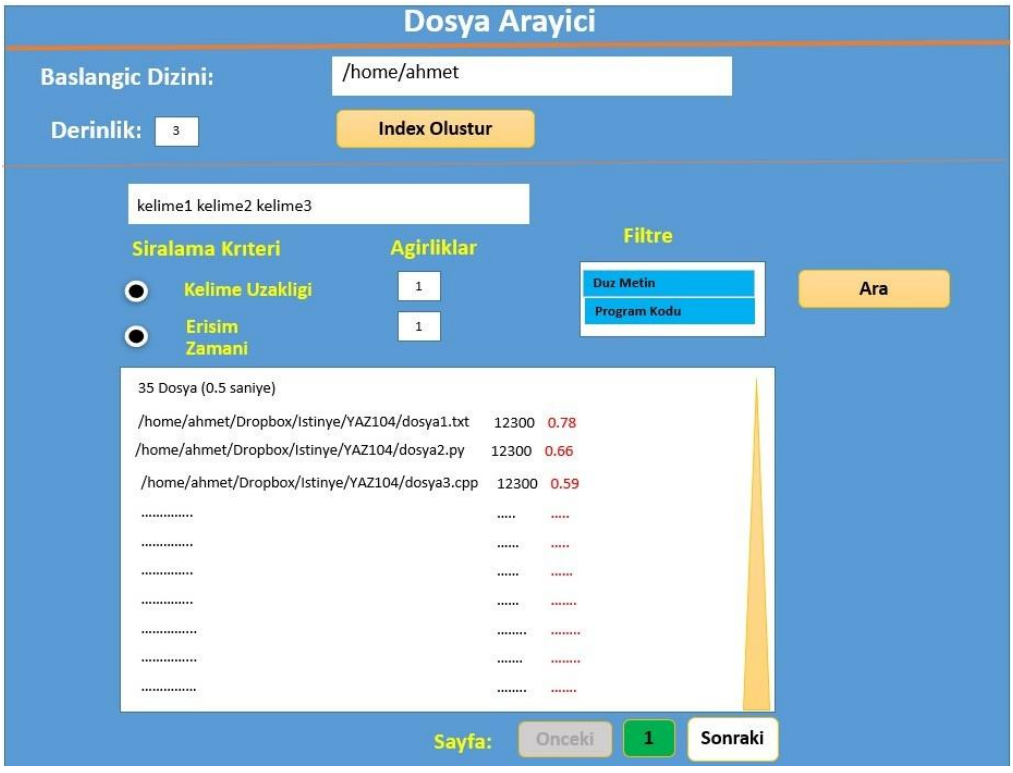
YAZ 104 Temel Programlama II
Bahar 2020
Final Projesi
(300 puan)
10 Haziran 2020

Bu projede, bilgisayarınızdaki text dosyalar içinde arama yapmayı kolaylaştıracak ve arama sonuçlarını kullanıcının belirleyeceği birtakım kriterlere göre sıralayıp listeleyecek bir Python programı geliştirmeniz beklenmektedir. Projenin gereksinimlerini aşağıda bulabilirsiniz.

- Programınızın Şekil 1'de verilene benzer bir grafik kullanım ara yüzü (GUI) olmalıdır. Şekildeki renklendirmeler sadece bir örnektir, siz projenizde dilediğiniz renkleri kullanabilirsiniz. Ancak widget'ların yerleşimi şekildeki gibi veya ona çok yakın olmalıdır. **Dikkat!:** Projenizde Place Geometry Manager kullanmamalısınız; Pack ya da Grid Geometry Manager'dan herhangi birini kullanabilirsiniz.



Şekil 1

- Kullanıcı ilk olarak bilgisayarda mevcut bir dizinin patikasını belirtecek ve burası daha sonraki arama işlemleri için bir başlangıç noktası olarak kabul edilecektir. "**İndexi Olustur**" butonuna tıklandığında, programınız başlangıç dizini olarak belirtilen dizin ve yine kullanıcı tarafından belirtilen derinliğe kadar onun altındaki tüm alt-dizinlerdeki text dosyaları dolaşıp index'lemelidir. Eğer kullanıcı herhangi bir derinlik değeri belirtmediyse, programınız varsayılan bir derinlik değeri kullanmalıdır.

