* In Python orice este un obiect. Crearea unei variabile include intai crearea obiectului si stabilirea locatiei lui (pentru ceea ce se afla dupa egal) si apoi atribuirea variabilei acelui nou obiect, adica variabila se leaga de locatia lui. Daca mai apoi variabila este egalata cu altceva, ea iarasi ia pozitia deja noului obiect. Exact asa se face si la functii. Parametrii formali iau locatiile parametrilor actuali, si daca se egaleaza cu altceva, deja isi schimba locatiile. Totusi, daca modificam elemente din namespace la obiecte, de ex trimitem functiei un obiect Object si apoi in functie facem o modificare a unui element din Object, de ex Object.result = 100, atunci parametrul formal nu va lua alt obiect, ci va modifica chiar valoarea lui result din Object.
* In Java, totul are loc exact ca in Python, cu exceptia faptului ca datele primitive, exact ca in C++, odata ce ocupa o locatie, nu o mai schimba. Ele mereu doar copie valorile altor variabile daca sunt egalate, nu locatia! Variabilele ce sunt de tip primitiv isi aleg locatia chiar la declararea lor, fie ca sunt sau nu egalate cu ceva, dar obiectele trebuie sa foloseasca new pentru a aloca memorie. Variabilele de tip obiect intai creaza obiectul dupa egal sau il obtin si abea apoi leaga variabila de obiect.
* In C++ o variabila sau obiect mereu doar copie valoarea de la o locatie, fie ca e vorba de date primitive sau obiecte. Semnul = mereu va face ca variabila sau obiectul din stanga doar sa copie valoarea celui din dreapta. Totusi, daca se pune si & doar cand se declara variabila sau obiectul si se egaleaza cu ceva, atunci variabila sau obiectul chiar se va lega de acea locatie, nu doar va copia valoarea. Totusi, nu e valid asa ceva:

int a = 100;

int &b;

b = a;

deoarece variabilele/obietele la declarare deja isi aleg o locatie si in cazul dat, b nu isi poate alege o locatie la declarare daca are simbolul & si nu I se spune ce locatie trebuie sa ia la declarare. In C++ orice variabila sau obiect isi creaza o zona de memorie chiar la declarare,de aceea obiectele nu necesita operatorul new la creare. Pointerii pot crea automat o zona de memorie, sau cream noi cu new mai multe zone pentru ei. Mai mult ca atat, la declararea variabilelor ce stocheaza tipuri de date primitive, ele nu mai copie valorile lor, ci le pun direct:

int a = 10

10 nu are o locatie separata de a si se pune direct in zona de memorie a lui a

Totusi, in C++ nu putem schimba obiectul de care se leaga variabila obiect de cate ori vrem, ci putem doar decide daca aceasta la declarare se leaga de un obiect sau isi creaza propriul.In rest, variabila doar va copia datele celeilalte de dupa = . Aici deja pointerii pot doar sa schimbe locatia spre care pointeaza de cate ori vor.Chiar si functiile care returneaza o locatie vor face ca variabila obiect ce deja are o locatie determiata sa copie pur si simplu obiectul returnat, fara a face ceva cu locatia lui.

Functiile, odata ce termina executarea, isi curate namespaceul, adica rup legatura cu obiectul functie al lor, dar obiectul functie de care au fost ele legate continua sa existe mai departe in memorie si va exista mereu.