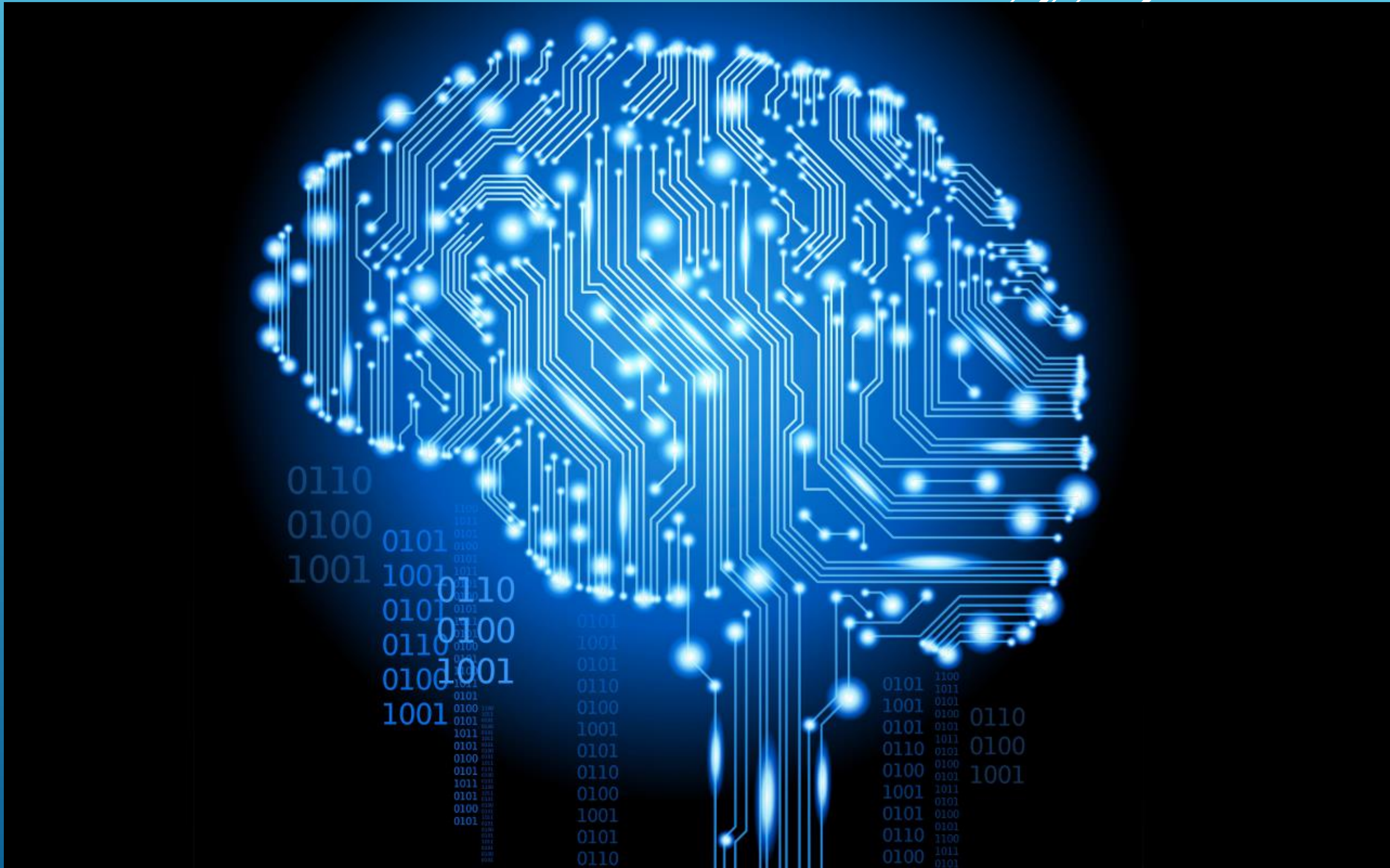
ENGENHARIA INFORMÁTICA

INTELIGÊNCIA ARTIFICIAL



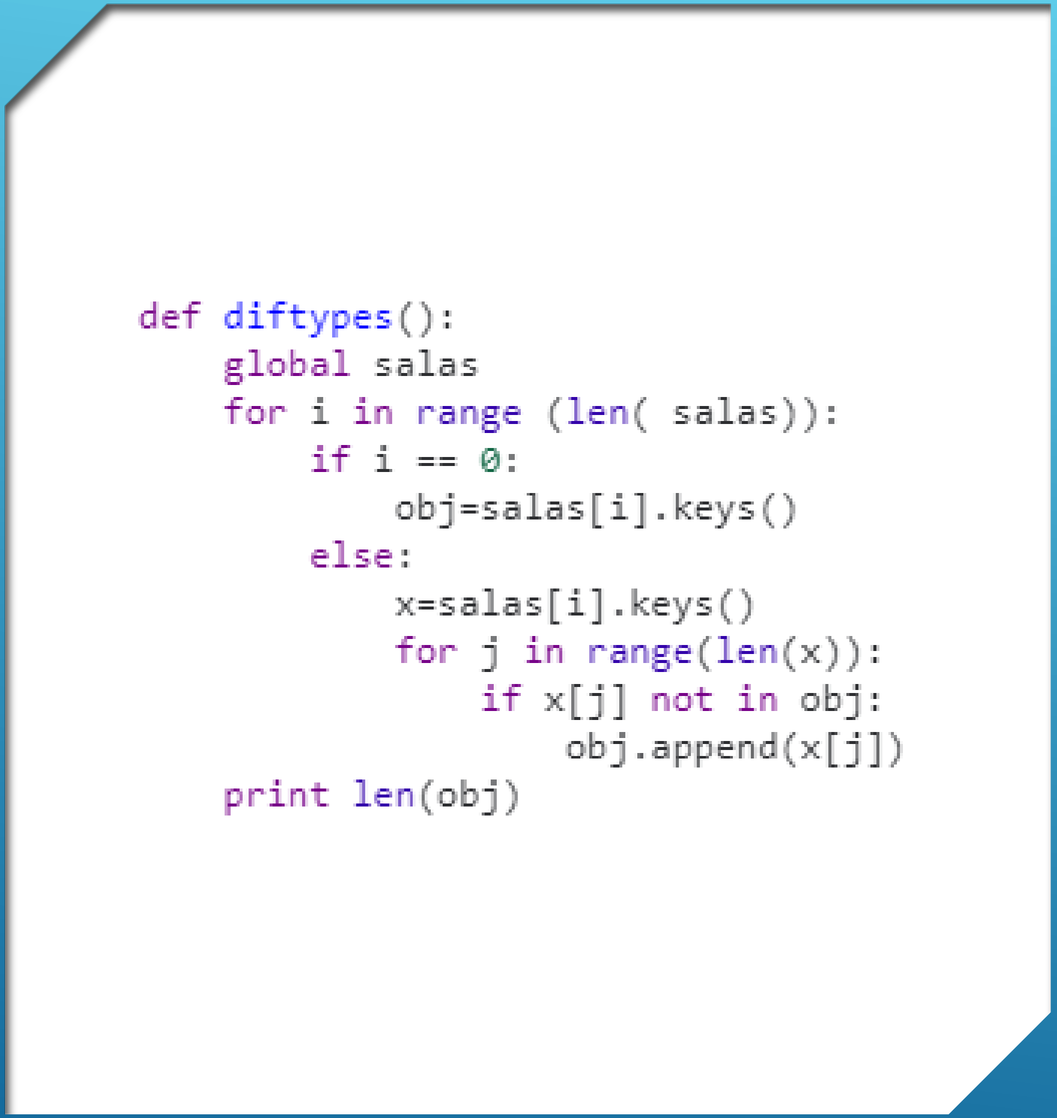
