksiyonunu tekrar çağırır. Bu, yılanın çarpışma kontrolünü yeniden kontrol etmek için bir döngü oluşturur.



**Şekil 9.5 Kod**  **Şekil 9.6 Kod**

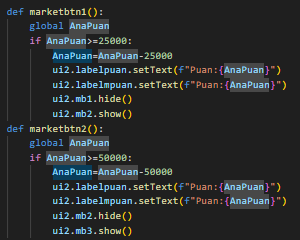
yilanpuan\_dondur() Amaç: Yılan oyununda kazanılan puanı global AnaPuan değişkenine eklemek. global AnaPuan: Global AnaPuan değişkenini kullanır. yilan\_p = snake\_window.yilan\_puan(): Yılan oyununda kazanılan puanı yilan\_p değişkenine atar. AnaPuan = AnaPuan + yilan\_p: Yılan oyununda kazanılan puanı AnaPuan'a ekler. print(yilan\_p): Yılan oyununda kazanılan puanı konsola yazdırır.

renkp\_dondur() Amaç: Renk hafıza oyununda kazanılan puanı global AnaPuan değişkenine eklemek. global AnaPuan: Global AnaPuan değişkenini kullanır. renkpuan = ui8.son\_kontrol(): Renk hafıza oyununda kazanılan puanı renkpuan değişkenine atar. AnaPuan = AnaPuan + renkpuan: Renk hafıza oyununda kazanılan puanı AnaPuan'a ekler. Bu fonksiyonlar, farklı oyunlardan kazanılan puanları toplar ve toplam puanı günceller. Böylece kullanıcı, tüm oyunlardan kazandığı toplam puanı görebilir.

pencere7.close(): MainWindow7 penceresini kapatır. puan = 0: Puanı sıfırlar. ui7.pnlabel.setText(f"{puan}"): Puanı ekranda sıfır olarak gösterir. ui7.kelime = random.choice(ui7.kelimeler): Yeni bir kelime rastgele seçer.

**Şekil 9.7 Kod**

ui7.gizli\_kelime = ["\_"] \* len(ui7.kelime): Yeni kelimeyi gizlemek için çizgilerle başlatır. ui7.cizgilabel.setText(" ".join(ui7.gizli\_kelime)): Çizgileri ekranda gösterir. ui7.tahminler = set(): Tahmin edilen harfleri sıfırlar. ui7.hata\_sayisi = 0: Hata sayısını sıfırlar. ui7.imglabel.setPixmap(ui7.resimler[0]): Resmi başlangıç resmine geri döndürür. pencere2.show(): Ui\_MainWindow2 penceresini gösterir. Bu fonksiyon Adam Asmaca oyununu sıfırlayarak yeni bir oyun başlatmak için ana menü penceresini gösterir.

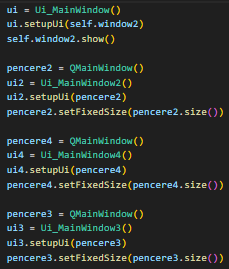


**Şekil 9.8 Kod Şekil 9.9 Kod**

**MarketBtn1:** Ana puanın (AnaPuan) 25.000 veya daha fazla olup olmadığını kontrol eder. Eğer yeterli puan varsa, AnaPuan'dan 25.000 çıkarır. Güncellenen puanı labelpuan ve labelmpuan etiketlerinde gösterir. İlk market butonunu (mb1) gizler, ikinci market butonunu (mb2) gösterir.

**MarketBtn2:** Ana puanın (AnaPuan) 50.000 veya daha fazla olup olmadığını kontrol eder. Eğer yeterli puan varsa, AnaPuan'dan 50.000 çıkarır. Güncellenen puanı labelpuan ve labelmpuan etiketlerinde gösterir. İkinci market butonunu (mb2) gizler, üçüncü market butonunu (mb3) gösterir.

**MarketBtn3:** Ana puanın (AnaPuan) 100.000 veya daha fazla olup olmadığını kontrol eder. Eğer yeterli puan varsa, AnaPuan'dan 100.000 çıkarır. Güncellenen puanı labelpuan ve labelmpuan etiketlerinde gösterir. Üçüncü market butonunu (mb3) gizler. Bir mesaj kutusu açarak "Oyun Bitti. KAZANDIN VE TEŞEKKÜR EDERİM..." mesajını kullanıcıya gösterir.

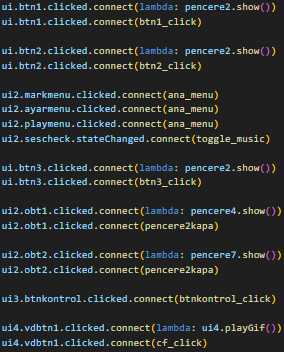
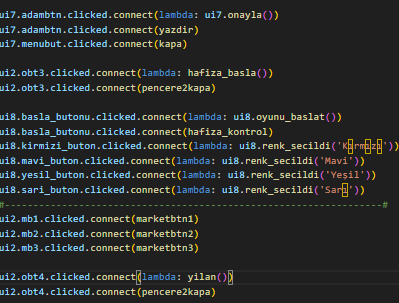
İlk 3 satırda: Ui\_MainWindow sınıfından bir nesne oluşturulur ve setupUi metodu çağrılarak self.window2 penceresi üzerine kullanıcı arayüzü yerleştirilir. Daha sonra pencere gösterilir.

