İLERİ PROGRAMLAMA 2 FİNAL

20181129011

**DİLARA KARACA**



**Javascript Ve Node.JS İle Yapay Zeka Uygulamaları Geliştirilebilir Mi?**

Yapay zeka kodlamaları ile problemlerinize tam ve hızlı çözüm bulurken sizlere algoritmik düşünce ile yardımcı olmaktadır. Hangi programlama dili kullanılırsa kullanılırsın hedef iyi analiz edilmelidir. Geliştiriciler, JavaScript kütüphanesi sayesinde gerçek zamanlı olarak ses veya yüz ifadesi gibi kullanıcı girişlerine tepki veren akıllı uygulamalar oluşturmak için kolay bir şekilde kullanabilirler. İçerik önerisi, sesli komutlar veya jestler yoluyla etkileşim, ürünleri veya yerleri tanımlamak için cep telefonu kamerasını kullanma ve kullanıcıya günlük görevlerde yardımcı olmayı öğrenme gibi uygulama örnekleri verebiliriz.

Örneğin, TensorFlow.js (TFJS), özellikle görüntü veya video verileri söz konusu olduğunda, tipik JS uygulamalarınızdan çok daha fazla kaynak kullanabilir. TensorFlow.js, Node.js’de CUDA desteğiyle de çalışır. Bu anlamda, JavaScript ile hem istemcide hem de sunucuda yayınlama konusundaki geleneksel faydasını yapay zeka alanına genişletir.

**Python Ve R Diline Göre JS/Node.JS Artı Ve Eksileri**

 Performans, uygulamanızın yanıt hızını tanımlar ve bu da kodunuzun ne kadar hızlı yürütüleceğine bağlıdır. Python ile Node.js performansını karşılaştırırsak Node.js'in daha hızlı olduğunu göreceğiz. Bu performans, Node.js'nin çok hızlı ve güçlü bir motor olan Chrome'un V8'i üzerine kurulu olmasından kaynaklanıyor. Bu nedenle, Node.js'in hızı Python'a oranla çok daha fazladır; bu, gerçek zamanlı uygulamalar için, örneğin sohbet işlevselliği içeren uygulamalar için daha iyi bir seçim olmasını sağlar.Ayrıca, Python, yoğun bellek kullanan uygulamalarda optimum performans göstermez. Bu nedenle, ağır 3D grafikler kullanan uygulamalar geliştirmek için tercih edilen bir çözüm olmamalıdır.

Programlama dili, uygulama ölçeklenebilirliği üzerinde de etkiye sahiptir. Node.js, iş parçacığı dışında tamamlanan I/O işlemleriyle tek iş parçacıklı eşzamansız mimariyi oluşturur ve bu nedenle engellemez. Bu özellik basit web uygulamalarında sorunsuz Node.js ölçeklenebilirliğini garanti eder, ancak aynı anda birçok işlemi içeren karmaşık uygulamaların geliştirilmesi derinlemesine bilgi, dikkat ve titiz mühendislik araştırmaları gerektirir.Python, sırasıyla asenkron programlamayı desteklemez, ancak eşzamansız işlemeye kolayca erişilebilen koridorları desteklemektedir. Dolayısıyla, mimarisi Nodejs kadar ölçeklenebilir gibi görünmese bile, Python gerekli ölçeklenebilirliğe ulaşılabilecek araçlara sahiptir.

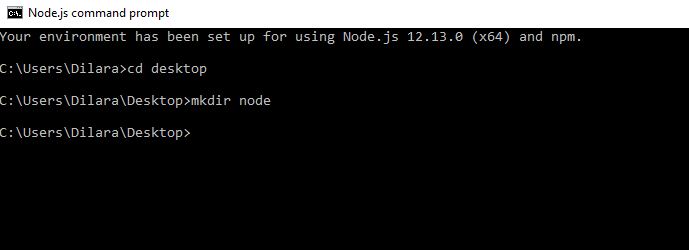
Bir programlama dilini seçerken hata yönetiminin kolaylığı ve şeffaflığı kritik bir konudur. Node.js ve Python, istisnaları atma ve yakalama ile, yani kod yürütülürken oluşan hataları iyi ele alır. Bununla birlikte genel fikir, Python'da istisnaların işlenmesinin daha kolay olduğunu ve hataları ayıklamak ve düzeltmek için daha az zaman almaktadır.

