|  |  |
| --- | --- |
|  |  |
| **Unit:** Methodology | **Turn In List:** **1. Terms, 2. Post timeline, and 3. Grid** |
| *“I can create and use many data types in a simple computer program.”* | |

**Data Types and Variables: A look at the major data types for modern languages**

**Content Objectives:** Students will be able to declare, initialize and assign variable for a program.

|  |
| --- |
| **Starter Activity** |

float grade = random(0, 101);

if (grade >= 94) {

println("Assign letter grade A." + grade);

} else if (grade >= 89 ) {

println("Assign letter grade A-." + grade);

}

else if (grade >= 87) {

println("Assign letter grade B+." + grade);

}

else if (grade >= 83) {

println("Assign letter grade B." + grade);

} else if (grade >= 80) {

println("Assign letter grade B-." + grade);

}else if (grade >= 77) {

println("Assign letter grade C+." + grade);

}else if (grade >= 73) {

println("Assign letter grade C." + grade);

}else if (grade >= 70) {

println("Assign letter grade C-." + grade);

}else if (grade >= 67) {

println("Assign letter grade D+." + grade);

}else if (grade >= 63) {

println("Assign letter grade D." + grade);

}else if (grade >= 60) {

println("Assign letter grade D-." + grade);

}else {

println("Assign letter grade F." + grade);

}

|  |  |
| --- | --- |
| **Key Terms:** | |
| Interpreted Language | A programming language that executes instructions directly without compiling into machine-language instructions. Most commonly used by Java and C# |
| Compiled Language | Translators that generate machine code from source code, opposite of interpreted language |
| Low Level Language | A programming language that provides little or no abstraction from a computer’s instruction set architecture |
| High Level Language | A programming language that allows the programmer to write programs that are more or less independent than the type normally used on the computer |
| Execute | The way a computer performs the instructions of a computer program |
| Identifiers | A string of alphanumeric characters that begins with an alphabetic character or underscore character that helps the program identify things such as variables, structures, etc |
| Declare Variables | This is what declares whether something is a variable or not, in a data type such as Boolean, Long, or Decimal |
| Initialize Variables | The initial value for a data object, and helps the variable initialize what it is doing |
| Assign Variables | Helps assign the variable for what it is doing or needs to do, by using an assign statement |

|  |
| --- |
| **Assignment:** |
| For each data type give the following information. Use the Processing reference as an aid (note that all data types follow the java standard.) You may write N/A where applicable.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **Memory Used** | **Possible Values (Min)** | **Possible Values (Max)** | **Purpose** | **Syntax** | | boolean | One bit (Not specified by Java |  |  |  |  | | byte | 8 bits | -128 | 127 |  | byte var | | char | 2 bytes |  |  |  |  | | color | 32 bits |  |  |  |  | | double |  |  |  |  |  | | float |  |  |  |  |  | | int |  |  |  |  |  | | long |  |  |  |  |  |     Create a new processing project with a medium gray canvas size of 1000 x 1000 pixels and draw a black grid on the first made up of lines at every 100 pixels vertically and horizontally. Provide text labels (100, 200, etc.) on the left margin and top margin.  void setup() {  size(1000,1000);  }  void draw(){  background(55,55,55);  stroke(0);  //horizontal lines  line(0,100,1000,100);  line(0,200,1000,200);  line(0,300,1000,300);  line(0,400,1000,400);  line(0,500,1000,500);  line(0,600,1000,600);  line(0,700,1000,700);  line(0,800,1000,800);  line(0,900,1000,900);  //vertical lines  line(100,0,100,1000);  line(200,0,200,1000);  line(300,0,300,1000);  line(400,0,400,1000);  line(500,0,500,1000);  line(600,0,600,1000);  line(700,0,700,1000);  line(800,0,800,1000);  line(900,0,900,1000);  //horizontal text  textSize(10);  text("0,0",0,15);  text("0,100", 0,115);  text("0,200", 0,215);  text("0,300",0,315);  text("0,400",0,415);  text("0,500",0,515);  text("0,600",0,615);  text("0,700",0,715);  text("0,800",0,815);  text("0,900",0,915);  //vertical text  text("100,0",100,15);  text("200,0",200,15);  text("300,0",300,15);  text("400,0",400,15);  text("500,0",500,15);  text("600,0",600,15);  text("700,0",700,15);  text("800,0",800,15);  text("900,0",900,15);  //inside grid text 100 column  text("100,100",105,115);  text("100,200",105,215);  text("100,300",105,315);  text("100,400",105,415);  text("100,500",105,515);  text("100,600",105,615);  text("100,700",105,715);  text("100,800",105,815);  text("100,900",105,915);  //inside grid text 200 column  text("200,100",205,115);  text("200,200",205,215);  text("200,300",205,315);  text("200,400",205,415);  text("200,500",205,515);  text("200,600",205,615);  text("200,700",205,715);  text("200,800",205,815);  text("200,900",205,915);  //inside grid text 300 column  text("300,100",305,115);  text("300,200",305,215);  text("300,300",305,315);  text("300,400",305,415);  text("300,500",305,515);  text("300,600",305,615);  text("300,700",305,715);  text("300,800",305,815);  text("300,900",305,915);  //inside grid text 400 column  text("400,100",405,115);  text("400,200",405,215);  text("400,300",405,315);  text("400,400",405,415);  text("400,500",405,515);  text("400,600",405,615);  text("400,700",405,715);  text("400,800",405,815);  text("400,900",405,915);  //inside grid text 500 column  text("500,100",505,115);  text("500,200",505,215);  text("500,300",505,315);  text("500,400",505,415);  text("500,500",505,515);  text("500,600",505,615);  text("500,700",505,715);  text("500,800",505,815);  text("500,900",505,915);  //inside grid text 600 column  text("600,100",605,115);  text("600,200",605,215);  text("600,300",605,315);  text("600,400",605,415);  text("600,500",605,515);  text("600,600",605,615);  text("600,700",605,715);  text("600,800",605,815);  text("600,900",605,915);  //inside grid text 700 column  text("700,100",705,115);  text("700,200",705,215);  text("700,300",705,315);  text("700,400",705,415);  text("700,500",705,515);  text("700,600",705,615);  text("700,700",705,715);  text("700,800",705,815);  text("700,900",705,915);  //inside grid text 800 column  text("800,100",805,115);  text("800,200",805,215);  text("800,300",805,315);  text("800,400",805,415);  text("800,500",805,515);  text("800,600",805,615);  text("800,700",805,715);  text("800,800",805,815);  text("800,900",805,915);  //inside grid text 00 column  text("900,100",905,115);  text("900,200",905,215);  text("900,300",905,315);  text("900,400",905,415);  text("900,500",905,515);  text("900,600",905,615);  text("900,700",905,715);  text("900,800",905,815);  text("900,900",905,915);  } |

Notes (Points of interest, mistakes, lessons learned, web resources, and thoughts):

|  |
| --- |
|  |