Sonraki 4 satırda: pencere2 adlı bir QMainWindow nesnesi oluşturulur. Ui\_MainWindow2 sınıfından bir nesne oluşturulur ve setupUi metodu ile pencere2 üzerine kullanıcı arayüzü yerleştirilir.Pencere boyutu sabitlenir (setFixedSize), böylece kullanıcı pencerenin boyutunu değiştiremez.

**Şekil 10.0 Kod**

pencere4 adlı bir QMainWindow nesnesi oluşturulur. Ui\_MainWindow4 sınıfından bir nesne oluşturulur ve setupUi metodu ile pencere4 üzerine kullanıcı arayüzü yerleştirilir. Pencere boyutu sabitlenir (setFixedSize).

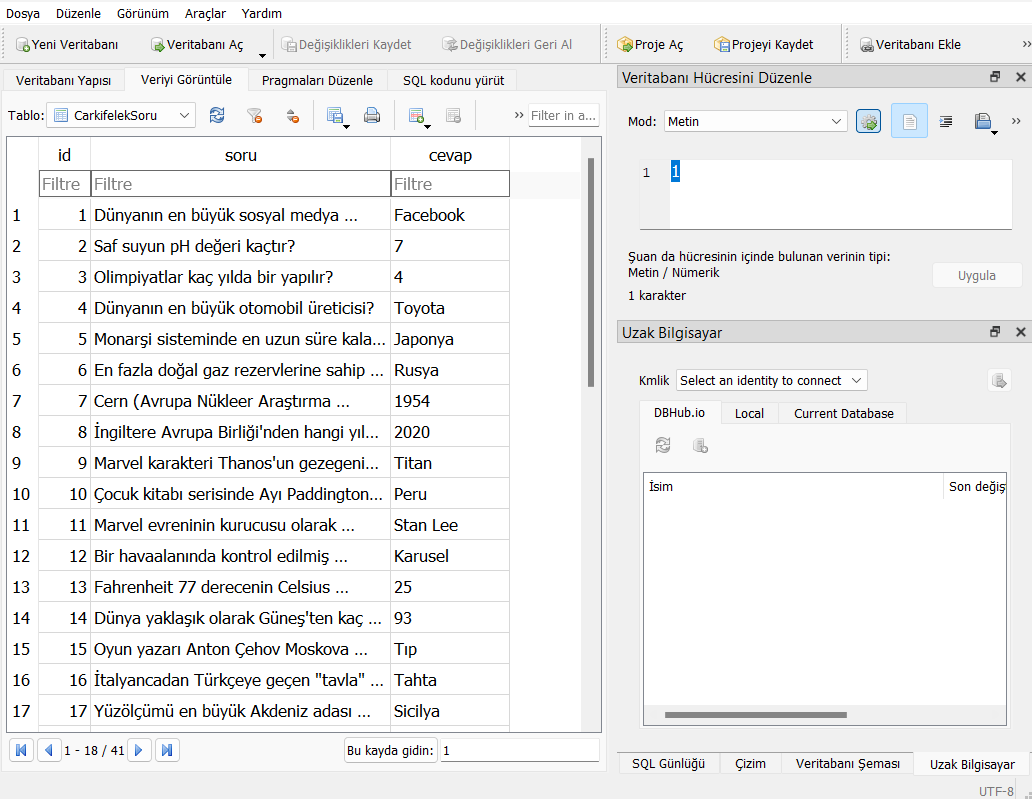
pencere3 adlı bir QMainWindow nesnesi oluşturulur. Ui\_MainWindow3 sınıfından bir nesne oluşturulur ve setupUi metodu ile pencere3 üzerine kullanıcı arayüzü yerleştirilir. Pencere boyutu sabitlenir (setFixedSize).



**Şekil 10.1 Kod** **Şekil 10.2 Kod**

Bu kodlar çeşitli kullanıcı etkileşimlerini (buton tıklamaları ve checkbox değişimleri) belirli fonksiyonları çağırır. Bu sayede kullanıcı arayüzü üzerindeki etkileşimler, ilgili işlevlerin gerçekleştirilmesini sağlar. Kod, oyunlar arasında geçiş yapma, müzik çalma/durdurma, puan güncelleme ve çeşitli oyun fonksiyonlarını çağırma gibi görevleri yerine getirir.

# 3.3 carkifeleksoru.db



**Şekil 10.3 Veritabanı**

Bir veritabanı dosyasıdır. .db uzantısı, "database" kelimesinin kısaltmasıdır ve genellikle veritabanı yönetim sistemleri (DBMS) tarafından kullanılır. Bu tür dosyalar, verilerin yapılandırılmış bir şekilde saklandığı, erişildiği ve yönetildiği dosyalardır. Çarkıfelek oyunumuzun soru ve cevap verilerini içerir. 41 adet soruya karşılık 41 adet cevap vardır. Her bir veri random id ler üzerinden çağırılır.