Her şeyden önce istatistiksel bir dil olduğu için R, veri analizi için en iyi yöntemlerden biri olarak kabul edilir. R aklınıza gelebilecek her şey için smorgasbord paketlerine sahiptir, ancak temel istatistiği analizdir .Temel kare'den faktör analizi ve tehlike modellerine kadar, kolay ve sağlamdır. İstatistiklerdeki en büyük isimlerden bazıları R paketleri yaratır ve her ihtiyacınıza yardımcı olacak canlı bir topluluğu vardır.

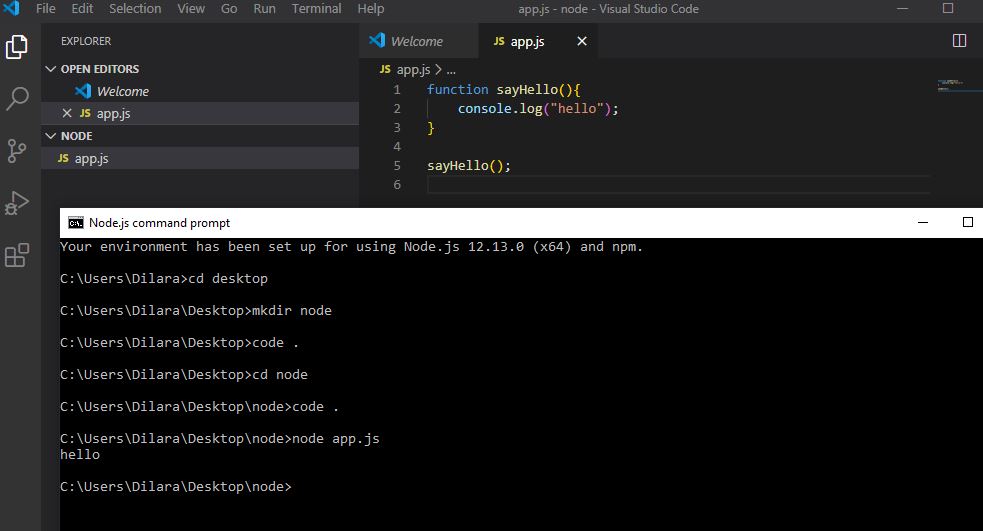
Python öğrenmek daha temiz söz dizimi ve daha kompakt bir kodlamaya sahip olduğu için daha kolay olabilir. Python, aynı sonuca ulaşmak için Node.js'den daha az kod satırı gerektirir. Ayrıca, Python 25 yılı aşkın bir süredir bu piyasanın içinde ve geniş bilgi birikimi içerdiğinden kaynak sıkıntısı çekmezsiniz.Ortak bakış, her dilin artı ve eksileri vardır.

**JS/Node.JS Örnek Uygulaması**

Örnek bir node.js uygulamam için öncelikle bilgisayarımda node.js komut satırından yararlanarak aşağıdaki kodları yazarak masaüstünde node isimli klasör açtım.

****

Daha sonra “code . “ yazdım ve VS Code’ da node konumlanmış şekilde gelmiş oldu.İçerisine app.js isminde bir klasör oluşturdum.Onunda içerisine basit bir function oluşturdum.Ve sonrasında komut satırına “node app.js” yazdım.Ve konsol ekranımda functionum çalışmış oldu.Tarayıcıya bağlı olmadan bir örnek uygulama yapmış oldum.



**Project Euler Cevapları**

**Problem1-Multiples of 3 and 5**

//10'un altında 3 veya 5'in katları olan tüm doğal sayıları listelersek,

//3, 5, 6 ve 9'u alın. Bu katların toplamı 23'tür.