- Her bir dosya index'lenirken, programınız dosyaya en son erişim zamanını ve text dosyanın tipini de saptayıp, dosyanın içeriğine ilave olarak bu bilgileri de bir veri tabanında saklamalıdır. Text dosyalarının ya herhangi bir programlama dilindeki bir program kodu, ya da düz metin olarak iki farklı tipte mevcut olabileceğini kabul edebilirsiniz. Bir başka deyişle arama motorunuz düz metinden ibaret text dosyalarına ek olarak program kodlarını da index'leyip içerikleri üzerinden arama yapılmasına olanak sağlamalıdır. Örneğin aylar önce yazmış olduğunuz bir Python (ya da C, C++) programınızın bilgisayarınızda bir dizinde durduğunuzu biliyorsunuz, ama zaman içerisinde dizin yapınız çok karıştığı için bulamadığınızı düşünün. Eğer programınızın içinde yazmış olduğunuz bir yorum satırı veya kullandığınız bir değişken ismini hatırlıyorsanız, geliştireceğiniz bu projeyi kullanarak o programın kaynak koduna kolayca ulaşabilmelisiniz. Bilgisayarlardaki işletim sistemlerinin (Windows, Mac OS, Linux) benzer dosya arama servisleri veren araçları olduğunu fark etmişsinizdir.

- Veri tabanı olarak **shelve** modülünü kullanmalısınız, bu projede başka bir veri tabanı modülünün (**sqllite** vb. derslerde görmediğimiz modüller) kullanımına izin **yoktur**. Ne tür sözlük yapıları kullanmanız gerektiğine kendiniz karar vermelisiniz. Derslerde gördüğümüz ***mysearchengine.py*** modülünü, örneğin index'leme yapan fonksiyonlarını, inceleyebilir ve o modülde kullanılan sözlük yapılarından esinlenebilirsiniz ancak bu proje için daha farklı yapıda sözlüklere ihtiyacınız olabileceğini de göz ardı etmemelisiniz. Web'de hipertext'lerle birbirlerine bağlanmış sayfaların içeriklerini değil, bir dizin hiyerarşisinde yer alan dosyaların içeriklerini index'leyeceğinizi unutmayın.

- Index'leme aşaması tamamlandığında (bunu bir şekilde kullanıcıya hissettirmelisiniz, nasıl yapacağınıza kendiniz karar verebilirsiniz) kullanıcı bir veya daha fazla arama kelimeleri girerek dosya aramaya başlayabilmelidir. Arama amaçlı kutuya kullanıcı boşluk karakteriyle ayrılmış olarak kelime veya kelimeleri yazıp "**Ara**" butonuna tıkladığında, programınız daha önce index'lemiş olduğu dosyalar arasından arama kelimelerinin tamamını içerenleri bulmalı ve bu dosyalara ilişkin bilgileri ara yüzün alt bölümündeki text alanına yazmalıdır.

- Kullanıcı sorgu alanına girdiği kelimelerini tırnak işaretleri (çift ya da tek tırnak) arasına alarak da belirtebilir. Bunun anlamı, kullanıcı tırak içinde belirttiği kelime grubunu olduğu gibi birebir o yapıda bulmak istiyor demektir. Yani örneğin kullanıcı şu şekilde bir sorgu cümlesi yazmışsa;