# 3.4 Kullanılan Görseller (img, gif, jpeg, png)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |

# 3.5 music.mp3

 Programın başlamasından sonuna kadar sonsuz bir döngüde çalar. Kullanıcıyı rahatlatacak ve iyi hissettirecek bir mp3 dosyası belirlenmiştir. Ayarlardan ses kapatılabilir.

# 3.6 .ui dosyaları

.ui dosyası, Qt Designer gibi araçlarla oluşturulan kullanıcı arayüzü tasarım dosyasıdır. Bu dosyalar, PyQt veya PySide gibi Python ile Qt kütüphanelerini kullanan programlarda kullanılır.

Qt Designer, görsel bir arayüz tasarlamak için kullanılan bir WYSIWYG (Ne Görürsen Onu Alırsın) araçtır. Bu araçla kullanıcılar, bir uygulamanın arayüzünü sürükleyip bırakarak oluşturabilir ve düzenleyebilirler. Tasarım işlemi bittiğinde, Qt Designer .ui uzantılı bir dosya olarak tasarımı kaydeder. Bu .ui dosyası, XML tabanlı bir formatta kullanıcı arayüzünün yapısını, öğelerini ve özelliklerini içerir.

Python ile Qt kütüphanelerini kullanan bir uygulama geliştirirken, .ui dosyası, PyQt veya PySide gibi araçlarla yüklenebilir ve program içinde kullanılabilir. Uygulama çalıştığında, .ui dosyası yüklenir ve kullanıcı arayüzü belirli bir şekilde oluşturulur. Bu şekilde, kullanıcı arayüzünü kodlamak yerine görsel bir araç kullanılarak tasarlanabilir ve daha sonra programlama diline entegre edilebilir.

# Kaynakça

*"Qt Designer." Qt Documentation, doc.qt.io/qt-5/qtdesigner-manual.html.*

*"Qt Designer Nedir?" Python-istihza.yazbel.com/qt/qt\_designer.html.*

*"Qt Designer." Wikipedia, tr.wikipedia.org/wiki/Qt\_Designer.*

*"PyQt5 - QtDesigner ile UI Dosyası Oluşturma." GeeksforGeeks,* [*www.geeksforgeeks.org/pyqt5-qt-designer-with-python/*](http://www.geeksforgeeks.org/pyqt5-qt-designer-with-python/)*.*

*"DB Browser for SQLite." SQLite Browser, sqlitebrowser.org/.*

*"DB Browser for SQLite Nedir? Nasıl Kullanılır?" Bilgiustam, bilgiustam.com/db-browser-for-sqlite-nedir-nasil-kullanilir/.*

*"SQLite Veritabanı İşlemleri." Özgür Yazılım Atölyesi, ozguryazilimatolyesi.com/sqlite-veritabani-islemleri/.*

*"DB Browser for SQLite: SQLite İçin Ücretsiz Bir Veritabanı Tarayıcısı." Tech Tutorials,* [*www.techtutorials.net/articles/db\_browser\_for\_sqlite*](http://www.techtutorials.net/articles/db_browser_for_sqlite)*.*

*"Video Oyunu Tarihçesi." Gamepazar, www.gamepazar.com.tr/video-oyunu-tarihcesi/.*

*"Video Oyun Endüstrisi Tarihi: Kısa Bir Bakış." Bigumigu, bigumigu.com/haber/video-oyun-endustrisi-tarihi-kisa-bir-bakis/.*

*"Video Oyun Tarihi: Gelişimi ve Değişimi." Webtekno,* [*www.webtekno.com/video-oyun-tarihi-h28594.html*](http://www.webtekno.com/video-oyun-tarihi-h28594.html)*.*

*"Visual Studio Code ile Başlamanın Temelleri." Medium,* [*medium.com/@ataberk0turk/visual-studio-code-ile-ba%C5%9Flaman%C4%B1n-temelleri-e9ae16712a49*](mailto:medium.com/@ataberk0turk/visual-studio-code-ile-ba%C5%9Flaman%C4%B1n-temelleri-e9ae16712a49)*.*

*"Stable Diffusion ile Fotoğraflarda Yapay Zeka Kullanımı." Yapay Zeka, www.yapayzeka.net/stable-diffusion-ile-fotograflarda-yapay-zeka-kullanimi/.*

*"Qt Designer Kullanımı ve PyQt5 ile Entegrasyon." Medium, medium.com/@aliyilmaz-28060/qt-designer-kullan%C4%B1m%C4%B1-ve-pyqt5-ile-entegrasyon-1e0f9e0dc1e2.*

*"DB Browser for SQLite: SQLite İçin Ücretsiz Bir Veritabanı Tarayıcısı." Tech Tutorials,* [*www.techtutorials.net/articles/db\_browser\_for\_sqlite*](http://www.techtutorials.net/articles/db_browser_for_sqlite)*.*