T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ LİSANS YAZILIM MÜHENDİSLİĞİ PROGRAMI

Python İle Konsol Menü

HAZIRLAYANLAR:

YUSUF USTAOĞLU 220502003 https://github.com/Katlicia

AHMET EREN ŞENGÜL 220502036
https://github.com/Eren1213

DERS SORUMLUSU:

PROF. DR. HÜSEYİN TARIK DURU

30.10.2023

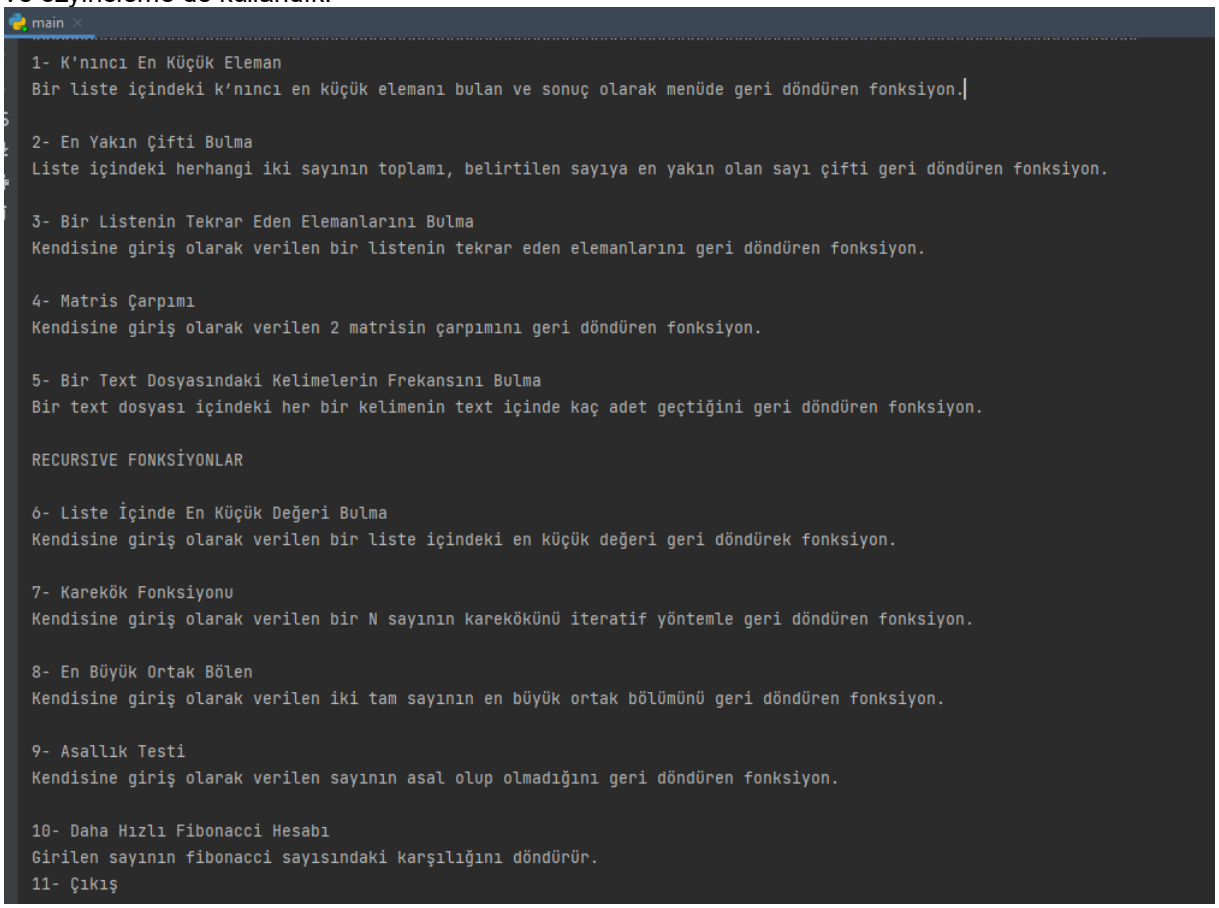
Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli
info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

1. ÖZET

Bizden istenen ödev, 10 farklı fonksiyona sahip bir Python kodunu içerir ve kullanıcıdan önce hangi fonksiyona ulaşmak istediğini , sorup ardından o fonksiyonda kullanmak istediği girdileri girmesini sağlayan bir konsol menüsü yapmamız isteniyor.Bu nedenle öncelikle bir konsol menüsü yapıp ardından o konsol menüsünde bizden yapılması istenen fonksiyonları yapabilecek python kodlarını yazmak üzerine odaklandık. Tamamlandıktan sonra uyuşmayan bazı fonksiyonlar ve koşullar üzerinde düzeltme yaptık.Sonuç olarak, işlevsel bir Python konsol menüsü oluşturduk.

