Understanding Computer Systems

Definitions!

Computer System – combination of all the components required to process and store data using a computer

Hardware – the equipment or physical devices associated with a computer

- Keyboards, mice, speakers, printers, monitors, CPU

Software – computer instructions that tell the hardware what to do

- Applications, operating system, device drivers, scripts

Programs – Instruction sets written by programmers for a specific task

Definitions!

Application Software – all the programs used on a computer to perform a user task

- Word processing, spreadsheets, payroll, inventory, games

System Software – programs used to manage the computer

- Operating systems (Windows, Linux, Google Android, Apple iOS)
- Device drivers (for printers and other hardware devices)

Three Major Operations

Input – data items are entered into the computer system and are placed in memory

- Keyboards, mice, touch screens – devices that perform input operations
- **Data Items** – all text, numbers and other data entered into and processed by a computer
- Names, products, images, sounds, mouse movements

- **Data vs. Information** - Data is used to describe items that are input and information is used to describe data that has been processed and output

Three Major Operations

Processing – involves organizing, sorting, checking and performing calculations on data items.

- **Central Processing Unit (CPU)** – hardware component that performs processing

Output - Presentation of the information that has resulted from processing the data item inputs.

- Data is sent to an output device (printer, monitor, etc.)

Programming Languages

Programming Language – used to write computer instructions

- Visual Basic, C#, C++, Java
- Scripting languages – Python, Perl, PHP

Syntax – language's rules, computers cannot interpret incorrect syntax at all.

- Incorrect English syntax – we can still understand what is being said

Programming Languages

- We write code in a Programming Language – **Source Code**

- Before the computer can execute the instructions, it must be translated to **Machine Language** – the ones and zeros that the computer can understand (also called **Binary Language** and **Object Code**)

Programming Languages

Two types of translators

- **Compiler** – translates the entire program first and then you Run or Execute the program (C++)
- **Interpreter** – translates each instruction and then executes it (JavaScript, Ruby)

Translators will find **syntax errors**, but not necessarily **logic errors**.

Computer Memory

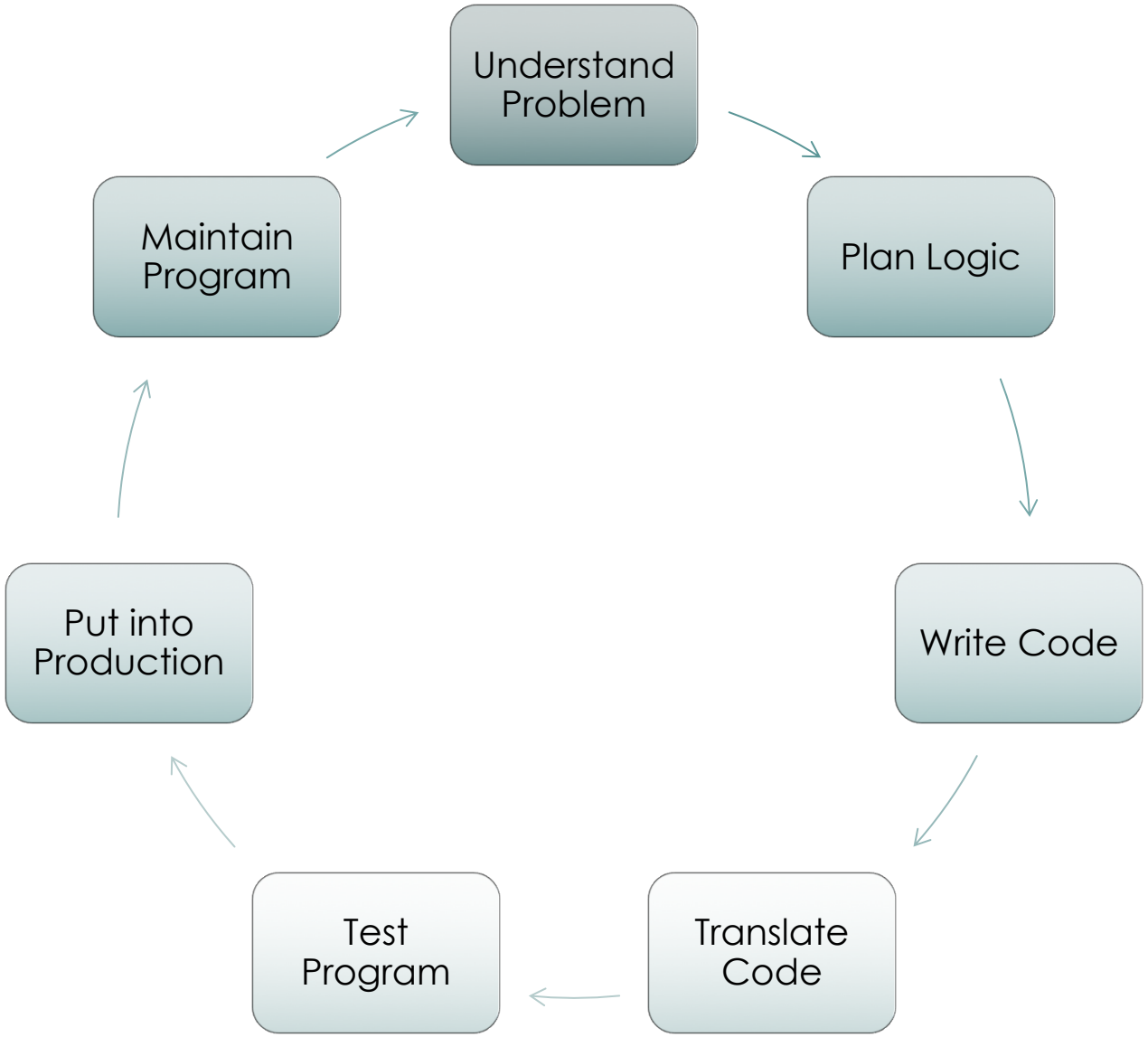
Volatile – temporary internal memory, where a program's instructions are stored when it is running and where the data items are manipulated.

