

## Reference

# White Space

White Space is the use of **spaces, tabs, and blank lines** to visually organize code. Programmers use White Space since it can group code into sensible, readable chunks without affecting how the code is read by a machine. For example, a program that would run a robot forward for 2 seconds, and then backward for 4 seconds, could look like either of these:

### Program Without White Space

```
task main()
{
  bMotorReflected[port2]=1;
  motor[port3]=127;
  motor[port2]=127;
  wait1Msec(2000);
  motor[port3]=-127;
  motor[port2]=-127;
  wait1Msec(4000);
}
```

### Program With White Space

```
task main()
{
  bMotorReflected[port2]=1;
  motor[port3]=127;
  motor[port2]=127;
  wait1Msec(2000);

  motor[port3]=-127;
  motor[port2]=-127;
  wait1Msec(4000);
}
```

Both programs will perform the same, however, the **second uses white space** to organize the code to **separate the program's two main behaviors**: moving forward and moving in reverse. In this case, line breaks (returns) were used to vertically segment the tasks. Horizontal white space characters like spaces and tabs are also important. Below, white space is used in the form of indentations to indicate which lines are within which control structures (task main, while loop, if-else statement).

### Program Without White Space

```
task main()
{
bMotorReflected[port2]=1;
while(true)
{
if(SensorValue(touch)==0)
{
motor[port3]=127;
motor[port2]=127;
}
else
{
motor[port3]=-127;
motor[port2]=-127;
}
}
}
```

### Program With White Space

```
task main()
{
  bMotorReflected[port2]=1;
  while(true)
  {
    if(SensorValue(touch)==0)
    {
      motor[port3]=127;
      motor[port2]=127;
    }
    else
    {
      motor[port3]=-127;
      motor[port2]=-127;
    }
  }
}
```