//create dataframe containg additional column with the type of each oxygen
In order to do this we must figure out if it is from the H2O or form the slab
In order to do this on demand, we must first determine H2O, then all others are from slabs and there are sub categories of them and so on.
In order to intially calculate if from H2O we must first have the distance between
that O and a hydrogen is == H2O_L. If it is then we have to look at that hydrogen
see if it has exaclty one other oxygen that is equal to H2O_L. If so then it is an O from H2O. In order to find both of those distances, we must first calculate the distance between every hydrogen and that Oxygen, then calculate the distance between that Hydrogen and every other oxygen.

Pass X,Y,Z directly form df into dist function to calc. dsit w/o needing to creat O1list, O2list, Hlist.

```
func():
      for i (nstart, nstop):
            find 'O'
```

Want to search first for the Oxygens dist form hydrogen than for hydrogen dist from second oxygen.

From the first serch we go through the time step and find a O and return its index. We then go through
the time step and find a H that is H2O_L from that O, and we return its index. Then, find a O != to
previous O and return its index. Then, search for another O != to O1 or O2, returning it's index and
determining the molecule to be an H3O. So everytime, we are looking through the time step and looking
for an atom that is a distance away from the second atom (or we could do first putting H first) and we
break the loop at that point when we find it and return the index of that atom. If we had one general
function to do this it would take in an atom type, indicate wether it was first or subsequent (in order to
determine to use H2O_L dist or not), in other words you could have just an index getter method. Declaring
beforehand, the Species to be searched for, the the order, and the amount to define a molecule. So, you
could say, we have O and H, we could have 1 & 1, 2 & 1, or >3 & 1. Then call a method which recieves, the
instructions for the molecule, that being the consituent species, the amount of each species, and how they
are arranged, that is, to which atom is another connected and by what distance. Once we have this information
we can then have a method which checks the necessary condition in relation to the previous atom (ie. if
O != prevOindex) and then calls the getter method again for the subsequent atom. Of course we will need to
know how to determine that order to begin with so this or a mehtod will have to compare the queried atom type
to a reference of molecules of which that atom type is a constituent and by searching the distances around it,
determine which molecule it is a candidate for. After this a particular sequnce must be followed to follow the
structure of that particular molecule from whatever starting atom we may be happen to be qureing from. So, it
may look like this:
molstructure: H2O
shellMethod():
if(species A,B,C,D,E,F):
then find which mol may belong to

```
test by comparing to adj. atoms
find which mol it is from if H2O then must test if H3O, H2O, or HO
do this by testing if
H2O: H=1 O=>3 H2O_L=1.00
getatom(): returns O
if(atomtype == 'O'):
check to see what is around said atom
if there is an 'H' at dist. H2O_L from this atom then it's from H30, H2O, or HO
so, if(getter() == 'H' and dist('O', 'H') == H2O_L):
then look for a second O dist H2O_L from H
if(getter() == ('O' != last O) and dist('O', 'H') == H2O_L):
then look for another. If not then it's HO so call setter.
if(getter() == ('O' != last O and lastlast O) and dist('O', 'H') == H2O_L):
then its H3O; setter(); If not then it's H2O; setter().
if(getter() == atomtype != prevAtomtype() and dist() == Lgetter(atomtype)):
```