Elaborado por:

Ricardo Domingos, nr 37906

João Domingos, nr 38023

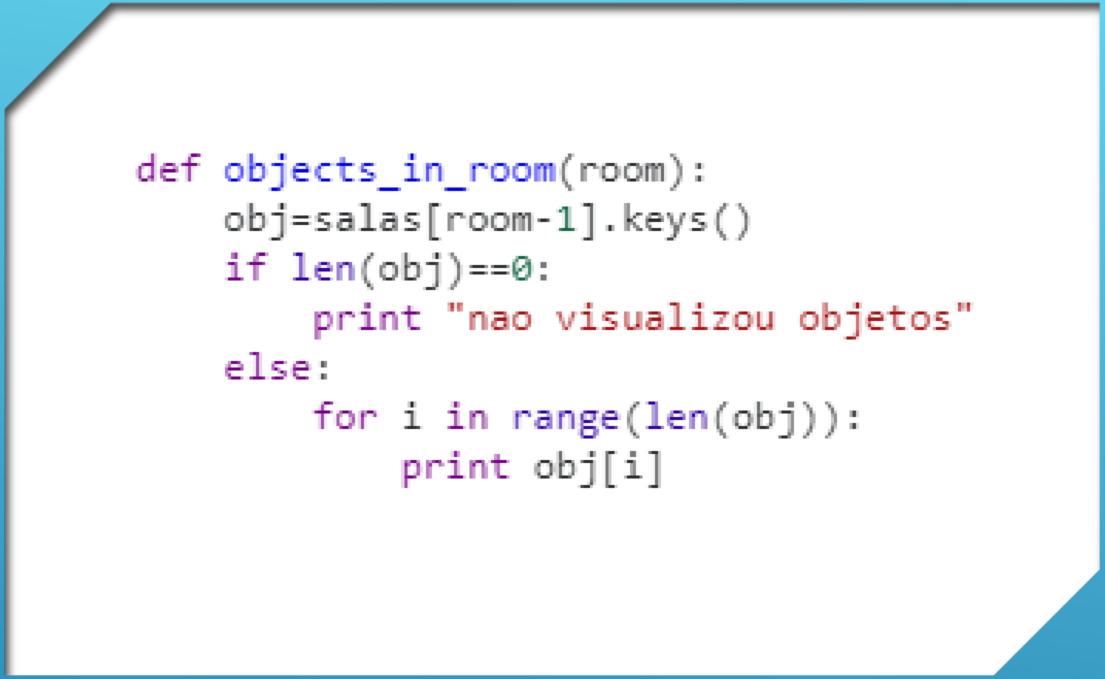
Professor:

Luís Alexandre



```
def diftypes():
    global salas
    for i in range (len( salas)):
        if i == 0:
            obj=salas[i].keys()
        else:
            x=salas[i].keys()
            for j in range(len(x)):
                if x[j] not in obj:
                    obj.append(x[j])
    print len(obj)
```

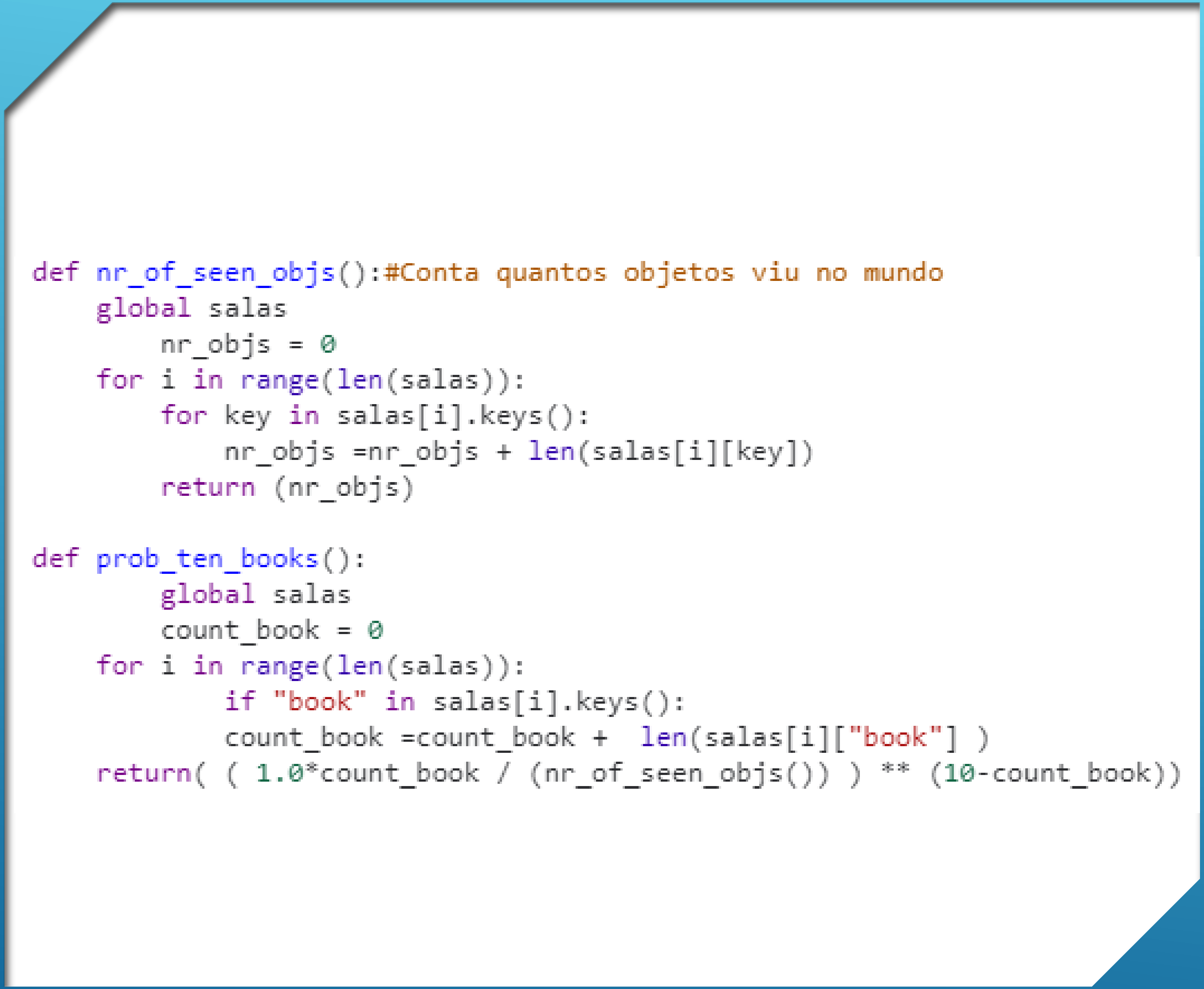
1. HOW MANY DIFFERENT TYPES OF OBJECTS DID YOU RECOGNIZE UNTIL NOW?



```
def objects_in_room(room):
    obj=salas[room-1].keys()
    if len(obj)==0:
        print "nao visualizou objetos"
    else:
        for i in range(len(obj)):
            print obj[i]
```

