Solution Review: Calculate Distance Between Two Points

This lesson discusses the detailed solution review to the problem in the previous lesson.

We'll cover the following

Solution:

```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
        x: i32,
        y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.y,2);
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}", test(point1, point2));
}
```

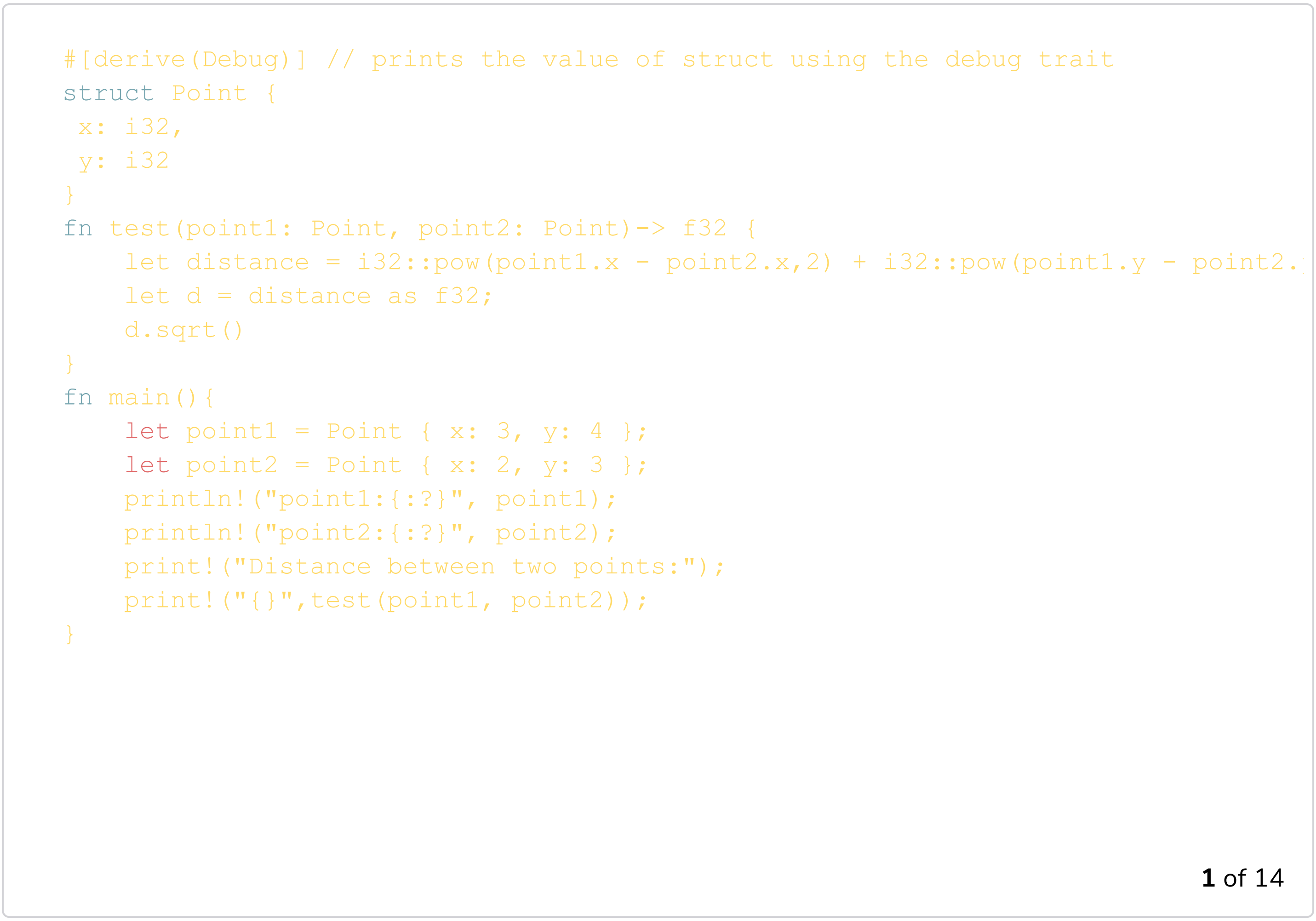
Explanation

- `struct` `Point`
    - On **line 2**, a `struct` `Point` is defined which has two items `x` of type `i32` and `y` of type `i32`.
- `test` **function**

- On **line 6**, a function `test` is defined which takes parameter `point1` and `point2` of type `Point` and returns an `f32` type, i.e., the distance between the two points.
- On **line 7**, $(x1-x2)^2+(y1-y2)^2$, is calculated and stored in variable `distance`.
- On **line 8**, `distance` is converted as `f32` and the result is stored in variable `d`.
- On **line 9**, `d.sqrt()` takes the square root of distance and returns the result.

- The value of instance is printed on the console.

The following illustration explains the above code:



```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3

4

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3   2

4   3

```
Output : point1:Point { x: 3, y: 4 }
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3   2

4   3

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

```
3        2

4        3
```

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

```
3        2

4        3
```

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:
```

```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

Output :

```
point1:Point { x: 3, y: 4 }
point2:Point { x: 2, y: 3 }
Distance between two points:
```

```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

Output :

```
point1:Point { x: 3, y: 4 }
point2:Point { x: 2, y: 3 }
Distance between two points:
```

```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt() return 1.4142135
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:
```

```rust
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}
```

3 2

4 3

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:1.4142135
```

```
#[derive(Debug)] // prints the value of struct using the debug trait
struct Point {
 x: i32,
 y: i32
}
fn test(point1: Point, point2: Point)-> f32 {
    let distance = i32::pow(point1.x - point2.x,2) + i32::pow(point1.y - point2.
    let d = distance as f32;
    d.sqrt()
}
fn main(){
    let point1 = Point { x: 3, y: 4 };
    let point2 = Point { x: 2, y: 3 };
    println!("point1:{:?}", point1);
    println!("point2:{:?}", point2);
    print!("Distance between two points:");
    print!("{}",test(point1, point2));
}end of program code
```

```
Output : point1:Point { x: 3, y: 4 }
         point2:Point { x: 2, y: 3 }
         Distance between two points:1.4142135
```

14 of 14

Now you have learned about structs. What if you want a structure that only has ordered listing of all items? Let's learn about enumeration data types called, "enums" in the next chapter.