- If the computer is turned off, this data is lost
- Known as Random Access Memory (RAM)

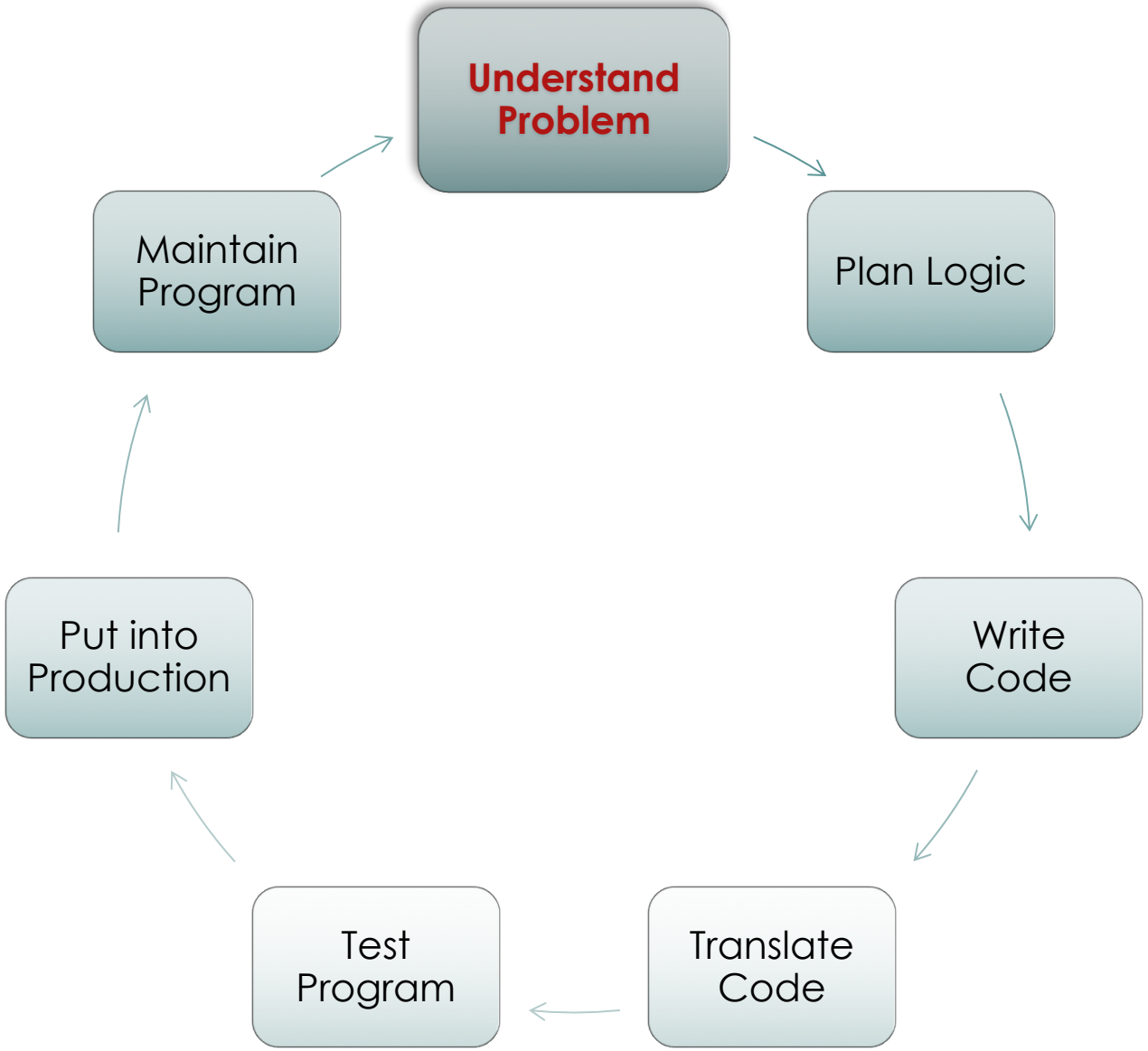
Non-volatile – permanent storage, such as a disk, where contents are retained even if power is lost. The translated program and the source code are stored to disk so that they can be executed again. Data items can also be store to disk.