//1000'in altındaki 3 veya 5'in katları;

**function sumOfMultiple(limit) {**

**var i, sum = 0;**

**for (i = 3; i < limit; i += 1) {**

**if (i % 3 === 0 || i % 5 === 0) {**

**sum += i;**

**}**

**}**

**return sum;**

**}**

**console.log(sumOfMultiple(1000));**

**CEVAP:** **233168**

**Problem11-** **Largest product in a grid**

// Bu sorunu benzer bir şekilde başlattım, sonunda ızgarayı bir diziye dönüştürdüm. Her bir değer arasındaki ayrımı yapmak yerine, her iki alan arasındaki ayrımı yaptım, çünkü bunlar 2 basamaklı sayı gruplarından oluşuyor. Izgarayı bir dizi olarak aldıktan sonra, bir sonraki bölüm oldukça basittir. Bitişik sayılar dizisini yeni bir diziye itmek istedim. Bir sayıyı her ittiğimde, dört basamaklı bir dizi oluştu. Böylece buradan, sadece dizi boyunca döngüye girdim ve mevcut yinelemeyi ittim, bu yineleme ile birlikte +1, +2, +3, vb.

**let compareArr = []**

**for (var i = 0; i < ridArr.length; i++) {**

**compareArr.push([ridArr[i], ridArr[i+1], ridArr[i+2], ridArr[i+3]]);**

**compareArr.push([ridArr[i], ridArr[i-1], ridArr[i-2], ridArr[i-3]]);**

**compareArr.push([ridArr[i], ridArr[i-1], ridArr[i+1], ridArr[i+2]]);**

**compareArr.push([ridArr[i], ridArr[i-2], ridArr[i-1], ridArr[i+1]]);**

//Bu bana tüm yatay olarak bitişik basamakları verir. Aynı şeyi dikey olarak da yapabilirim, ancak +1, +2, +3 vb.'yi +20, +40, +60 vb.

**compareArr.push([ridArr[i], ridArr[i+20], ridArr[i+40], ridArr[i+60]]);**

**compareArr.push([ridArr[i], ridArr[i-20], ridArr[i-40], ridArr[i-60]]);**

**compareArr.push([ridArr[i], ridArr[i-20], ridArr[i+20], ridArr[i+40]]);**

**compareArr.push([ridArr[i], ridArr[i-40], ridArr[i-20], ridArr[i+20]]);**

//Çapraz olarak doğru dizini bulmak için biraz aldatıcıydı, çünkü iki yönde çapraz tutmanız gerekiyor:

**compareArr.push([ridArr[i], ridArr[i+21], ridArr[i+42], ridArr[i+63]]);**

**compareArr.push([ridArr[i], ridArr[i-21], ridArr[i-42], ridArr[i-63]]);**

**compareArr.push([ridArr[i], ridArr[i-21], ridArr[i+21], ridArr[i+42]]);**

**compareArr.push([ridArr[i], ridArr[i-42], ridArr[i-21], ridArr[i+21]]);**

**compareArr.push([ridArr[i], ridArr[i+19], ridArr[i+38], ridArr[i+57]]);**

**compareArr.push([ridArr[i], ridArr[i-19], ridArr[i-38], ridArr[i-57]]);**

**compareArr.push([ridArr[i], ridArr[i-19], ridArr[i+19], ridArr[i+57]]);**

**compareArr.push([ridArr[i], ridArr[i-38], ridArr[i-19], ridArr[i+19]]);**

//For döngüsü kapattıktan sonra, diziler dizisinden en büyük ürünü bulmak ve döndürmek için kodluyorum.

**let greatestSum = 0;**

**for (var i = 0; i < compareArr.length; i++) {**

**let tempSum = compareArr[i].reduce(function(a,b){return a\*b;});**

**if (tempSum > greatestSum){**

**greatestSum = tempSum;**

**}**

**}**

**console.log(greatestSum);**

**CEVAP: 70600674**

**Problem21-** **Amicable numbers**

//Belirtilen sayı için faktörleri alır.