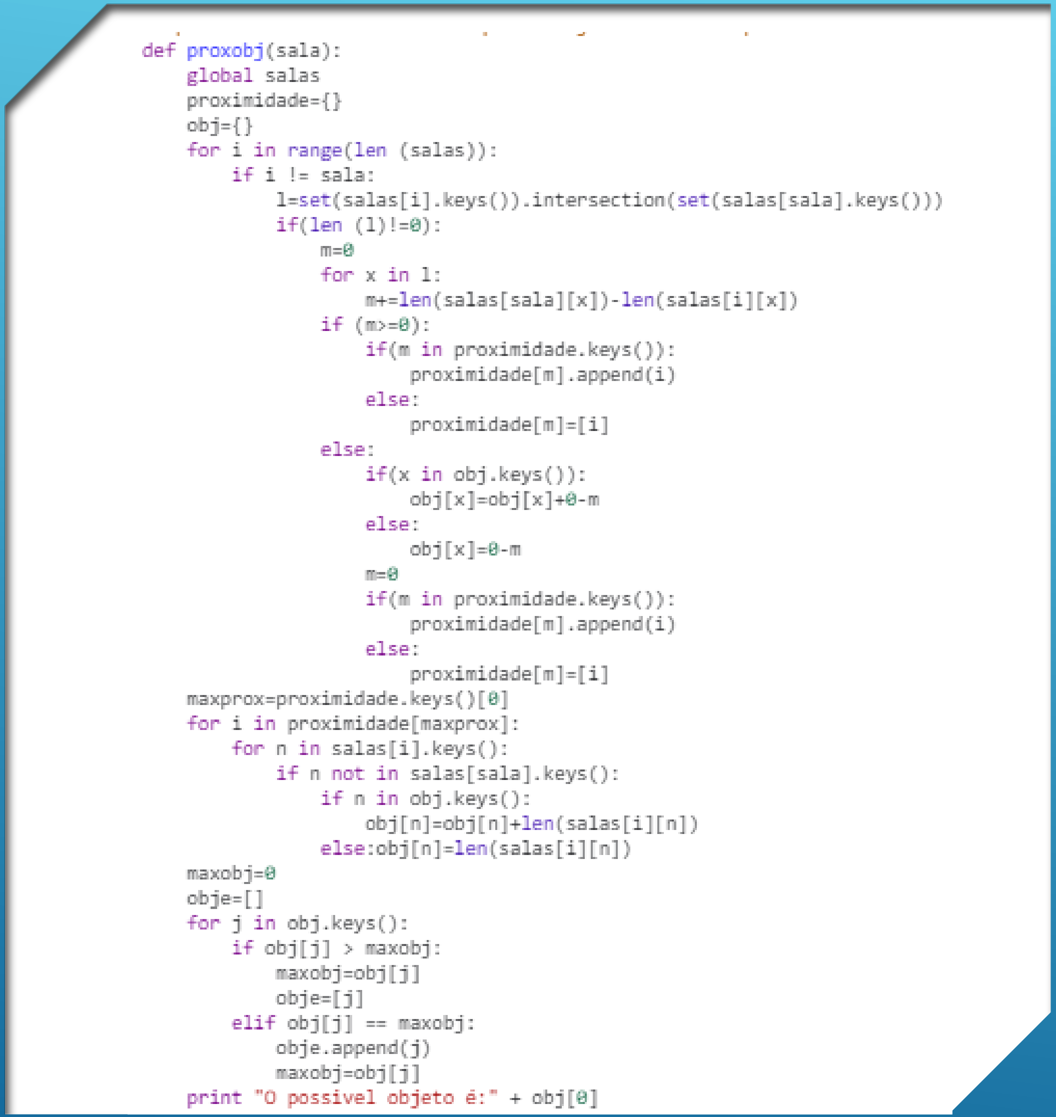
2. WHICH OBJECTS WERE IN THE ROOM YOU VISITED BEFORE THIS ONE?