Programming Development Cycle

Programming Development Cycle



Programming Development Cycle



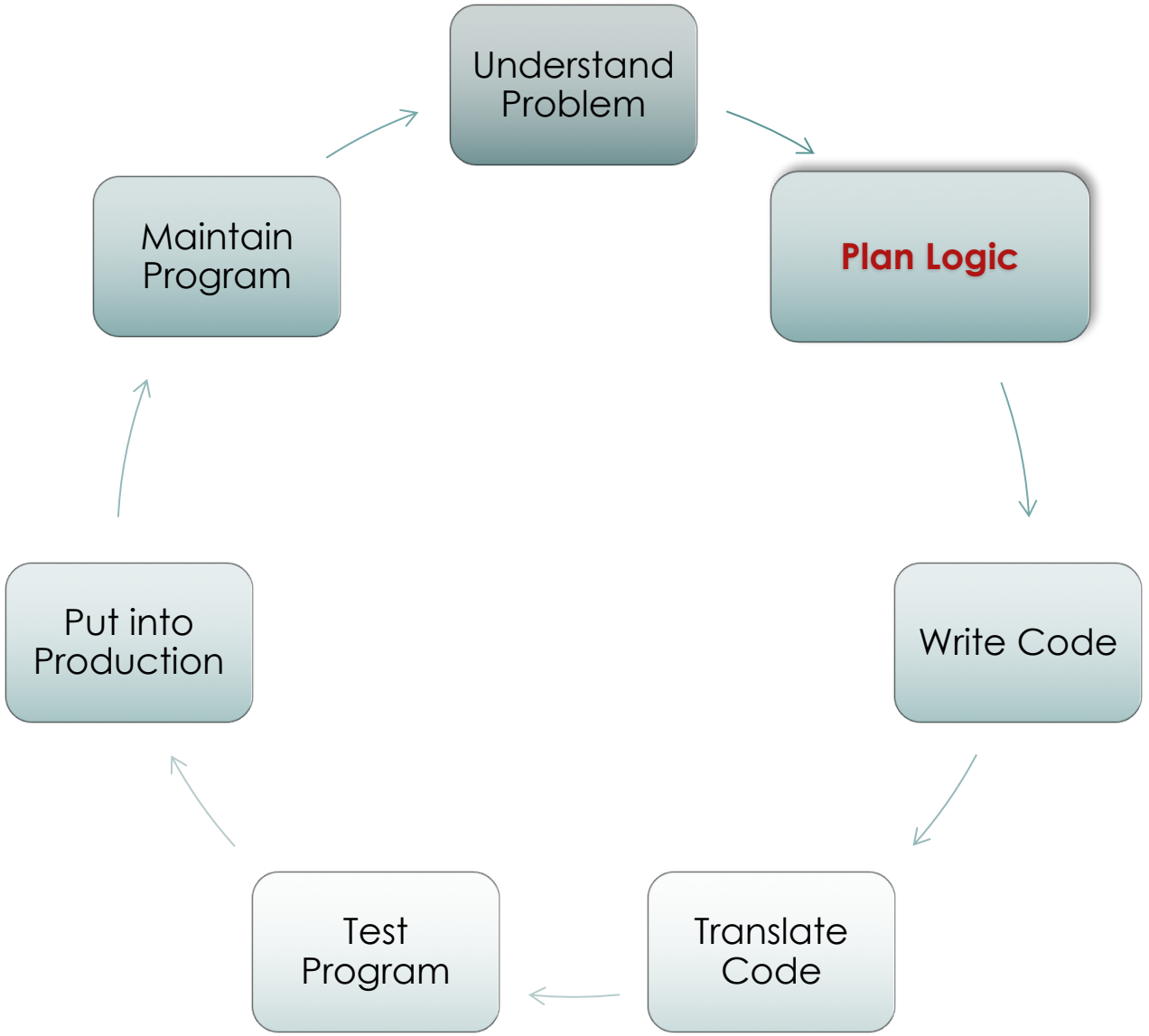
Understanding the Problem

Programs are written to satisfy the needs of the Users

Sometimes difficult to determine

- Vague requirements
- Incomplete requirements
- Needs that the Users are not yet aware of