2. GİRİŞ

Aşağıda gördüğümüz python menüsü projesinin ana amacı olmakta ve bu menüyü kullanarak 1 ile 10 arası numaralandırılmış fonksiyonları çağırarak kullanıcıya sunmak amaçlanmaktadır. Bu fonksiyonları yaparken if, else, elif, while, while true, break, gibi koşulları kullandık ayrıca bazılarında for döngülerini ve özyineleme de kullandık.



3. YÖNTEM

3.1 K'nıncı En Küçük Elemanı Bulma

Verilen listedeki k'nıncı en küçük elemanı bulma.

Bu fonksiyonda ilk olarak kullanıcıdan geçerli değerler aldığımıza emin oluyoruz. Eğer girilen parametre geçerli ise küçükten büyüğe sıraladığımız listemizden istenilen elemanı geri döndürüyoruz.

```python
# 1. Örnek
def k_kucuk(num, my_list):

    # Hata Kontrolleri
    if num < 0:
        raise ValueError("Girilen sayı negatif olmamalı.")

    if type(my_list) != list:
        raise ValueError("Geçerli liste giriniz. ")

    if num > len(my_list):
        raise IndexError(f"Listede {num} eleman yok. ")

    my_list.sort()

    return my_list[num-1]
```

3.2 En Yakın Çifti Bulma

Verilen sayının verilen listedeki toplamları o sayıya en yakın olan 2 elemanın geri döndürülmesi.

Bu fonksiyonda da hata kontrolü yaptıktan sonra listedeki 2'li elemanların toplamını alıp yeni bir listeye kaydediyoruz. Daha sonra toplamlar ile sayının farkını 3. bir listeye kaydediyoruz ve o listenin en küçük değerini alıyoruz. Daha sonra toplamları en yakın olan 2 elemanı ilk listeden buluyoruz ve döndürüyoruz.

```python
# 2. Örnek
def en_yakin_cift(num, my_list):

    # Hata kontrolleri.
    if type(my_list) != list:
        raise ValueError("Geçerli liste girin.")

    # Küçükten büyüğe sırala.
    my_list.sort()

    new_list = []

    # Sırayla liste içindeki 2 li sayıları toplama.
    for i in range(len(my_list)):
        for j in range(i+1, len(my_list)):
            toplam = my_list[i] + my_list[j]
            new_list.append(toplam) # Toplamları yeni bir listeye kaydet.

    third_list = []

    # Toplamlar ile verilen sayı arasındaki farkı bulup yeni bir listeye kaydet
    for x in new_list:
        fark = abs(num-x)
        third_list.append(fark)

    # Listeyi küçükten büyüğe sırala.
    third_list.sort()

    # Kullanıcıdan alınan listedeki sayıların en yakın çiftini geri döndür.
    for y in range(len(my_list)):
        for z in range(y+1, len(my_list)):
            toplam = my_list[y] + my_list[z]
            sayilar = (my_list[y], my_list[z])
            if abs(toplam - num) == third_list[0]:
                return sayilar
```

3.3 Bir Listenin Tekrar Eden Elemanlarını Bulma

Fonksiyona giriş olarak verilen listedeki tekrar eden elemanları list comprehension yöntemi kullanarak döndürme.

Hata kontrolünden sonra list comprehension yöntemiyle tekrar eden elemanları bir kümeye kaydediyoruz ve tekrardan liste olarak döndürüyoruz.

```python
# 3. Örnek
def tekrar_eden_elemanlar(my_list):
    # Hata kontrolleri.
    if type(my_list) != list:
        raise ValueError("Geçerli liste giriniz. ")

    # List comprehension kullanılarak listedeki elemanın birden fazla olup olmadığını kontrol ediyoruz.
    # Eğer birden fazlaysa o değeri dize olarak kaydeder.
    # Böylelikle aynı eleman sadece 1 kez gözükür.
    return list(set([x for x in my_list if my_list.count(x) > 1]))
```

3.4 Matris Çarpımı

Fonksiyona giriş olarak verilen 2 matrisin çarpımını döndürme.

