Managing Program Flow

INTRODUCTION

Simon Allardice
STAFF AUTHOR, PLURALSIGHT
@allardice www.pluralsight.com

“Begin at the beginning,” the King said, very gravely, “and go on till you come to the end: then stop.”

Lewis Carroll, *Alice in Wonderland*

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

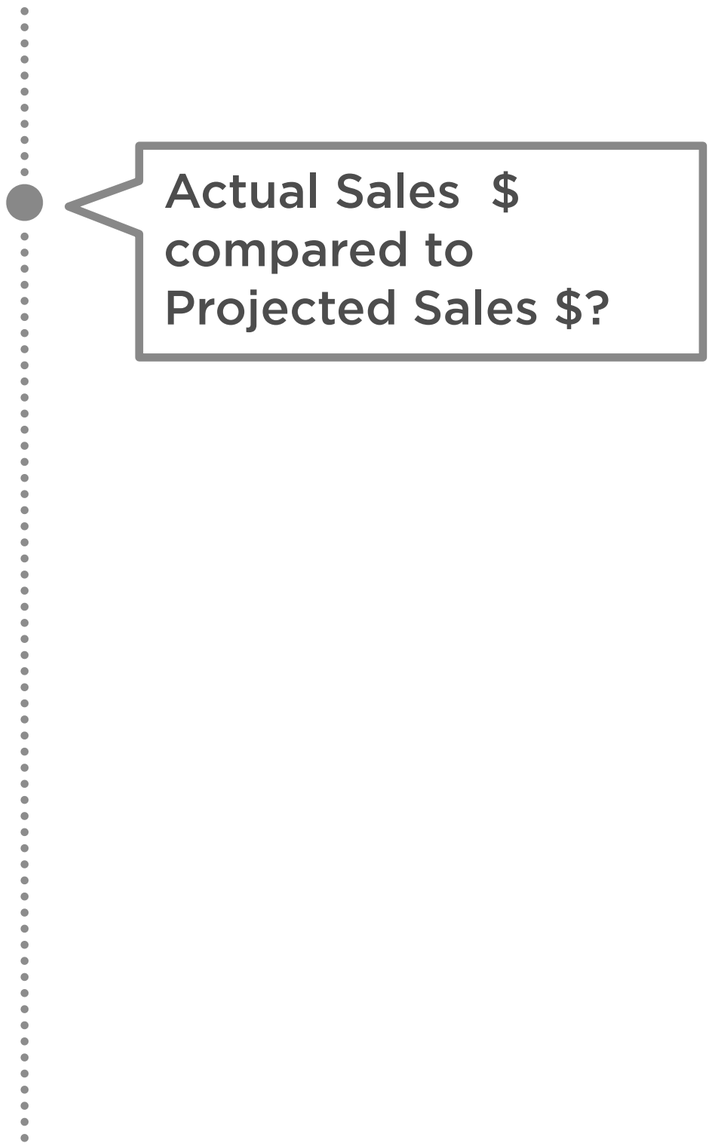
Statements Execute in Order

```
statement one
statement two

statement three might be long and be split across
   multiple lines to make it
   easier to read

statement four
statement five
```