- Wide variety / level of user skills
- Needs change

Programming Development Cycle



Plan the Logic

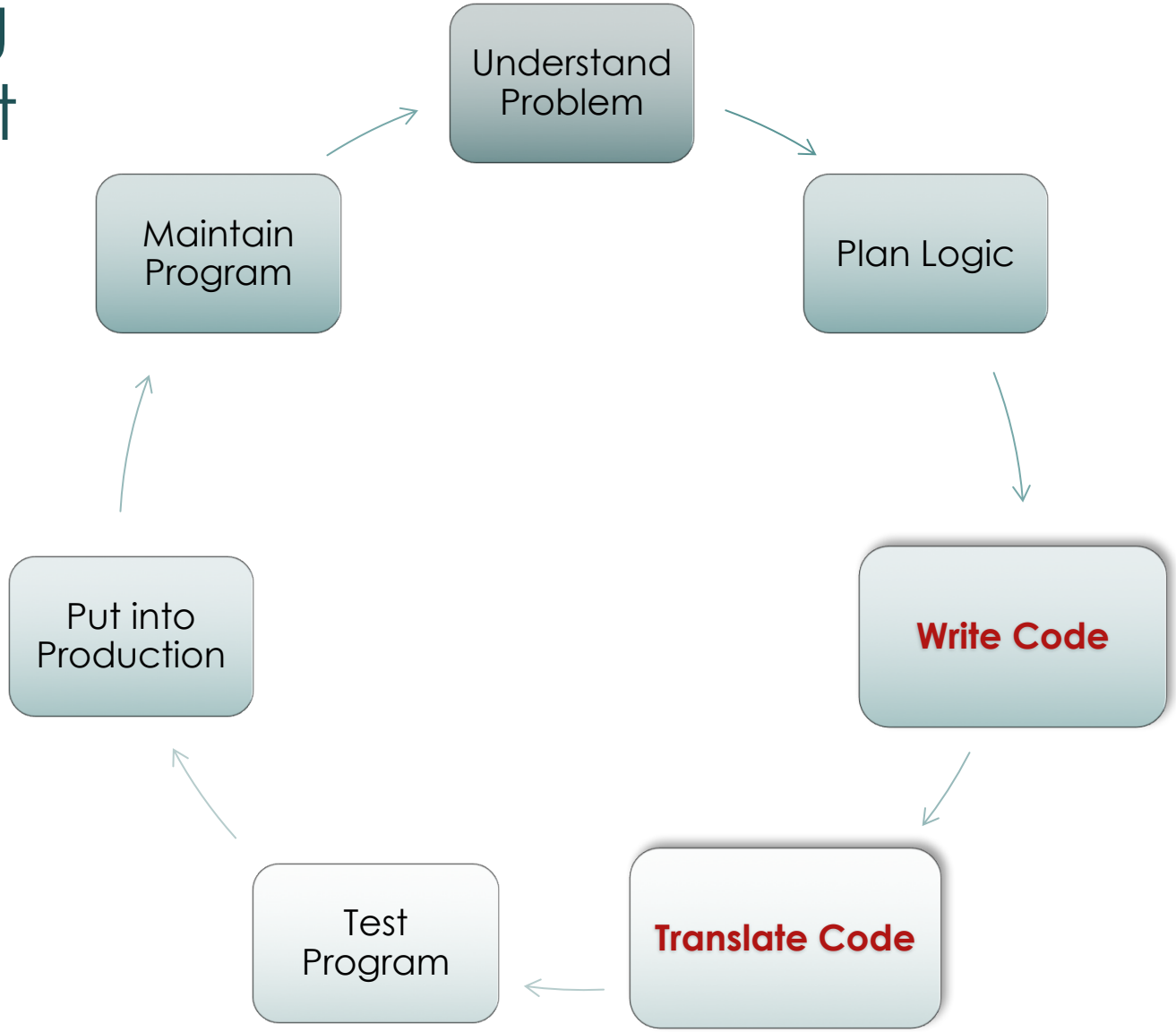
- Programmer develops the steps of program
- What steps to include, what order
- Many different ways to do the same thing
- **Algorithm** – Sequence of Steps or Rules you follow to solve a problem
- Independent of programming language