 \* @param sayı Faktörlerin alınacağı sayı.

 \* @return Sayının en küçükten en büyüğe dizi olarak faktörleri.

**var getNaturalFactors = function(number){**

**var naturalFactors = factors.getFactors(number);**

**naturalFactors.splice(naturalFactors.length - 1, 1);**

**return naturalFactors;**

**};**

**var getSumOfArray = function(array){**

**var sum = 0;**

**for(var i = 0; i < array.length; i++){**

**sum += array[i];**

**}**

**return sum;**

**}**

**var amicableNumbersSum = 0;**

**for(var i = 1; i < 10000; i++){**

**var naturalFactorsSum = getSumOfArray(getNaturalFactors(i));**

**if(naturalFactorsSum != i){**

**var testI = getSumOfArray(getNaturalFactors(naturalFactorsSum));**

**if(i == testI){**

**amicableNumbersSum += i;**

**console.log('Amicable number found: ' + i);**

**}**

**}**

**console.log(amicableNumbersSum); CEVAP:** **31626**

**Problem31-Coin Sums**

//Karşılık gelen bozuk para hedef tutara sığarsa, bozuk parayı (soldaki hücre) kullanmadan hedefi oluşturma yollarımızın sayısını artı oluşturabileceğimiz yolların sayısını ekleyerek \ text {target } - \ text {columnValue} target − columnValue, yani jetonu hedeften çıkarırsak geri kalanını oluşturmanın yollarının sayısı. İlgili para sığmazsa, sadece hücreden sola değerin üzerine kopyalarız.

**function solution(target) {**

**var coins = [1, 2, 5, 10, 20, 50, 100, 200];**

**var table = new Array(target + 1);**

**for (var i = 0; i <= target; i++) {**

**table[i] = new Array(coins.length);**

**table[i][0] = 1;**

**}**

**for (var i = 0; i <= target; i++) {**

**for (var j = 1; j < coins.length; j++) {**

**table[i][j] = table[i][j - 1];**

**if (coins[j] <= i)**

**table[i][j]+= table[i - coins[j]][j];**

**}**

**}**

**return table[i-1][j-1];**

**} CEVAP: 73682**

**Problem41-Pandigital Prime**

//n basamaklı bir pandigital numarasının rakamları n \* (n + 1) / 2'ye eklenir.

// Bu, 3'ün katına eşit olamaz veya 3, bölen olacaktır.

// Bu, n'nin 4 veya 7 olması gerektiği anlamına gelir.

**public class Problem41 {**

**static long biggest = 0;**

**public static void main (String args[]) {**

**long t1 = System.nanoTime();**

**generate("1", '2');**

**long t2 = System.nanoTime();**

**System.out.println ((t2-t1)/1000000000.0 + " seconds elapsed"); }**

**public static void generate(String str, char digit) {**

**long n = Long.parseLong(str);**

**if (n > biggest && Primes.isPrime(n)) {**

**biggest = n;**

**System.out.println("Biggest pandigital prime so far = " + n);**

**}**

**// System.out.println (str);**

**if (digit <= '7') {**

**generate (digit + str, (char) (digit + 1));**

**for (int i = 1; i <= str.length(); i++) {**

**String prefix = str.substring(0, i);**

**String suffix = str.substring(i);**

**// System.out.println (str + " = " + prefix + " + " + suffix);**

**generate (prefix + digit + suffix, (char) (digit+1)); CEVAP:** **7652413**

**LeetCode Cevapları**

**Problem1-Two Sum**

//Bir tamsayı dizisi verildiğinde, iki sayının dizinlerini belirli bir hedefe eklenecek şekilde döndürün. Her bir girişin tam olarak bir çözümü olacağını varsayabilir ve aynı öğeyi iki kez kullanamazsınız.