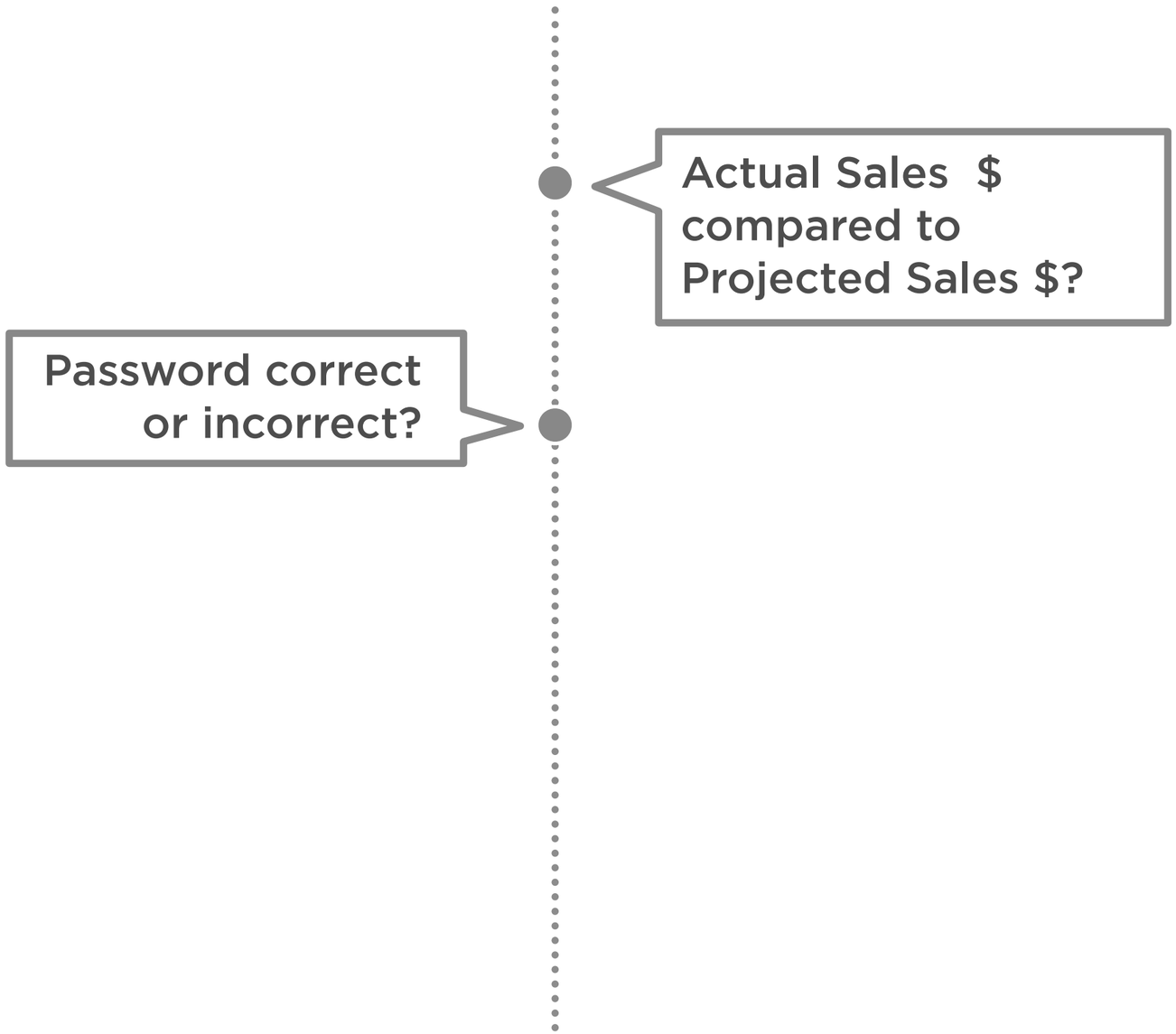
Controlling the Flow of Your Program

Controlling the Flow of Your Program

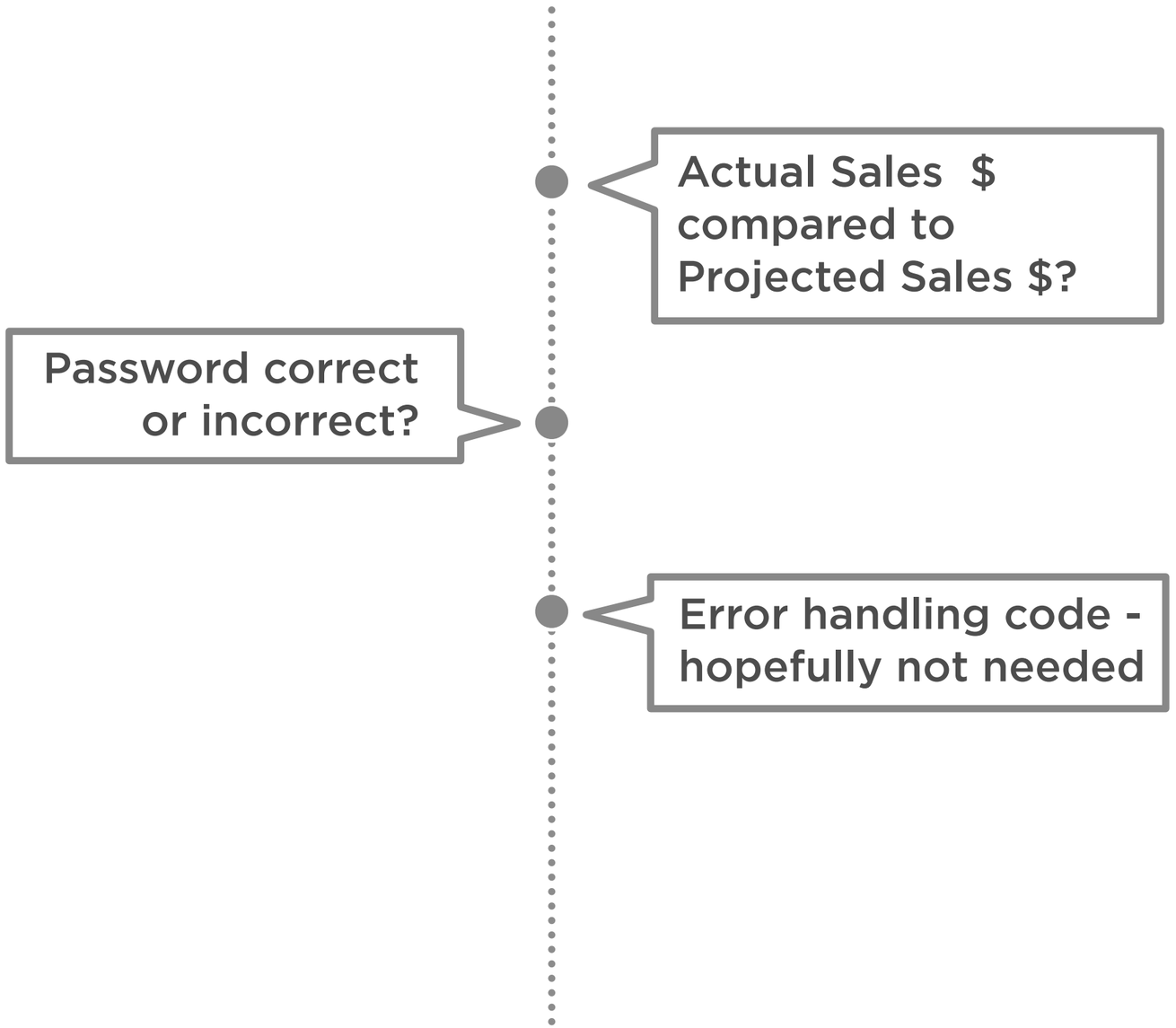
Controlling the Flow of Your Program



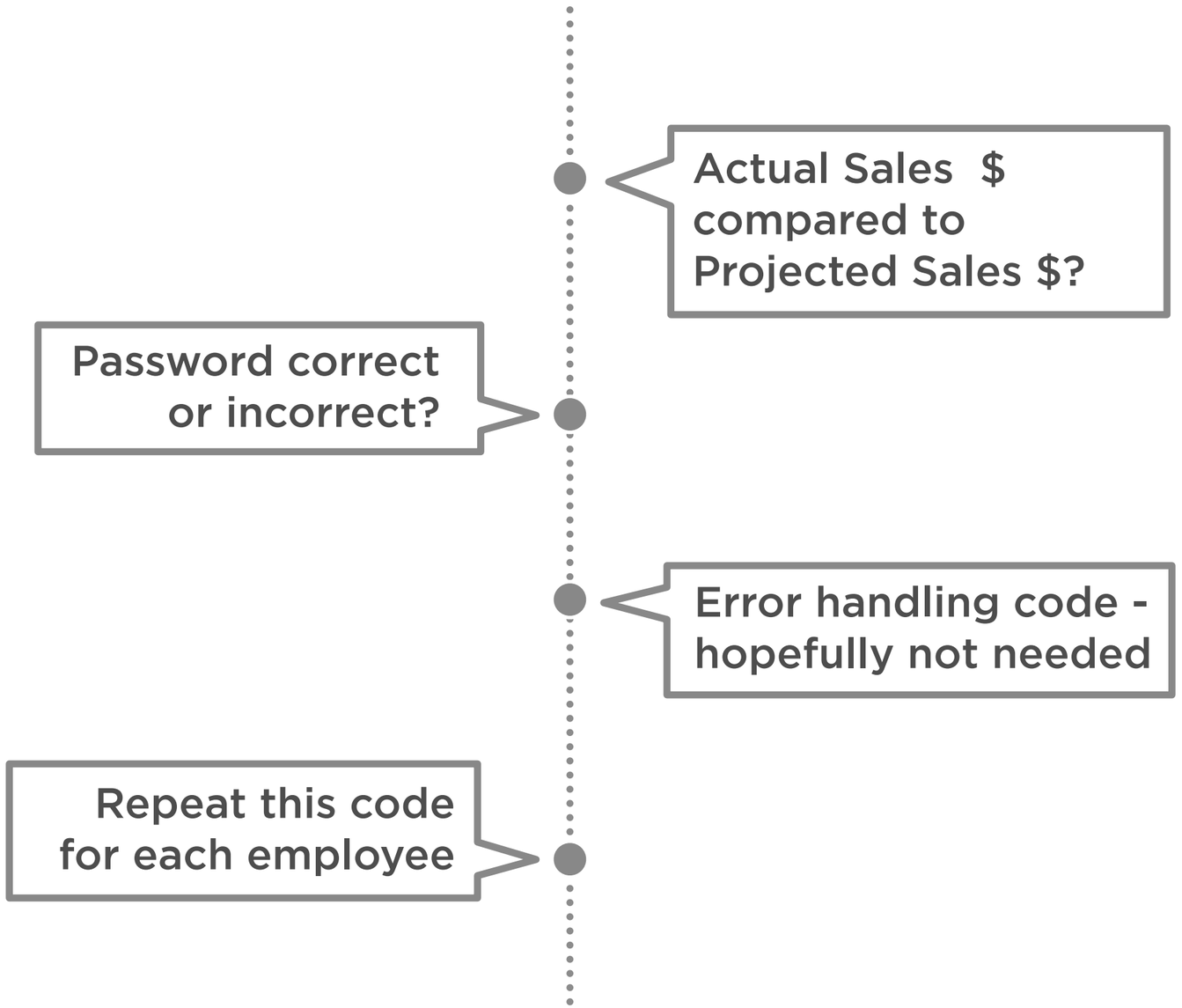
Controlling the Flow of Your Program



Controlling the Flow of Your Program



Controlling the Flow of Your Program



Program Flow: Conditions

if

Example: **if**

```
if balance > 1000  {
   print("You earn interest on this account.")
   // any additional statements
}
```

Example: **if**

```
if balance > 1000  {
    print("You earn interest on this account.")
    // any additional statements
}
```

Example: **if**

condition

```
if balance > 1000 {
    print("You earn interest on this account.")
    // any additional statements
}
```

Example: **if**

condition - any result must be either true or false

```
if balance > 1000 {
    print("You earn interest on this account.")
    // any additional statements
}
```

Conditions

```
if balance > 1000        {...}

if balance < 0           {...}

if temperature <= 55     {...}

if creditsRemaining >= 0 {...}
```

Checking Equality in a Condition

Checking Equality in a Condition

```
if balance = 0

if petName = "Jock"
```

Checking Equality in a Condition

```
if balance = 0

if petName = "Jock"
```

Incorrect:
Single = is for assignment (to set a value)

Checking Equality in a Condition

```
if balance = 0

if petName = "Jock"
```

Incorrect:
Single = is for assignment (to set a value)

```
if balance == 0

if petName == "Jock"
```

Checking Equality in a Condition

```
if balance = 0
if petName = "Jock"
```

Incorrect:
Single = is for assignment
(to set a value)

```
if balance == 0
if petName == "Jock"
```

Correct:
Double == is for equality
(to check a value)