3. WHAT IS THE PROBABILITY OF FINDING 10 BOOKS IN THIS WORLD?

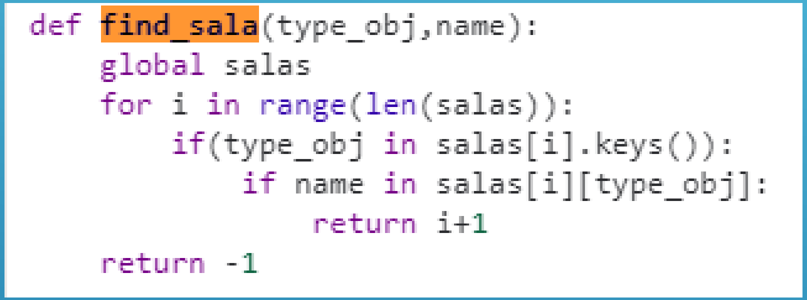


```
def nr_of_seen_objs():#Conta quantos objetos viu no mundo
    global salas
        nr_objs = 0
    for i in range(len(salas)):
        for key in salas[i].keys():
            nr_objs =nr_objs + len(salas[i][key])
        return (nr_objs)

def prob_ten_books():
        global salas
        count_book = 0
    for i in range(len(salas)):
            if "book" in salas[i].keys():
            count_book =count_book +  len(salas[i]["book"] )
    return( ( 1.0*count_book / (nr_of_seen_objs()) ) ** (10-count_book))
```

4. WHAT TYPE OF OBJECT DO YOU THINK IS THE ONE WITHOUT IDENTIFICATION, THAT APPEARS CLOSE TO JOE?

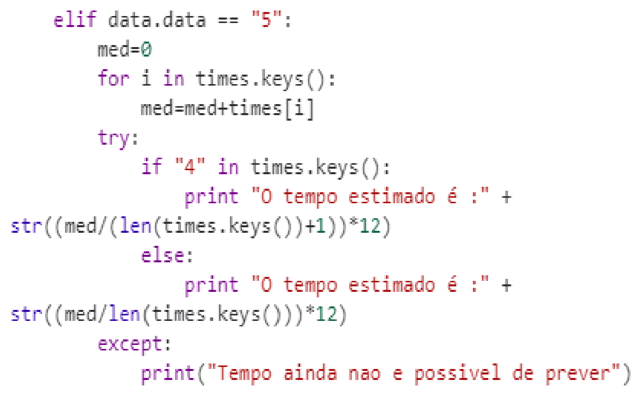


```python
def proxobj(sala):
    global salas
    proximidade={}
    obj={}
    for i in range(len (salas)):
        if i != sala:
            l=set(salas[i].keys()).intersection(set(salas[sala].keys()))
            if(len (l)!=0):
                m=0
                for x in l:
                    m+=len(salas[sala][x])-len(salas[i][x])
                if (m>=0):
                    if(m in proximidade.keys()):
                        proximidade[m].append(i)
                    else:
                        proximidade[m]=[i]
                else:
                    if(x in obj.keys()):
                        obj[x]=obj[x]+0-m
                    else:
                        obj[x]=0-m
                    m=0
                    if(m in proximidade.keys()):
                        proximidade[m].append(i)
                    else:
                        proximidade[m]=[i]
    maxprox=proximidade.keys()[0]
    for i in proximidade[maxprox]:
        for n in salas[i].keys():
            if n not in salas[sala].keys():
                if n in obj.keys():
                    obj[n]=obj[n]+len(salas[i][n])
                else:obj[n]=len(salas[i][n])
    maxobj=0
    obje=[]
    for j in obj.keys():
        if obj[j] > maxobj:
            maxobj=obj[j]
            obje=[j]
        elif obj[j] == maxobj:
            obje.append(j)
            maxobj=obj[j]
    print "O possivel objeto é:" + obj[0]
```

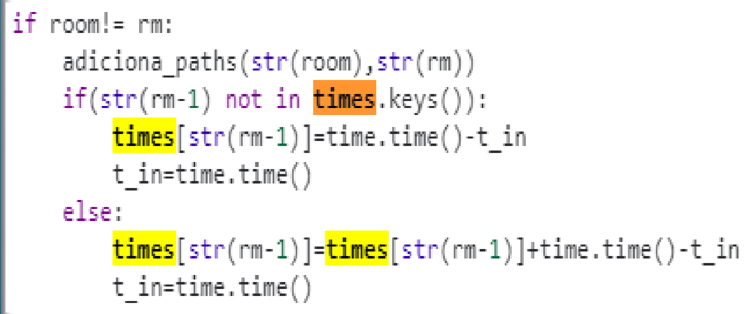


```python
def find_sala(type_obj,name):
    global salas
    for i in range(len(salas)):
        if(type_obj in salas[i].keys()):
            if name in salas[i][type_obj]:
                return i+1
    return -1
```