Plan the Logic

Use of planning tools

- Flowcharts – graphical representation of the steps
- Pseudo code – English code-like steps
- IPO chart – inputs, processing and outputs
- TOE charts – object-oriented, tasks, objects and events
- UML – Unified Modeling Language
- Storyboards – used to identify “use cases” for the program

Programming Development Cycle



Write the Code
Translate the Code

Choose a **high-level programming language**

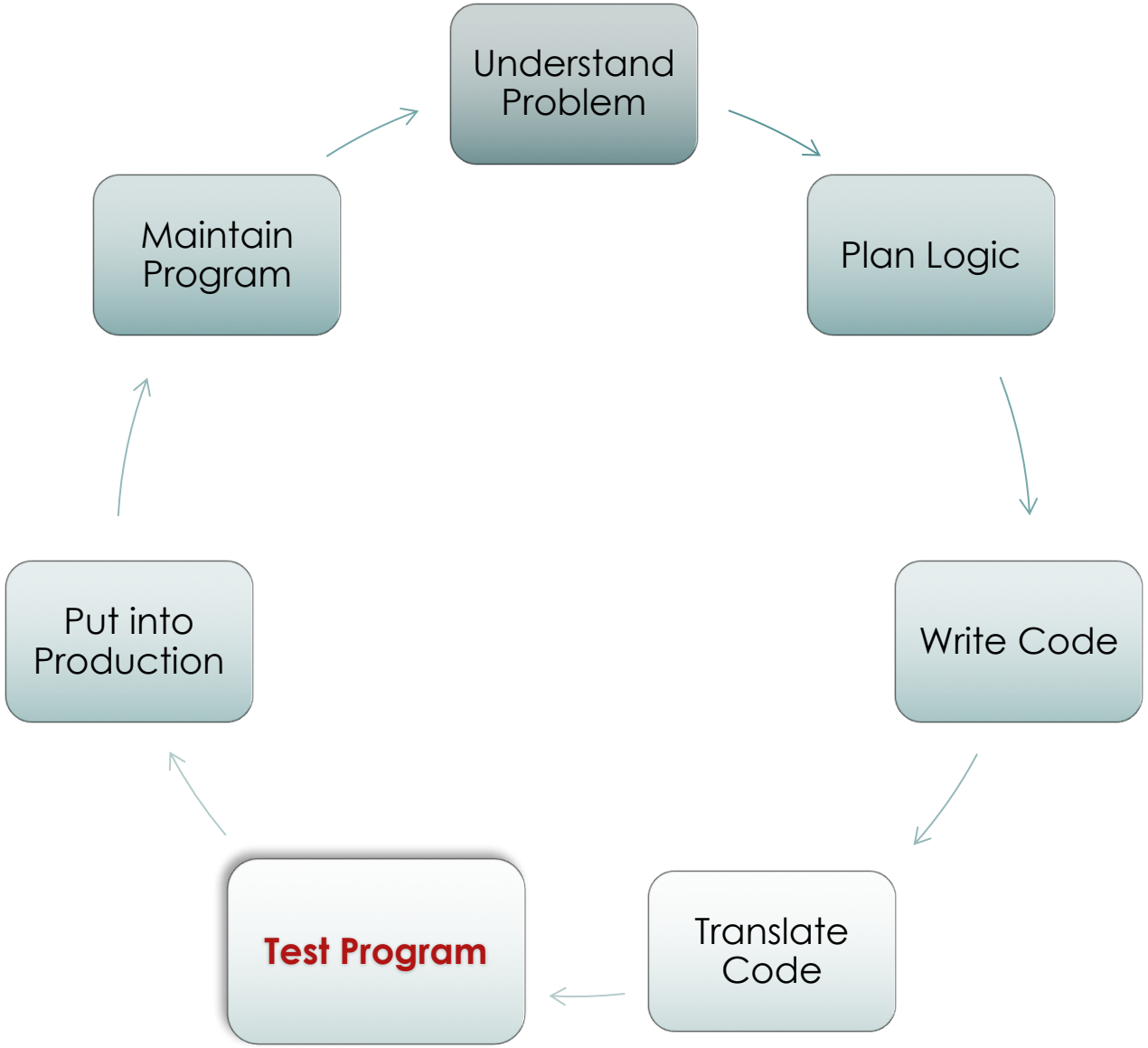
- Write the code using the syntax for that language

Use a compiler or interpreter to

- translate the program from the **high-level programming language** to the **low-level machine language** that the computer understands
- The compiler/interpreter will identify **syntax errors**

Once syntax errors are corrected, the executable program is ready to test.