Checking Equality in a Condition

```
if balance = 0

if petName = "Jock"
```

Incorrect:
Single = is for assignment
(to set a value)

```
if balance == 0

if petName == "Jock"
```

Correct:
Double == is for equality
(to check a value)

```
if petName != "Fido"  // check for inequality
```

Code Blocks

```
if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}
```

Code Blocks

```
if balance > 1000 {
    print("You earn interest on this account.")
    // additional statements
}
```

start of the code block

Code Blocks

```
if balance > 1000 {
    print("You earn interest on this account.")
    // additional statements
}
```

start of the code block

end of the code block

Symbol Check

Symbol Check

Curly Braces

{ }

Symbol Check

Curly Braces

{ }

Square Brackets

[ ]

Symbol Check

Curly Braces	Square Brackets	Parentheses
{ }	[ ]	( )

```
struct file_handler {
 const char *type;
 int (*handler)(const char *line);
};

static void push_string(const char *name)
{
 unsigned int name_len = strlen(name) + 1;

 fputs(name, stdout);
 putchar(0);
 offset += name_len;
}

static void push_pad (void)
{
 while (offset & 3) {
  putchar(0);
  offset++;
 }
}
```

```
struct file_handler {
 const char *type;
 int (*handler)(const char *line);
};

static void push_string(const char *name)
{
 unsigned int name_len = strlen(name) + 1;

 fputs(name, stdout);
 putchar(0);
 offset += name_len;
}

static void push_pad (void)
{
 while (offset & 3) {
  putchar(0);
  offset++;
 }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

```
struct file_handler {
  const char *type;
  int (*handler)(const char *line);
};

static void push_string(const char *name)
{
  unsigned int name_len = strlen(name) + 1;

  fputs(name, stdout);
  putchar(0);
  offset += name_len;
}

static void push_pad (void)
{
  while (offset & 3) {
    putchar(0);
    offset++;
  }
}
```

Code Blocks

```
// more code above

if balance > 1000 {
    print("You earn interest on this account.")
    // additional statements
}

// more code below
```

start of code block

end of code block

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}

// more code below
```

Code Blocks

```
// more code above

if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

Code Blocks: Brace Style

Code Blocks: Brace Style

Opening brace on same line

```
if someVariable > 0  {
   // Any additional statements
}
```

Code Blocks: Brace Style

Opening brace on same line

```
if someVariable > 0  {
   // Any additional statements
}
```

Opening brace on next line

```
if someVariable > 0
{
   // Any additional statements
}
```

Code Blocks: Brace Style

Opening brace on same line

```
if someVariable > 0  {
    // Any additional statements
}
```

Opening brace on next line

```
if someVariable > 0
{
    // Any additional statements
}
```

Code Blocks: Alternative

Code Blocks: Alternative

BASIC Style

```
If Balance > 1000 Then
   Print("You earn interest on this account.")
   ' Any additional statements
End If
```

Code Blocks: Alternative

BASIC Style

```
If Balance > 1000 Then
   Print("You earn interest on this account.")
   ' Any additional statements
End If
```

C Style

```
if balance > 1000  {
   print("You earn interest on this account.")
   // Any additional statements
}
```

Code Blocks: Alternative Styles

Code Blocks: Alternative Styles

Python Style

```
if balance > 1000:
     print('You earn interest on this account.')
     #any additional statements



print('Now outside the code block.')
```

Code Blocks: Alternative Styles

Python Style

```
if balance > 1000:
    print('You earn interest on this account.')
    #any additional statements

print('Now outside the code block.')
```

Code Blocks

```
if balance > 1000  {
   print("You earn interest on this account.")
   // additional statements
}


// more code below
```

If / Else

```
if balance > 1000  {
   // this block only if condition is true
   print("You earn interest on this account.")
}
```

If / Else

```
if balance > 1000  {
   // this block only if condition is true
   print("You earn interest on this account.")
} else  {
   // this block only if condition is false.
   print("You do NOT earn interest on this account.")
}
```

If / Else