  "kelime1 kelime2"

  bu tip bir sorgu sonucunda listelenecek dosyalar, içlerinde *kelime1*'in *kelime2*'nin hemen ardından geldiği, aralarına başka kelime veya kelimelerin girmediği dosyalar olmalıdır. Yukarıdaki örnek için hem *kelime1*'i hem de *kelime2*'yi içerdiği halde bu iki kelime arasında başka kelimelerin de olduğu, bu iki kelimenin ard arda hiç bulunmadığı dosyalar arama sonucunda hiç görünmemelidir. Kullanıcının ikiden fazla kelimeyi de tırak içine alarak (örneğin ***"kelime1 kelime2 kelime3"*** gibi) sarama yapabileceği de göz önünde bulundurulmalıdır (*İpucu ve yönlendirme*: b*u özelliği aşağıda belirtilen iki sıralama ölçütüne ek, kullanıcı seçiminden bağımsız olarak sürekli mevcut olan üçüncü bir sıralama ölçütü olarak gerçeklemeyi düşünebilirsiniz).*

- Programınız arama sonuçlarının hangi sırada listeleneceğine dair iki farklı sıralama ölçütü sunmalıdır.

- Kelime Uzaklığı Ölçütü: Yazacağınız bir metot/fonksiyon ile kelime uzaklığı ölçütünü hesaplayabilmelisiniz. Örneğin kullanıcı ***kelime1 kelime2 kelime3*** şeklinde üç kelimelik bir sorgu girdiyse ve bir dosyada kelime1 ile kelime2 arasında 10 başka kelime, kelime2 ile kelime3 arasında da 7 başka kelime varsa, o dosyanın kelime uzaklık skoru 10 + 7 = 17 olmalıdır. Eğer dosyada aranan kelimelerin bir ya da birkaçı birden fazla kere geçiyorsa, yukarıdaki hesabın birden fazla sonucu olabileceği için, o dosyanın kelime uzaklık skoru, olası skorlardan minimum olanına eşitlenmelidir. Arama sonuçlarının listelendiği bölgesinde dosyalar kelime uzaklık skorları büyükten küçüğe doğru sıralanmalıdırlar, yani buradaki kabulümüz bir dosyanın kelime uzaklık skoru ne kadar büyükse, yapılan aramayla olan ilgisi o derecede fazla olduğu anlamındadır.

  Kullanıcı sorgu alanına girdiği kelimelerini tırnak işaretleri (çift ya da tek tırnak) arasına almadıysa, kelimelerin dosyada hangi sırada geçtiklerinin hesaplanan skora herhangi bir etkisi olmamalıdır. Yani örneğin, kullanıcının ***kelime1 kelime2 kelime3*** şeklinde yaptığı bir arama için, bir dosyada *kelime1*'in *kelime2*'den ya da önce ya da sonra geliyor olmasının o dosyanın kelime uzaklık skoruna herhangi bir etkisi olmamalıdır.

  Ancak, eğer kullanıcı sorgusunu yazarken iki ya da daha fazla kelimeyi tırnak içerisine alarak belirtmişse, yani örneğin sorgusunu şe şekilde yazmışsa;

  "kelime1 kelime2" kelime3

  bu durumda kelime uzaklık skoru hesaplanırken sadece kelime2'nin kelime1'den sonra geldiği durumların skora katkısı olmalı, ancak kelime2'nin kelime1'den önce geldiği durumların skor hesabına katkısı olmamalıdır. Dahası yukarıda da belirtildiği gibi, bu iki kelimenin ard arda hiç bulunmadığı dosyalar arama sonuçlarında dahi yer almamalıdır. Yukarıdaki örnek için *kelime3*'ün dosyada diğer kelimelere göre nerede olduğunun ve *kelime2* ile arasında ne kadar bir uzaklık olduğunun, skor hesabına yapacağı etki dışında bir önemi yoktur.
- Erişim Zamanı Ölçütü: Kullanıcı dosyaların en yakın geçmişteki erişim zamanlarını da bir diğer sıralama ölçütü olarak belirtebilmelidirler. En yakın zaman önce erişilmiş olan dosyalar, bu ölçüte göre, arama sonuç alanında daha önceden erişilmiş olan dosyalara göre sıralamada daha yukarıda yer almalıdırlar.

- Yukarıdaki ölçütlerin her birine göre elde edilen skorlar, uygun bir şekilde 0 le 1,0 değerleri arasına normalize edilerek kullanılmalı, ayrıca kullanıcı dilerse belirttiği ağrılıklar ile her iki ölçütün de beraberce dikkate alınmasını isteyebilmelidir.