Programming Development Cycle



Test the Program

Identifies **Logic Errors**

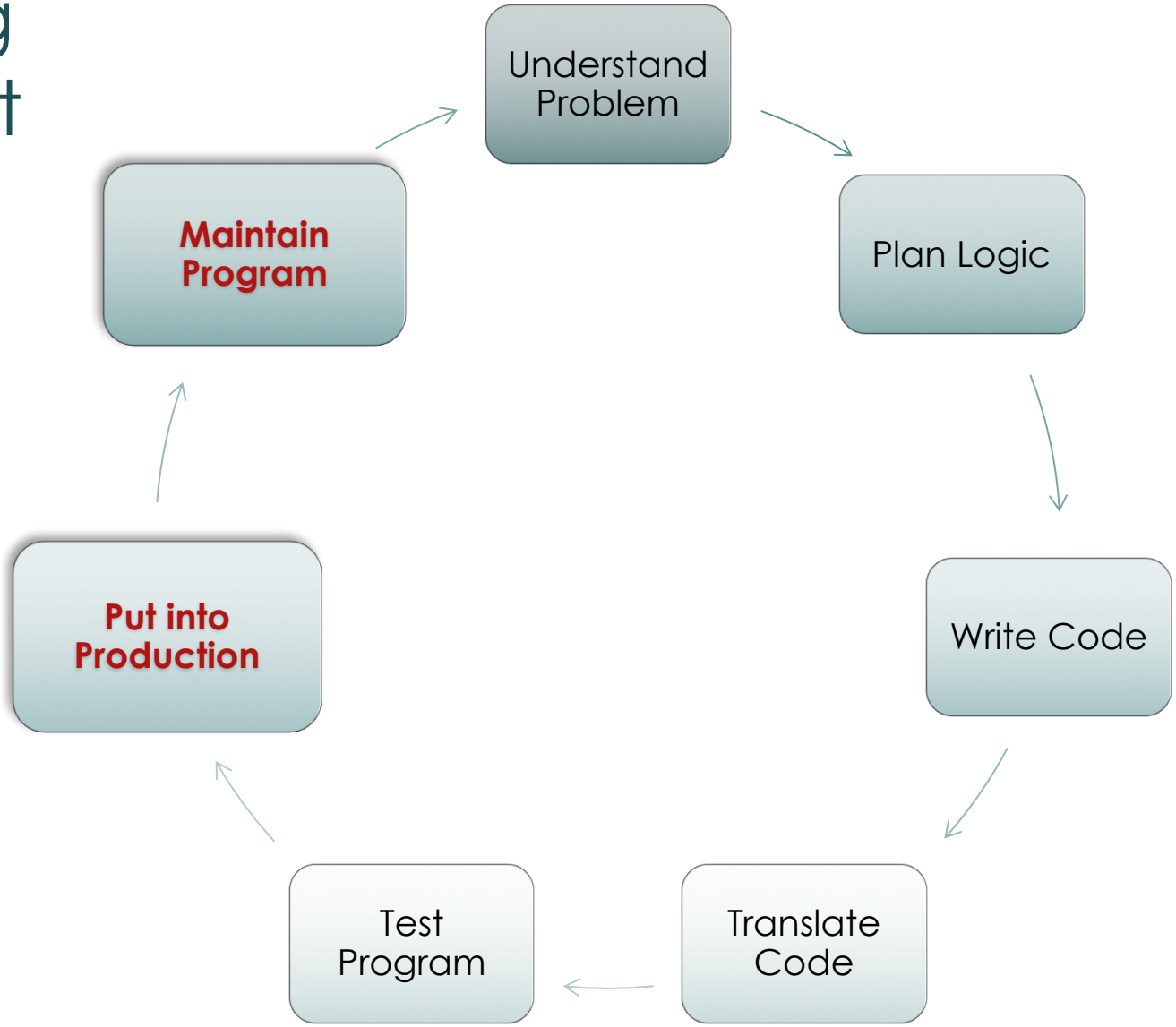
- Errors that result when you use a syntactically correct statement but in the wrong context

Test Data inputs should include representative data and cover all possible expected error conditions

- Examples: if a data value can be from 1-100, you should test 1, 100 (boundary conditions), a negative number or zero and a number greater than 100 (error conditions), and several numbers in the middle of the range

This testing process is called **Debugging**.

Programming Development Cycle



Put Program Into Production
Maintain the Program

Program is made available to Users

- May involve training of users
- Conversion – the entire set of actions an organization must take to switch over to using a new program

Maintenance – making necessary changes

- Due to new requirements, changes requested or required by the users, new data formats, etc.
- Repeat the development cycle

Problem Example

You have a kitchen robot (Rosie) that you are to program to make a peanut butter and jelly sandwich.

Understanding the problem:

- Describe the peanut butter and jelly sandwich
  - Type of bread – white, wheat, other
  - Type of peanut butter – creamy, chunky, all natural
  - Type of jelly – grape, strawberry, etc.
- What does it look like when it is done?
- What processes does the robot need to perform?
- Anything else?

Problem Example

Planning the Logic – Determine the following:

- Inputs
- Outputs
- Processing

Program Logic

Programs with syntax errors cannot be translated

Programs with no syntax errors can still contain **logic** errors

Programs are basically instructions to the computer

- Instructions must be in the correct order
- Instructions must be complete – you cannot leave any out
- You must not add extra instructions

Program Logic

Peanut Butter and Jelly Sandwich

- spread peanut butter on bread
- get knife
- spread jelly on bread
- get bread
- add anchovies
- get peanut butter and jelly

Program Logic

Syntax is correct English – however…

- Instructions are out of sequence
- There is an extra instruction (add anchovies)
- There are missing instructions

Program Logic

Simple computer programs include steps that perform input, processing and output.

Write a program to compute the square of a number – Three instructions – one input, one processing and one output

```
input myNumber
set myNumberSquared = myNumber * myNumber
output myNumberSquared
```

myNumber and **myNumberSquared** are examples of **variables** – a named memory location whose value can vary