```
if balance > 1000 {
    // this block only if condition is true
    print("You earn interest on this account.")
} else  {
    // this block only if condition is false.
    print("You do NOT earn interest on this account.")
}
```

If / Else

```
if balance > 1000 {
    // this block only if condition is true
    print("You earn interest on this account.")
} else {
    // this block only if condition is false.
    print("You do NOT earn interest on this account.")
}
```

If / Else

```
if balance > 1000 {
    // this block only if condition is true
    print("You earn interest on this account.")
} else {
    // this block only if condition is false.
    print("You do NOT earn interest on this account.")
}

// now outside the if/else
print("This statement is ALWAYS reached.")
```

Program Flow: Creating Complex Conditions

Complex Conditions

Complex Conditions

```
if
{
    // code to calculate interest
    // any additional statements
}
```

Complex Conditions

```
if  balance > 1000
{
    // code to calculate interest
    // any additional statements
}
```

Complex Conditions

```
if  balance > 1000  AND
{
    // code to calculate interest
    // any additional statements
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}


if
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}


if cartTotal > 100 OR
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions

```
if balance > 1000 AND accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

```
if cartTotal > 100 OR memberType == "Premium"
{
    print("You qualify for free shipping!")
    shippingCost = 0
}
```

Complex Conditions: Common Syntax

```
if    balance > 1000   AND  accountType == "Savings"
{
    // code to calculate interest
    // any additional statements
}
```

&& (and) **|| (or)**

Complex Conditions: Common Syntax

```
if  ( balance > 1000 ) AND (accountType == "Savings")
{
    // code to calculate interest
    // any additional statements
}
```

&& (and) **|| (or)**

Complex Conditions: Common Syntax

```
if ((balance > 1000 ) AND (accountType == "Savings"))
{
    // code to calculate interest
    // any additional statements
}
```

&& (and) **||** (or)

Complex Conditions: Common Syntax

```
if ((balance > 1000 ) && (accountType == "Savings"))
{
    // code to calculate interest
    // any additional statements
}
```

&& (and) **||** (or)

Complex Conditions: Common Syntax

```
if ((balance > 1000 ) && (accountType == "Savings"))
{
    // code to calculate interest
    // any additional statements
}
```

&& (and) **|| (or)**

```
if (condition1 && condition2 && condition3) {
...
}
```

Multiple Branches

Multiple Branches

```
If supermarket has full-fat milk
   buy full fat milk
Else
   buy skimmed milk
End
```

Multiple Branches

```
If supermarket has full-fat milk
   buy full fat milk
Else
   buy skimmed milk




End
```

Multiple Branches

```
If supermarket has full-fat milk
   buy full fat milk
Else
   If supermarket has 2% milk
      buy 2% milk
   Else
      buy skimmed milk
   End
End
```

Multiple Branches

```
If supermarket has full-fat milk
    If it's on sale AND expiration date > 2 weeks ahead
        buy two full fat milk
    Else
        buy one full fat milk
    End
Else
    If they have 2% milk
        buy 2% milk
    Else
        buy skimmed milk
    End
End
```

Multiple Branches

```
If supermarket has full-fat milk
    If it's on sale AND expiration date > 2 weeks ahead
        buy two full fat milk
    Else
        buy one full fat milk
    End
Else
    If they have 2% milk
        buy 2% milk
    Else
        buy skimmed milk
    End
End
```

Multiple Branches

```
If supermarket has full-fat milk
    If it's on sale AND expiration date > 2 weeks ahead
        buy two full fat milk
    Else
        buy one full fat milk
    End
Else
    If they have 2% milk
        buy 2% milk
    Else
        buy skimmed milk
    End
End
```

Multiple Branches

```
If supermarket has full-fat milk
    If it's on sale AND expiration date > 2 weeks ahead
        buy two full fat milk
    Else
        buy one full fat milk
    End
Else
    If they have 2% milk
        buy 2% milk
    Else
        buy skimmed milk
    End
End
```

Multiple Branches

```
If supermarket has full-fat milk
    If it's on sale AND expiration date > 2 weeks ahead
        buy two full fat milk
    Else
        buy one full fat milk
    End
Else
    If they have 2% milk
        buy 2% milk
    Else
        buy skimmed milk
    End
End
```

nested if

nested if

Dealing With Ranges

