1. Analise as seguintes funções escritas em Python e explique o que fazem. Não precisa de explique o que fazem. Não precis	Nome:	##
def f(x,y): if x=1]: return None z = (x(1:1,y) if z==None or y(x(0),z): return x(0) return z Abranzis du umma fumeçà du complanação y , jugando mo tillimo timme du x worned complanar com o antirior a jar ai adiamite. É xulornodo o walor emait xetumite a hatirjagur a fumeçà y. b) def g(x,y): if x=1] and y==1[return [x(0),y(0)]+z La at lista x a y tissueum o emanmo complaimento, a fumeçà xetuma umma lista du tuilos quu não es de dois alementos da ematerna foiço 2. Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "catxeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinas cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u uma lista de cidades ainda não visitadas e uma função de cidado de distrâncias entre cidades, devolve o valor heuristica. Posemvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u uma lista de cidades ainda não visitadas e uma função de cidado de distrâncias netre cidades, devolve o valor recta entre elas. Para não complicar muito o seu trabalho, a heuristica e calculada multiplicando o número transições que ainda faltura e soluturiores, funça escuturio de cidades. la qualitativa monor de complicar muito o seu trabalho, a heuristica e calculada multiplicando o número transições que ainda faltura e soluturiores, funça escuturior de cidades. la qualitativa de cidades.		ī
if x== : return None z = (x(1+1,y)) if z==None or y(x(0),z): return x(0) return x(0) return x(0) return x(0) return x(0) return z Altrania du surma funca com o anturior a for où adiannia. È sulornodo o tralor cmais xesenta a satisfager a funca y. b) def g(x,y): if x== and y== : return if x = and y = : z = g(x(1+1,y(1+1)) if z =None return (x(0),y(0)) +z La as lista x a y tissuerm o comprimento, a funca sulorno surma lista de tuellor que sõe a dois elemento da comercia sulorno surma lista de tuellor que sõe a dois elemento da comercia en determina raminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetir, e termina heuristica. Desenvolva uma funcão que, dad a a cidade actual, a cidade final (que foi trambém a cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar us uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades entre cidades entre cidades entre cidade entre complicar muito o seu trabalho, a heuristica é calculada multiplicando o número transições que ainda faltar are seminarementos estaturos esta	1. Analis	se as seguintes funções escritas em Python e explique o que fazem. Não precisa de expli- imento interno da função.
if x== : return None z = (x(1+1,y)) if z==None or y(x(0),z): return x(0) return x(0) return x(0) return x(0) return x(0) return z Altrania du surma funca com o anturior a for où adiannia. È sulornodo o tralor cmais xesenta a satisfager a funca y. b) def g(x,y): if x== and y== : return if x = and y = : z = g(x(1+1,y(1+1)) if z =None return (x(0),y(0)) +z La as lista x a y tissuerm o comprimento, a funca sulorno surma lista de tuellor que sõe a dois elemento da comercia sulorno surma lista de tuellor que sõe a dois elemento da comercia en determina raminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetir, e termina heuristica. Desenvolva uma funcão que, dad a a cidade actual, a cidade final (que foi trambém a cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar us uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades entre cidades entre cidades entre cidade entre complicar muito o seu trabalho, a heuristica é calculada multiplicando o número transições que ainda faltar are seminarementos estaturos esta	(a)	def f(x,y):
Z = f(x[1:],y)		if x==0:
de l'annie de suma funccè de companação y, jegondo mo tiltimo elemes de x avanta companar com o anterior e pa oi actiante. É sulornado o walor amoit secunte a satisfager a funccè y. b) def g(x,y): if x== and y== : retum if x!= and y= : z = g(x 1; y 1;)) if z!=None: retum (x 0)y 0)+z Le as listas x e y tissuem o amormo comprimento, a funcço sulorna suma lista de tultos que sõe a dois elementos da amesma fasiços 2. Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor excita entre elas. Para não complicar muito o seu trabalho, a heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la ransições que ainda faltar proportiores resoluçares de sur trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluçares de sur a função de cálculo de distâncias entre cidades, devolve o valor entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica e calculada multiplicando o número ransições que ainda faltar proportiores resoluções de sur trabalho, a heuristica duas cidades.		