Matris çarpımı kuralı gereği eğer A matrisinin satır sayısı B matrisinin sütun sayısına eşit değilse o matrisler çarpılamaz.

Zip metodu ile birleştirdiğimiz satın sütunları x, y değişkenlerine atıp çarpıyoruz ve yeni matrisimize giriyoruz.

```python
# 4. Örnek
def matris_carpimi(mat1, mat2):
    satirA = len(mat1)
    sutunB = len(mat2[0])

    # Hata kontrolü.
    if satirA == sutunB:
        matrisC = [] # Çarpım matrisi.
        for i in range(satirA):  # A matrisinin satırlarında dolaşır.
            satir = [] # Çarpım matrisimize eklemek için satır oluşturuyoruz.
            for j in range(sutunB): # B matrisinin sütünlarında dolaşır.
                total = 0
                # A matrisinin satırları ile B matrisinin sütünları x, y değerlerine atılır.
                for x, y in zip(mat1[i], (row[j] for row in mat2)):
                    total += x * y # Değerler çarpılır.
                satir.append(total) # Satır listesine eklenir.
            matrisC.append(satir) # C matrisine eklenir.
        return matrisC
    else:
        raise ValueError("Geçerli matris giriniz.")
```

3.5 Bir Text Dosyasındaki Kelimelerin Frekansını Bulma

Bir text dosyasındaki kelimelerin kaç kez geçtiğini döndüren fonksiyon.

Girilen metindeki kelimeleri listeye kaydedip daha sonra onları sözlüğe çeviriyoruz, böylece hangi kelimeden kaç tane olduğunu anlıyoruz.

Soruda reduce, map gibi metotların kullanılması istendiği için fonksiyonumuzun içine 1'i gizli 2 fonksiyon eklememiz gerekiyor. Çünkü bu metotlar parametre olarak fonksiyon alıyor.

Daha sonra map ile kelimeleri sayiyor, reduce ile de bunları sözlüğe kaydediyoruz.

```python
# 5. Örnek
from functools import reduce
def kelime_frekans(file_path):
    with open(file_path, "r") as f:
        text = f.read()
        # Dosyadaki kelimeleri listeye kaydet.
        kelimeler = text.split()

    def kelime_hesapla(kelime):
        # Kelime sayılarını sözlük döndürür.
        # kelime: key değerini, kelimeler.count(kelime) ise value değerini belirtir..
        return {kelime: kelimeler.count(kelime)}

    # Kelimeleri kümeye çevirir.
    kelime_dizi = set(kelimeler)

    # Reduce ile key ve value değerlerini yeni bir sözlüğe kaydederiz.
    # Map ile kelime_hesapla fonksiyonunu, kelime_dizi üstünde çalıştırırız.
    kelime_hz = reduce(lambda x, y: {**x, **y}, map(kelime_hesapla, kelime_dizi))

    return kelime_hz
```

3.6 Örneği dahil bundan sonraki örnekler özyinelemeli olacaktır.

3.6 Liste İçinde En Küçük Değeri Bulma

Kendisine verilen listedeki en küçük değeri döndüren özyinelemeli fonksiyon.

Özyinelemeli fonksiyonlarda kendimize bir temel durum belirlemezsek fonksiyon Python'un belirlediği max recursion değerine kadar çalışır ve sonunda uyarı verir.

Bu nedenle kendimize bir temel durum belirlememiz gerekir.

Aşağıda görüldüğü gibi belirlenen temel durum eğer liste tek elemanlı ise veya tek elemanlı kalmış ise onu döndürmesidir.

Fonksiyon her tekrar çağrıldığında eleman sayısı 1 azalır.