- ***Dosya Tipi Filtresi:*** Kullanıcı programın desteklediği iki dosya tipinden hangisi için arama yapmak istediğini belirtebilmelidir. İlgili Listbox'da kullanıcıya bu anlamda iki seçenek sunmalısınız; *Program Kodu* ve *Düz Metin*. Kullanıcı bunlarda en az birini, dilerse ikisini birden seçebilmelidir. Program ilk ayağa kalktığında her ikisi birden seçili gelmelidir.

- Kullanıcı "**Ara**" butonuna bastığında programınız daha önceden oluşturmuş olduğunuz veri tabanını tarayarak yukarıda belirtilen filtre, kriter ve sorgu kelimelerine uyan dosyaları bulup listelemelidir.
  - Sonuç bölgesinin en üst kısmında, arama işlemi sonucunda kaç adet uygun dosya saptandığı ve işlemin ne kadar zaman aldığı belirtilmelidir.
  - Arama sonuçlarına "***Sayfalama***" (*Pagination*) uygulanmalı ve en son sayfa hariç diğer tüm sayfalarda en fazla 10 adet sonuç bulunmalıdır. Sonuç alanında ayrıca, 10 adet sonucun sığmama ihtimaline karşı dikey ve yatay Scrollbar'lar da bulunmalıdır. Şekil 1'de görüldüğü gibi, en altta, sayfalar arasında gezinmeyi sağlamak üzere, "***Sonraki***" ve "***Önceki***" şeklinde

etiketlenmiş butonlar ve bu butonların arasında da o an gösterilen sayfanın göreceli sıra numarasını gösteren bir etiket yer almalıdır (İlk sayfa için bu etiket 1'i göstermelidir). İlk sayfadayken "Önceki", son sayfadayken de "Sonraki" butonları **tıklanamaz** olmalıdırlar.
    - Sonuçlar da 1'den başlayarak numaralandırılmalı; ayrıca her bir sonuç satırı için ilgili dosyanın boyutu ve hesaplanan sıralama skor değeri de dosyanın mutlak patika bilgisiyle birlikte listelenmelidir.
- Kullanıcınız bir arama yapıp sonuçlarını aldıktan sonra, en başa dönüp dilerse başlangıç dizinini değiştirip ya da başlangıç dizinini hiç değiştirmeden index'i yeniden inşa edebilmeli; index'i yeni baştan oluştursa da oluşturmasa da dilediği parametrelerde değişiklik yaparak tekrar arama da yapabilmelidir. Kullanıcının iki indeks oluşturma adımları arasında dizin hiyerarşisine yeni dosyalar ilave edilmiş olursa son indeks oluşturma aşamasından sonraki arama sorguları bu yeni eklenen dosyaları da dikkate almalıdır.

- Kullanıcı aşağıda belirtilen hareketlerden herhangi birini yaptığı takdirde, programınız kendisini uygun ve görebileceği bir hata mesajıyla uyarmalıdır:

    - Bilgisayarda var olmayan bir başlangıç dizinin patikasını yazdıysa
    - En az bir sorgu kelimesi belirtmeden "Ara" butonunu tıkladıysa
    - En az bir sıralama ölçütünü seçmediyse,
    - Birden fazla sıralama ölçütü seçtiği halde, her ikisinin de ağırlıklarını belirmediyse.
    - En az bir dosya kategorisi belirlemediyse

İşinize yarayacak bilgiler:

- Projenize "**mysearchengine.py**" import ederek **kullanmamalısınız**, ancak ilgili olduğunu düşündüğünüz kod parçalarını alıp uygun hale getirip kullanabilirsiniz.

- Bir dosyanın tipinin ne olduğunu saptayabilmek için dosyanın ismine ya da uzantısına güvenmemelisiniz. Bu amaçla ***Python Magic*** isimli modülden faydalanabilirsiniz. Python Magic ile ilgili bilgilere aşağıdaki link'den erişebilirsiniz:

  https://pypi.org/project/python-magic/

  Uygun kullandığınız takdirde Python Magic size bir dosyanın tipini (image, text, vs.) ve hatta bir program koduysa, hangi programlama dilinde yazılmış olduğunu da söyleyebilmektedir.