Abranis de uma funço de comparação y, fegando mo siltimo elemente de x vornos comparar com o anterior e pro ai adiante. É sulornodo o volor cracia secunda a satisfager a funço y. b) def g(x,y):		If z==None or y(x[0],z):
Abronir de uma junco de companação y, jegando mo cillimo elemente de x worns companar com o anterior e pro ai adiante. É xulornado o condo cracia xecemia a satisfager a funçõe y. b) def g(x,y):		
Les lisles x 2 y lissuum o mormo comprimento, a funço xilorna urma lista de liulos que sõe es dois elementos da merma faiços 2 Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetir, e terminar cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessario implementar un heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicia uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades e devolve a distância recebe como entradas duas cidades entre cidades.	_	def g(x,y): if x==[] and y==[]: return [] if x!=[] and y!=[]: z = g(x[1:],y[1:]) if z!=None:
2 Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessaino implementar u uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la rentire elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número as quaisquer cidades. [In terminal control of the	1	
2 Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP atravês de pesquisa A*, é necessaino implementar u heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade insciundades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la receita entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número as quaisquer cidades. [In terminal control of the contro	-32	as tistas x 2 y Tissum o rmomo comprimento, a funço subrina
2 Implemente em Python as seguintes funcionalidades relevantes para sistemas inteligentes: a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP atravês de pesquisa A*, é necessaino implementar u heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade insciundades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la receita entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número as quaisquer cidades. [In terminal control of the contro		a lista de tuflos que sóe os dois elementos da mesma faiços
a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicia uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la receita entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar are seminiferar, final, outration das quaisquer cidades.	mm	
a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicia uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la receita entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número ransições que ainda faltar are seminiferar, final, outration das quaisquer cidades.	mm	
a) Como sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar caminho que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessario implementar u uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em lis recta entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número transições que ainda faltar en particular, não, securior distributo de cálculo de cálculo de distâncias cidades e devolve a distância em lis recta entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número transições que ainda faltar en particular, não, securior distributo de cálculo de distâncias entre cidades en la complicar muito o seu trabalho, a heuristica é calculada multiplicando o número la quaisquer cidades.	Mun	
cidade inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar u heuristica. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inici uma lista de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor heuristica. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em la receta entre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número transições que ainda faltar en soluções que ainda ainda en soluções que ainda faltar en soluções que ainda ainda en soluções que ainda faltar en soluções que ainda ainda en soluções que ainda faltar en soluçõ		emente em Python as seguintes funcionalidades relevantes para sistemas inteligentes
and the state of t	2 Impl	10 sabe, o problems do "crivairo vistanta" crem
nor citya in late; for citya in late; for citya in late; citya in citya; date distancecitya, citya;	a) Com caminh cidade heuristic uma listi heuristic recta ent	no sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar us o que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetir, e termina i inicial. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar un ca. Desenvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicia de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor o ca. A função de cálculo das distâncias recebe como entradas duas cidades de devolve a distância em lini tre elas. Para não complicar muito o seu trabalho, a heuristica é calculades de devolve a distância em lini
dist = distance(city1, city2)	a) Com caminh cidade heuristic uma listi heuristic recta ent	no sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar us o que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina a cidade sem as repetie, e termina a cidade senvolva uma função que, dada a cidade actual, a cidade final (que foi também a cidade inicia a de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor o care a função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em lini tre elas. Para não complicar muito o seu trabalho, a heuristica é calculada multiplicando o número es que ainda faltar escantagar das acumentos.
smallest_dist = dist smallest_dist = dist return (len(not_visited) + 1) * smallest_dist	a) Com caminh cidade heuristic uma listi heuristic recta ent	no sabe, o problema do "caixeiro viajante" (TSP) tem muitas aplicações. Ele consiste em determinar us o que, partindo de uma cidade, passa por um conjunto de outras cidades sem as repetie, e termina i cai. No contexto da resolução do TSP através de pesquisa A*, é necessário implementar un a de cidades ainda não visitadas e uma função de cálculo de distâncias entre cidades, devolve o valor ca. A função de cálculo das distâncias recebe como entradas duas cidades e devolve a distância em lini tre elas. Para não complicar muito o seu trabalho, a heurística é calculada multiplicando-o número es que ainda faltar se segundores, fundo estavisted de seguis um se sur a função de câlculo das distâncias entre cidades.

b) No contexto da implementação de algoritmos de praquias em árvore, considere que os nos da árvore são representados por ruples (Id.State, Associat, How JaPares), em que Id é um identificador do nó, Sastr é o estado desse nó. Aussiliat é o custo scumulado desde o estado micial até Seste, Herr é o custo estimado para chegar à sologão a partir do no 14, e 14Parrar é o identificador do nó pai. Considerando que se está a usar uma estrategia de perquisa A*, programe uma função que, dada a lista de nos abertos (nos que agoardam expansão) e dada uma lara de novos nos (que arabam de ser criados por expansio de outro no), produra a nova lista de nos aberros. m L.O. sgente artificial Comilio necessita de alimento. O seu mundo baseia-se numa geelha de dimensão 10x10. essentedo bisecuros dispersos pela grelha. Há tumbém o Zarpachqui, um agente precluive que pode corner o Comilio. Cada celula da grelha pode estar livre, ou entilo ocupada pelo Comilio, pelo Zarpadaçus ou por um bircom. O Comilio pode mover-se na horizontal ou na vertical usando a acção, mostac, abi, em que des (-1,0,1), des (-1,0,1) e | de| ≠ | de|, alsa pedesdo ultrapassar os limites da grelha. Sempre que o Comillão se dedocu para uma celula com um bascuto, coese-o automaticamente e um novo biscuto suege numa posição alestária da grelha. Quando o Comilão se move para uma posição adjacente à do Zarpadaqui, este come-to. O Comilio consegue saber unde está o predador bem como os biscotros, e consegue amán saber qual a cellula que corresponde so centro geométrico da distribuição dos biscotos. Sendo insactivel, o seu objectivo é ir camerado o maior número possível de biscoitos, e cluro, fogir la garna do prechder. Sempre que o Comilio sem um biscoino em posição adiacente a si proprio e não adiacente ao predador, desloca-se para essa posição. Caso contration tenta aproximar-se do centro geométrico da distribuição de biscostos. a) Identifique e caracterize um conjunto de perdicados que possan ser usados para discrever altaques em que se energatre o Comilio. Identifique também as variaveis de estado, caso seja necessárias.