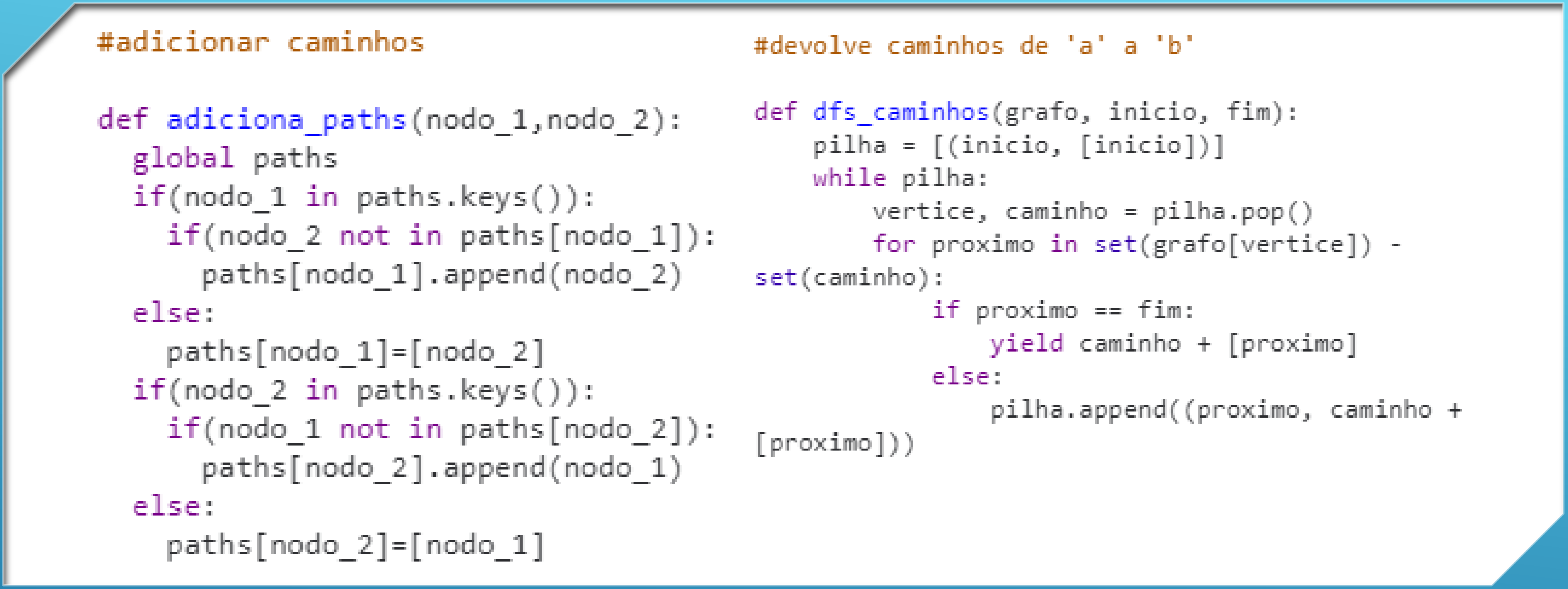
5. WHAT IS YOUR ESTIMATE OF THE TIME IT TAKES TO VISIT ALL THE ROOMS?



```
    elif data.data == "5":
        med=0
        for i in times.keys():
            med=med+times[i]
        try:
            if "4" in times.keys():
                print "O tempo estimado é :" +
str((med/(len(times.keys())+1))*12)
            else:
                print "O tempo estimado é :" +
str((med/len(times.keys()))*12)
        except:
            print("Tempo ainda nao e possivel de prever")
```



```
if room!= rm:
    adiciona_paths(str(room),str(rm))
    if(str(rm-1) not in times.keys()):
        times[str(rm-1)]=time.time()-t_in
        t_in=time.time()
    else:
        times[str(rm-1)]=times[str(rm-1)]+time.time()-t_in
        t_in=time.time()
```