**var twoSum = function(nums, target) {**

**let obj = {}**

**let result = false**

**for (let i=0; i<nums.length; i++){**

**if(!obj.hasOwnProperty(nums[i])){**

**obj[nums[i]] = i**

**}**

**if(obj[target-parseInt(nums[i])]!==undefined && i != obj[target-nums[i]] ){**

**return [i,obj[target-nums[i]]]**

**}**

**}};**

**CEVAP: Your input[2,7,11,15]9**

**Output [1,0]**

**Expected [0,1]**

**Problem11-** **Container With Most Water**

//\* @param {number[]} height

\* @return {number}

**var maxArea = function(height) {**

**};**

**var maxArea = function(height) {**

**var lh = 0;**

**var rh = height.length-1;**

**var ans;**

**var max=Math.min(height[lh],height[rh])\*(rh-lh);**

**while(lh < rh){**

**ans = Math.min(height[lh],height[rh])\*(rh-lh);**

**if(ans > max){**

**max=ans;**

**}**

**if(height[rh] > height[lh]){**

**lh++;**

**}else{**

**rh--;**

**} }**

**return max;**

**};**

**CEVAP:** **Your input [1,8,6,2,5,4,8,3,7]**

**Output 49**

**Expected 49**

**Problem21-** **Merge Two Sorted Lists**

//Tek bağlantılı listenin tanımı;

 \* işlev ListNode (val, next) {

 \* this.val = (val === tanımsız? 0: val)

 \* this.next = (next === tanımsız? null: next)

 \* @param {ListNode} l1

 \* @param {ListNode} l2

 \* @return {ListNode}

**const mergeTwoLists = (l1, l2) => {**

**if(!l1 || !l2) return l1 || l2**

**let start = new ListNode()**

**let p = start**

**while(l1 && l2) {**

**if(l1.val < l2.val) {**

**p.next = l1**

**l1 = l1.next**

**} else {**

**p.next = l2**

**l2 = l2.next }**

**p = p.next }**

**p.next = l1 || l2**

**return start.next**

**}**

**CEVAP:Your input [1,2,4] [1,3,4]**

**Output [1,1,2,3,4,4]**

**Expected [1,1,2,3,4,4]**

**Problem31- Next Permutation**

@param (sayı)

@return(void)\*Hiçbir şey döndürmeyin bunun yerine numaralı değiştirin.

**var nextPermutation = function(nums) {**

**if(!nums || nums.length < 2) {**

**return nums;**

**}**

**const swap = (a, b) => {**

**const temp = nums[a];**

**nums[a] = nums[b];**

**nums[b] = temp;**

**};**

**const reverse = (start, end) => {**

**while(start < end) {**

**swap(start, end);**

**start++;**

**end--;**

**} };**

**let i = nums.length - 2;**

**while(i >= 0 && nums[i + 1] <= nums[i]) {**

**i--;**

**}**

**if(i >= 0) {**

**let j = nums.length - 1;**

**while(j >=0 && nums[j] <= nums[i]) {**

**j--;**

**}**

**swap(i, j);**

**}**

**reverse(i+1, nums.length - 1);**

**};**

**CEVAP: Your input[1,2,3]**

**Output[1,3,2]**

**Expected[1,3,2]**

**Problem61- Rotate List**

\* Tekli bağlantılı liste tanımı.

\* işlev ListNode (val, next) {

\* @param {ListNode} kafası

\* @param {sayı} k

\* @return {ListNode}

**var rotateRight = function(head, k) {**

**if(head == null || head.next == null) return head;**

**let n = null;**

**let oldtail = head;**

**for(n = 1; oldtail.next !== null; n++){**

**oldtail = oldtail.next;**

**}**

**oldtail.next = head;**

**let newTail = head;**

**for(let i = 0; i < n - k % n - 1; i++){**

**newTail = newTail.next;**

**}**

**let newHead = newTail.next;**

**newTail.next = null;**

**return newHead;**

**};**

**CEVAP:** **Your input[1,2,3,4,5]2**

**Output[4,5,1,2,3]**

**Expected[4,5,1,2,3]**