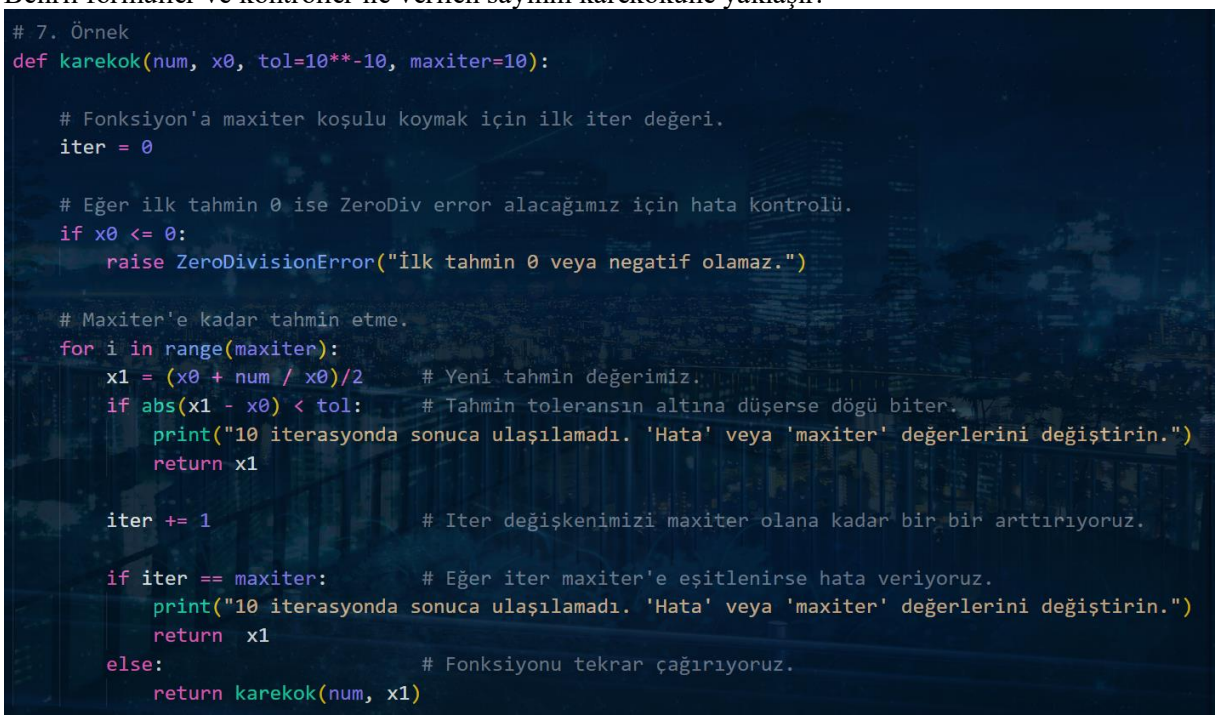
Fonksiyonun çalışma mantığı ise min metoduyla listenin 0. indexindeki eleman ile kalan elemanların karşılaştırılmasıdır.

```python
# 6. Örnek
def en_kucuk_deger(my_list):
    if type(my_list) != list:
        raise ValueError("Geçerli liste giriniz. ")

    # Eğer liste tek elemanlı ise o elemanı geri döndürür.

    # Temel Durum
    if len(my_list) == 1:
        return my_list[0]

    # Listenin ilk elemanı ile diğer elemanları arasında min() fonksiyonu ile karşılaştırma yapar.
    else:
        return min(my_list[0], en_kucuk_deger(my_list[1:]))
```

3.7 Karekök Fonksiyonu

Verilen bir N sayısının karekökünü bulmak için Babil döneminden beri kullanılan iteratif yöntemi uygulayan fonksiyon.

Belirli formüller ve kontroller ile verilen sayının karekköüne yaklaşır.

