# Making Decisions in Turing

For our programs to become more useful, we need to start making decisions.

Suppose we want to decide, “Is the user old enough to vote?”

## Conditions

Conditions are statements that can be tested as either TRUE or FALSE.

```
input age
if (age is 18 or over) then “You can vote!”
if (age is under 18) then “You cannot vote.”
```

This is an example of pseudocode, where we use plain language, but it looks a bit like computer code.

## Example – Voting Age
## 1. design using comments

As programs get more complicated, start by using comments to build a framework to solve the problem:

```
% ask the user's age

% if they are 18 or older, they can vote

% if they are under 18, they cannot vote
```

# Example – Voting Age
## 2. Add easy code

```
var age : int      % declare a variable for age

% ask the user's age
put “How old are you? “..
get age

% if they are 18 or older, they can vote
put “You can vote!”
% if they are under 18, they cannot vote
put “You are not old enough to vote.”
```

# Example – Voting Age 3. Add new code

```
var age : int        % declare a variable for age

% ask the user's age
put "How old are you? " ..
get age

% if they are 18 or older, they can vote
if (age >= 18) then
    put "You can vote!"
end if

% if they are under 18, they cannot vote
if (age < 18) then
    put "You are not old enough to vote."
end if
```

# Example – Voting Age
## 3b. Add new code – another option

```
var age : int      % declare a variable for age

% ask the user's age
put "How old are you? " ..
get age

% if they are 18 or older, they can vote
if (age >= 18) then
    put “You can vote!”
else % if they are under 18, they cannot vote
    put “You are not old enough to vote.”
end if
```

## Comparison Operators

Making a decision using selection requires a comparison between <u>two</u> quantities or values. Each comparison will use one of the comparison operators listed below.

| | | | |
|---|---|---|---|
| < | less than | <= | less than or equal to |
| > | greater than | >= | greater than or equal to |
| = | equal to | not= | not equal to |

## Flowcharts

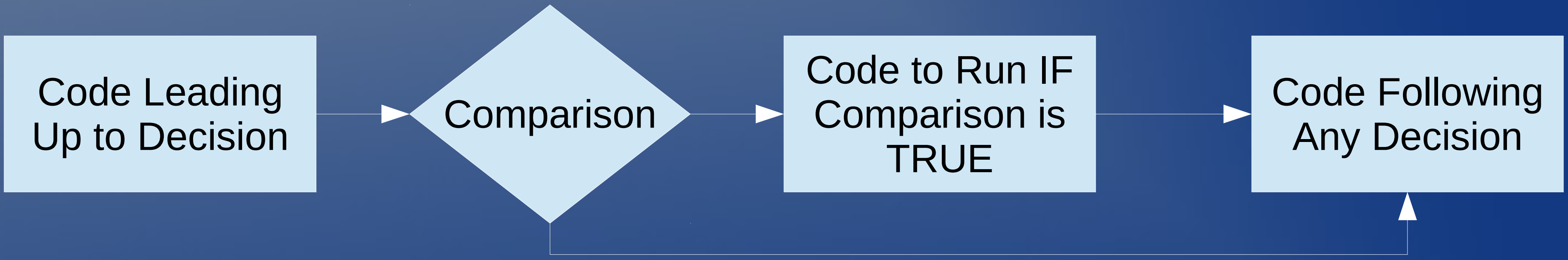
So far, we have discussed sequential programming, where the instructions are executed one after the other.

In order to design and write programs sequentially, we need to ensure that our sequence, or flow, is correct.

It is often useful to represent our program designs using a diagram, or flowchart.

# If-Then (one choice)

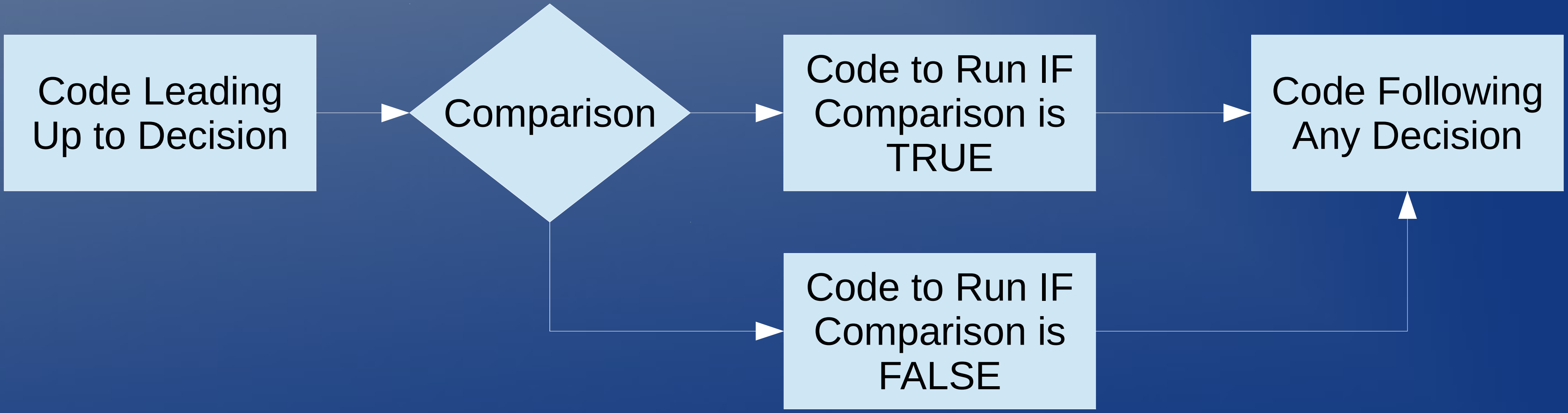
```
if (comparison) then
    statements if comparison is true
end if
```


Code Leading Up to Decision
Comparison
Code to Run IF Comparison is TRUE
Code Following Any Decision

Nothing Happens IF Comparison FALSE

# If-Then-Else (two choices)

```
if (comparison) then
    statements if comparison is true
else
    statements if comparison is false
end if
```


Code Leading Up to Decision
Comparison
Code to Run IF Comparison is TRUE
Code Following Any Decision
Code to Run IF Comparison is FALSE