Static Scoping:
When the scope of all names can be known by looking at the text of the program. I.e. at compile time.
aka Lexical Scoping.

Dynamic scoping:
When the scope of a name can only be determined at run time, dictated by flow of execution of program.

Referencing Environment:
Complete set of bindings at a given point in a program is called a referencing environment.

Can be determined by using stack frames and static/dynamic links. But there are some other
issues...due to aliasing, overloading, polymorphism, first-class values, etc.
5 Aliasing, Overloading, Polymorphism, and First-class Values

Monday, September 28, 2020 9:00 AM

Example Java:

```java
int [] a = {1, 2, 3, 4};
n = 10;
int [] b;
b = a;
b[0] = 10;
```

Aliasing: When one or more names in a program refer to the same object at the same point in a program, a and b are aliases.

Common in all PLs where we have reference/pinlks.

Example C++:

```cpp
void f (int &a, int &b) {
    // a and b refer to the same object.
    f(x, x);
    f(l[i], l[j]);
}
```

C:

```c
int *n;
void main () {
    f(n);
    void f (int *a) {
        // a and n are aliases.
    }
}
```
Java & Python
Overloading:
When a symbol in a program (PL) has more than one meaning.

```
public class C {
    static int max (int a, int b) {
        // 2 different meanings!
    }
    // static variables
    static int x;
    int f() {
        // static int x;
        return 3;
    }
    void f() {
        return 3;
    }

    static Area code {
        // static Area code
        code
    }

    // Arithmetic operators
    c = a + b;  // + is overloaded
    Widget w1, w2, w3;
    w1 = w2 + w3;
}
```