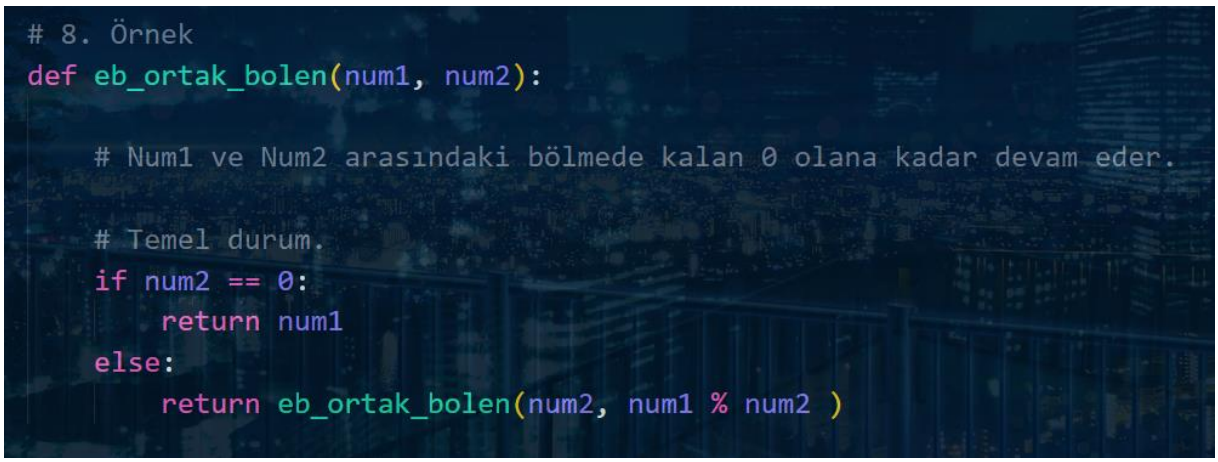


```python
# 7. Örnek
def karekok(num, x0, tol=10**-10, maxiter=10):

    # Fonksiyon'a maxiter koşulu koymak için ilk iter değeri.
    iter = 0

    # Eğer ilk tahmin 0 ise ZeroDiv error alacağımız için hata kontrolü.
    if x0 <= 0:
        raise ZeroDivisionError("İlk tahmin 0 veya negatif olamaz.")

    # Maxiter'e kadar tahmin etme.
    for i in range(maxiter):
        x1 = (x0 + num / x0)/2      # Yeni tahmin değerimiz.
        if abs(x1 - x0) < tol:      # Tahmin toleransın altına düşerse dögü biter.
            print("10 iterasyonda sonuca ulaşılamadı. 'Hata' veya 'maxiter' değerlerini değiştirin.")
            return x1

        iter += 1                   # Iter değişkenimizi maxiter olana kadar bir bir arttırıyoruz.

        if iter == maxiter:         # Eğer iter maxiter'e eşitlenirse hata veriyoruz.
            print("10 iterasyonda sonuca ulaşılamadı. 'Hata' veya 'maxiter' değerlerini değiştirin.")
            return  x1
        else:                       # Fonksiyonu tekrar çağırıyoruz.
            return karekok(num, x1)
```

3.8 En Büyük Ortak Bölen

Kendisine verilen 2 tam sayının ebobunu döndüren fonksiyon.

İki sayının kalansız bölümüne kadar devam ederek ebobu döndürür.



```python
# 8. Örnek
def eb_ortak_bolen(num1, num2):

    # Num1 ve Num2 arasındaki bölmede kalan 0 olana kadar devam eder.

    # Temel durum.
    if num2 == 0:
        return num1
    else:
        return eb_ortak_bolen(num2, num1 % num2 )
```