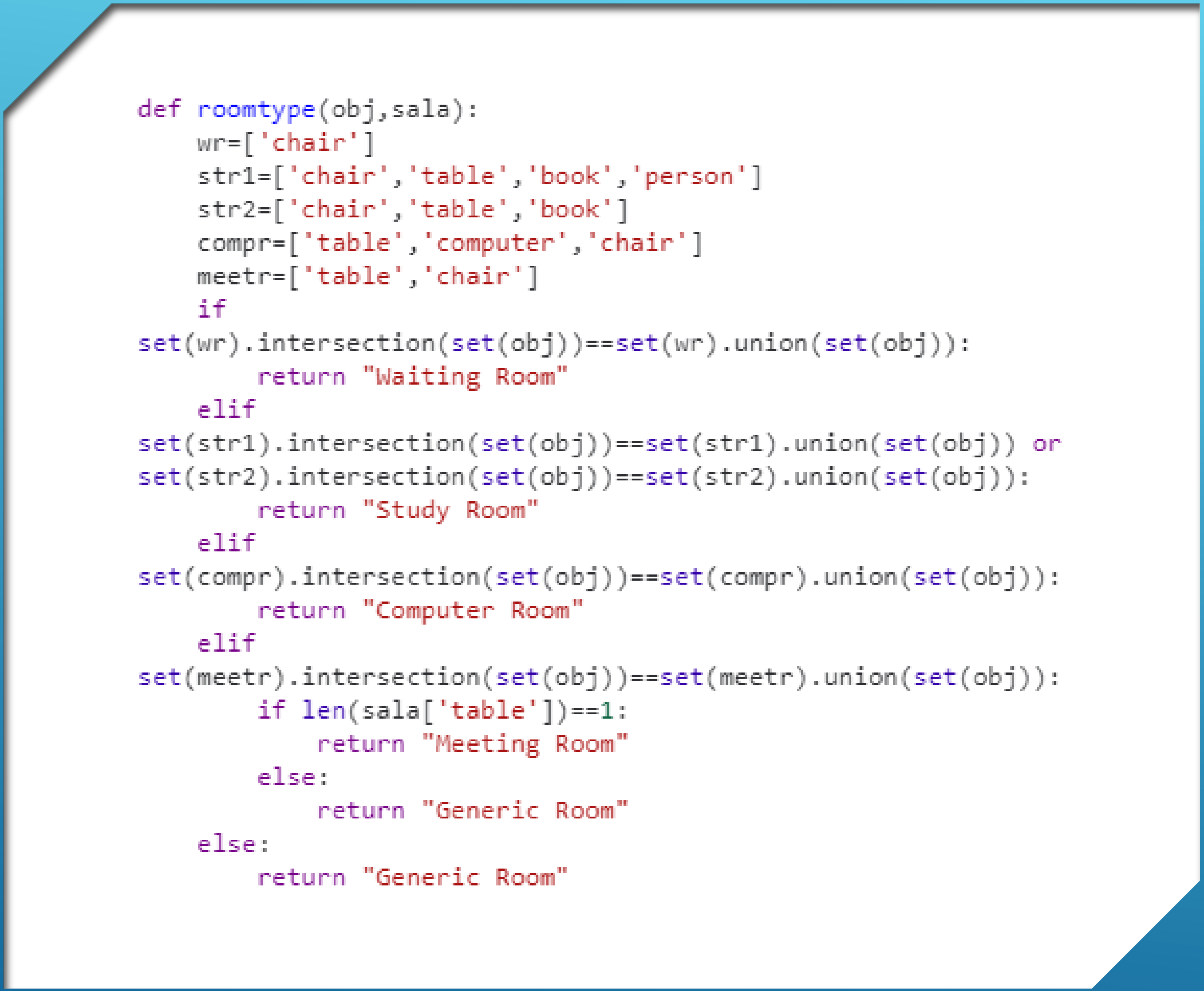


```
#adicionar caminhos

def adiciona_paths(nodo_1,nodo_2):
  global paths
  if(nodo_1 in paths.keys()):
    if(nodo_2 not in paths[nodo_1]):
      paths[nodo_1].append(nodo_2)
  else:
    paths[nodo_1]=[nodo_2]
  if(nodo_2 in paths.keys()):
    if(nodo_1 not in paths[nodo_2]):
      paths[nodo_2].append(nodo_1)
  else:
    paths[nodo_2]=[nodo_1]
```

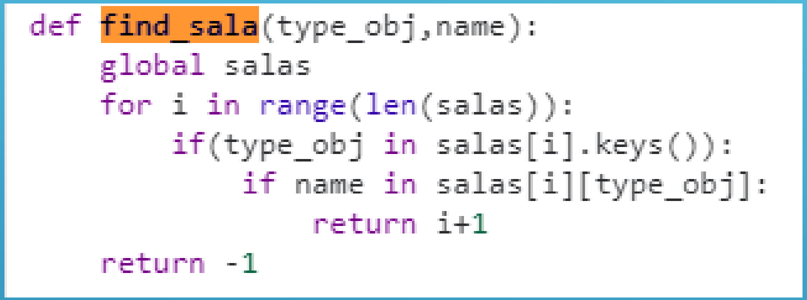
```
#devolve caminhos de 'a' a 'b'

def dfs_caminhos(grafo, inicio, fim):
    pilha = [(inicio, [inicio])]
    while pilha:
        vertice, caminho = pilha.pop()
        for proximo in set(grafo[vertice]) -
set(caminho):
            if proximo == fim:
                yield caminho + [proximo]
            else:
                pilha.append((proximo, caminho +
[proximo]))
```