- Bilgisayardaki dizin ve dosya işlemleri (dizin değiştirme, dizindeki dosyaları listeleme, vs.) için ***os*** modülünden faydalanmalısınız. Dosyalara erişim zamanlarını tespit edebilmek için de ***os.stat()*** fonksiyonu size yardımcı olacaktır.
- Ara sonuç alanında Text widget kullanabilirsiniz. Sonuç alanında çeşitli alanları değişik font ya da renkle zenginleştirmek isterseniz (zorunlu değil opsiyoneldir), Text widget'in ***tag_add, tag_modify*** gibi metotlarından faydalanabilirsiniz.
- Zaman ölçmek için aşağıdaki link'te ***Time*** modülünün kullanımına basit bir örnek bulabilirsiniz.

  https://stackoverflow.com/questions/3620943/measuring-elapsed-time-with-the-time-module

Dikkat edilmesi gereken konular:

- Proje metnini dikkatlice okuyun ve metinde geçen "**olmalıdır**", "**olmamalıdır**", "**yapabilmelidir**", "**edebilmelidir**" vb. Gereklilik kipinde ifadeler içeren cümlelere özel önem verip ne istendiğini anlamaya çalışın.
- Projenizi kodlamaya başlamadan önce problemi çözmek için hangi veri yapılarını kullanmanız gerektiğine karar verin.
- Proje en fazla iki kişilik gruplar halinde yapılabilir. Diğer gruplarla fikir alışverişinde bulunabilirsiniz ancak kesinlikle kod paylaşımı yapmamanız gerekmektedir.
- Eğer bir web sayfası ya da bir kitaptan bir kod parçasını direkt alıp kullandıysanız, programınızın o bölgesine yazacağınız yorum satırları içinde bu durumu (programınızın o parçasını nereden alıp kullandığınızı) açıkça belirtin.
- Projenizi tek bir Python dosyası olarak teslim etmeniz beklenmektedir,
- Son teslim tarihi 10 Haziran Çarşamba günü saat 14:00'dir. Bu tarihten sonra teslim edilen projeler kabul edilmeyecektir.

Eğer tek başınıza çalıştıysanız oluşturduğunuz Python dosya adı şu şekilde olmalıdır (tek bir dosya teslim edeceğiniz için ".zip" ya da ".rar"lamanıza gerek bulunmamaktadır):

ad-soyad_okulnumarası_final.py

Eğer iki kişilik gruplar halinde çalıştıysanız dosya adınız şu şekilde olmalı ve grup halinde tek bir ödev gönderimi yapmalısınız:

ad_soyad1_numara1_ad_soyad2_numara2_final.py

Ödev dosyalarınızı yukarıdaki isimlerle sakladıktan sonra Piazza sistemine yükleyiniz.

Dosya adlarında Türkçe karakterler (ı, ü, ğ, ö, ç, ş) ve büyük harfleri kullanmayınız.

Değerlendirme Kriterleri:

			İşlevsellik							
Class yapısı kullanımı (15)	Yorum satırları (15)	Anlamlı değişken isimleri (10)	Program ayağa kalkıp çalışabiliyor mu (15)	Ara yüz (GUI) görsel tasarım (15)	Dizin hiyerarşisinde dolaşarak index'i inşa etme (40)	Bir veya birden fazla kelimeyle arama yapma ve sonuçların listelenmesi (40)	Seçilen ölçüt veya ölçütlere uygun sıralama yapılması (40)	Başlangıç dizinini değiştirme dahil parametreleri ndeğiştirilerek yeni baştan arama yapılabilmesi (40)	İki indexleme arasında yeni gelen dosyaların saptanabilmesi (40)	Hata mesajları (daha önceki adımlar başarıyla gerçekleşebildiyse) (30)

Kolay gelsin.