3.9 Asallık Testi

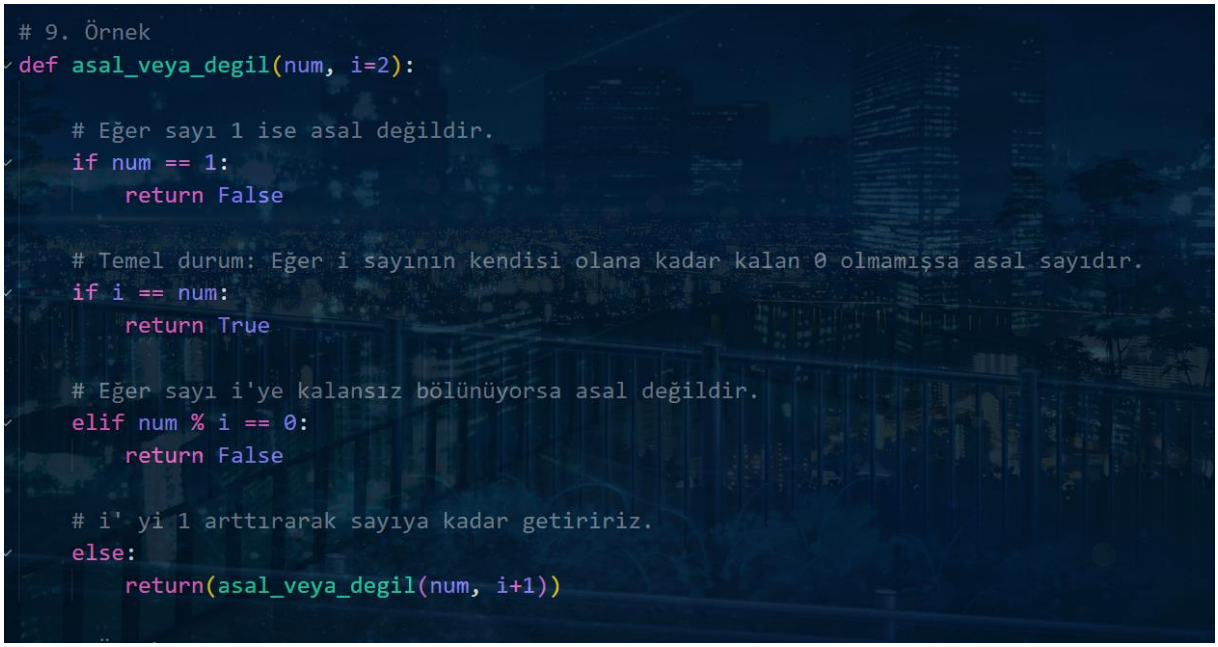
Verilen sayının asal olup olmadığını döndüren fonksiyon.

Eğer verilen i değeri ve sayının bölümü, i değerinin sayının kendisi olana kadar

kalanlı ise o sayı asaldır.

Burada yaptığımız şey, i değerini fonksiyonu her çağırdığımızda 1 arttırarak sayının kendisine kalansız bir şekilde bölünüp bölünmediğini kontrol ediyoruz.

İlk satırda anlattığım gibi eğer i değeri sayının kendisine eşit oluyorsa o sayı asaldır ve bu da temel durumumuzdur.



```python
# 9. Örnek
def asal_veya_degil(num, i=2):

    # Eğer sayı 1 ise asal değildir.
    if num == 1:
        return False

    # Temel durum: Eğer i sayının kendisi olana kadar kalan 0 olmamışsa asal sayıdır.
    if i == num:
        return True

    # Eğer sayı i'ye kalansız bölünüyorsa asal değildir.
    elif num % i == 0:
        return False

    # i' yi 1 arttırarak sayıya kadar getiririz.
    else:
        return(asal_veya_degil(num, i+1))
```

3.10 Fibonacci Hesabı

Verilen sayının fibonacci sayısı olarak karşılığını döndüren fonksiyon.

Temel durumumuz k'nın n'ye eşit olması.

Çünkü N'nin fibonacci sayısını fib(k) üzerinden buluyoruz.

fibk1 diye tanımlanan parametre ise fib(k-1) yani k sayısının 1 eksiği olan sayının fibonacci sayısının karşılığı.

Böyle bir parametre tanımlamamızın gerekçesi ise fibonacci sayısının değerinin kendinden önceki 2 sayının toplamına eşit olmasıdır.

```
# 10. Örnek
def hizlandirici(n, k, fibk, fibk1):
    if k == n:
        return fibk
    else:
        return hizlandirici(n, k+1, fibk+fibk1, fibk)
```

4. SONUÇ VE ÖĞRENİLEN DERSLER

Birbirinden farklı yapıda 10 parçadan oluşan bu menü ödevinde öncelikle unuttuğumuz python işlemlerini hatırlamakla başladık örneği; if-else , else-return ve elif kullanımı ayrıca for döngülerini kullanarak kodlarımda kullanmaya başladık.Örneklerin çözüm aşaması bittiken sonra showmenu ve getmenu kullanarak bu örneklerin çağırılmasını sağladık.Ardından çağırılınca olacak durumların koşullarını belirttik ki kod düzgün çalışabilsin.Tüm bu çalışmanın sonunda ödevi yapmış olmanın yanında Python bilgilerimizi ve kullanım şekillerini tekrar hatırlamış olduk.

5. KAYNAKÇA

https://stackoverflow.com

https://www.w3schools.com/python

https://www.youtube.com