```
if bathTemperature < 90 {
    print("Brrr!")
} else {
    if bathTemperature < 95 {
        print("That's lukewarm.")
    } else {
        if bathTemperature < 105 {
            print("Perfect!")
        } else {
            if bathTemperature < 110 {
                print("This isn't a hot tub")
            } else {
                print("Are you trying to boil a lobster?)
            }
        }
    }
}
```

Dealing With Ranges

```
if bathTemperature < 90 {
    print("Brrr!")
}
if bathTemperature > 90 && bathTemperature < 95 {
    print("That's lukewarm.")
}
if bathTemperature >= 95 && bathTemperature <= 105 {
    print("Perfect!")
}
if bathTemperature >= 105 && bathTemperature <= 110 {
    print("This isn't a hot tub")
}
if bathTemperature >= 110 {
     print("Are you trying to boil a lobster?)
}
```

Switch Statement

```
switch bathTemperature {
    case < 90:
       print("Brrr!")
    case 90...95:
       print("That's lukewarm.")
    case 96...105:
       print("Perfect!")
    case 106...110 {
       print("This isn't a hot tub"
    default:   // any other value
       print("Are you trying to boil a lobster?)
}
```

Switch Statement

```
switch bathTemperature {
    case < 90:
       print("Brrr!")
    case 90,91,92,93,94,95:
       print("That's lukewarm.")
    case 96,97,98,99,100,101,102,103,104,105:
       print("Perfect!")
    case 106,107,108,109,110 {
       print("This isn't a hot tub"
    default:   // any other value
       print("Are you trying to boil a lobster?)
}
```

Program Flow: Creating Loops

Loops / Iteration

Loops / Iteration

```
statement one
statement two
statement three
// etc.
```

Loops / Iteration

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this code"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this 10 times"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this 100000 times"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this for each employee in the company"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this for each document in the folder"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this for each track in the album"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

Loops / Iteration

"repeat this forever"

```
{
  statement one
  statement two
  statement three
  // etc.
}
```

While Loop

While Loop

```
statement one
statement two
statement three
// etc.
```

While Loop

```
while  some condition is true  {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (some condition is true) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

true / false

```
while (some condition is true) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (some condition is true) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

true

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

true

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

true

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

false

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

```
while (   myVariable < 1000   ) {
    statement one
    statement two
    statement three
    // etc.
}
```

While Loop

While Loop

```
int counter = 0
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

counter output

```
The counter is: 0
The counter is: 1
The counter is: 2
The counter is: 3
The counter is: 4
The counter is: 5
The counter is: 6
The counter is: 7
The counter is: 8
The counter is: 9
Now outside the loop.
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

While Loop

```
int counter = 0

while ( counter < 10 )  {
    print("The counter is: " + counter)
    counter = counter + 1
}

print("Now outside the loop.")
```

For Loop

For Loop

```
for (          )  {

}
```

For Loop

```
// C-style for loop
for (                                                    ) {

}
```

For Loop

```
// C-style for loop
for ( initializer    ;   condition  ; at-the-end   )  {

}
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++  )  {

}
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++  )  {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++  )  {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++  )  {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++ )  {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++ ) {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

counter output

```
The counter is: 0
The counter is: 1
The counter is: 2
The counter is: 3
The counter is: 4
The counter is: 5
The counter is: 6
The counter is: 7
The counter is: 8
The counter is: 9
Now outside the loop.
```

For Loop

```
// C-style for loop
for ( int counter = 0 ; counter < 10 ; counter++ )  {
    print("The counter is: " + counter)
}

print("Now outside the loop.")
```

For-In / For-Each Loop

For-In / For-Each Loop

```
for each item in collection  {
    // statements to execute
    // ...
}
```

For-In / For-Each Loop

```
for each person in contactList  {
    // statements to execute
    // ...
}
```

For-In / For-Each Loop

```
for each file in documentsFolder  {
    // statements to execute
    // ...
}
```

For-In / For-Each Loop

```
for each track in album  {
    // statements to execute
    // ...
}
```