6. HOW MANY DIFFERENT PATHS CAN YOU TAKE TO GO FROM THE CURRENT ROOM, BACK TO THE START ROOM?

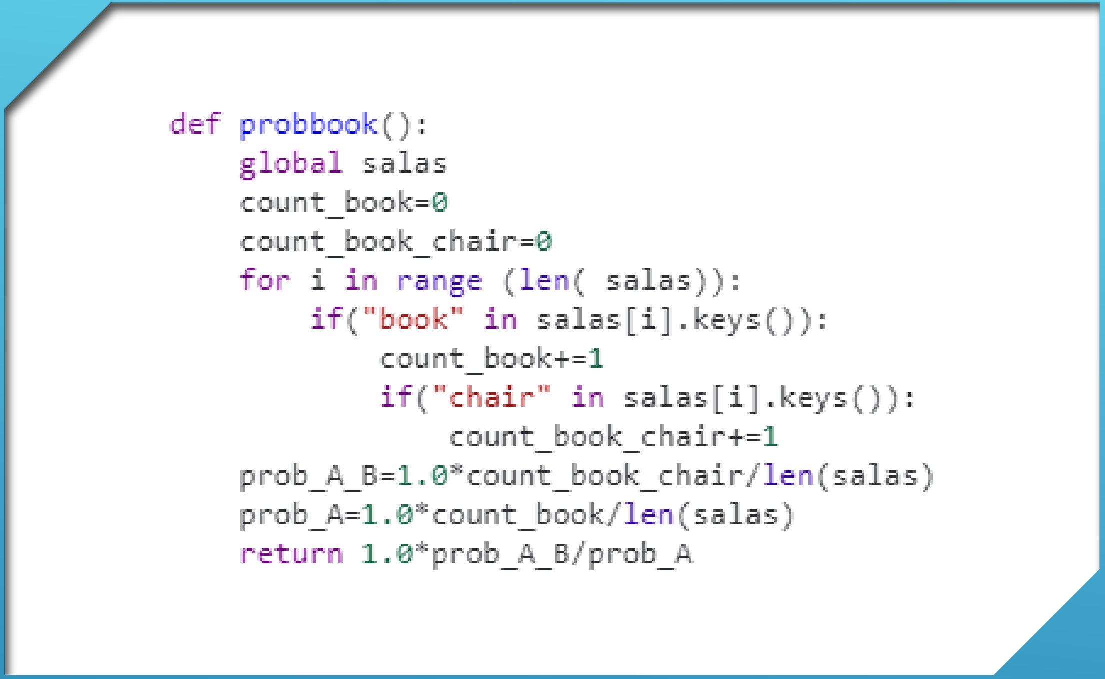
7. IN WHAT TYPE OF ROOM IS MARY IN?



```
def roomtype(obj,sala):
    wr=['chair']
    str1=['chair','table','book','person']
    str2=['chair','table','book']
    compr=['table','computer','chair']
    meetr=['table','chair']
    if
set(wr).intersection(set(obj))==set(wr).union(set(obj)):
        return "Waiting Room"
    elif
set(str1).intersection(set(obj))==set(str1).union(set(obj)) or
set(str2).intersection(set(obj))==set(str2).union(set(obj)):
        return "Study Room"
    elif
set(compr).intersection(set(obj))==set(compr).union(set(obj)):
        return "Computer Room"
    elif
set(meetr).intersection(set(obj))==set(meetr).union(set(obj)):
        if len(sala['table'])==1:
            return "Meeting Room"
        else:
            return "Generic Room"
    else:
        return "Generic Room"
```



```
def find_sala(type_obj,name):
    global salas
    for i in range(len(salas)):
        if(type_obj in salas[i].keys()):
            if name in salas[i][type_obj]:
                return i+1
    return -1
```



```python
def probbook():
    global salas
    count_book=0
    count_book_chair=0
    for i in range (len( salas)):
        if("book" in salas[i].keys()):
            count_book+=1
            if("chair" in salas[i].keys()):
                count_book_chair+=1
    prob_A_B=1.0*count_book_chair/len(salas)
    prob_A=1.0*count_book/len(salas)
    return 1.0*prob_A_B/prob_A
```

8. WHAT IS THE PROBABILITY OF FINDING A CHAIR IN A ROOM GIVEN THAT YOU ALREADY FOUND A BOOK IN THAT ROOM?