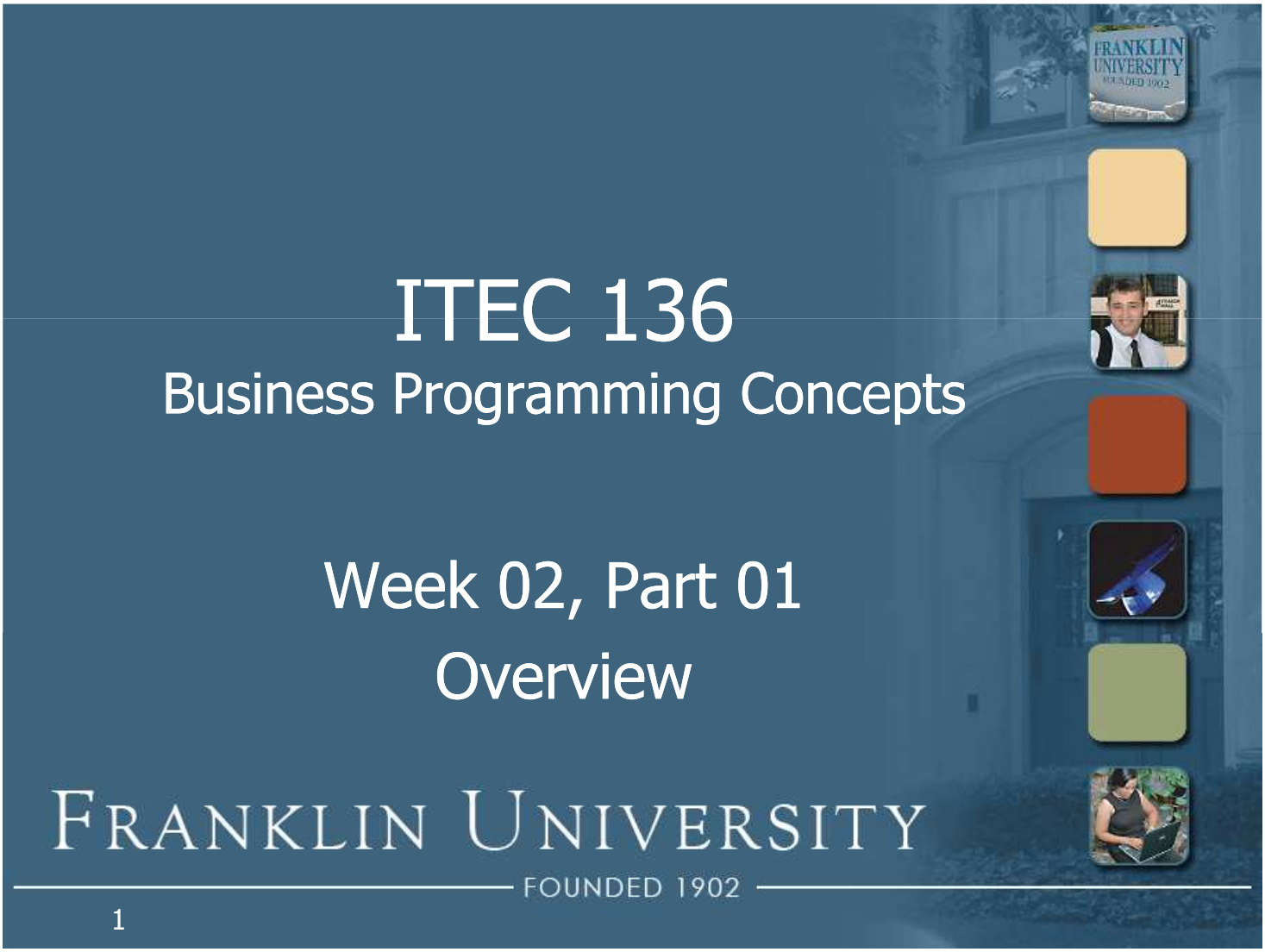


Week 2 Overview

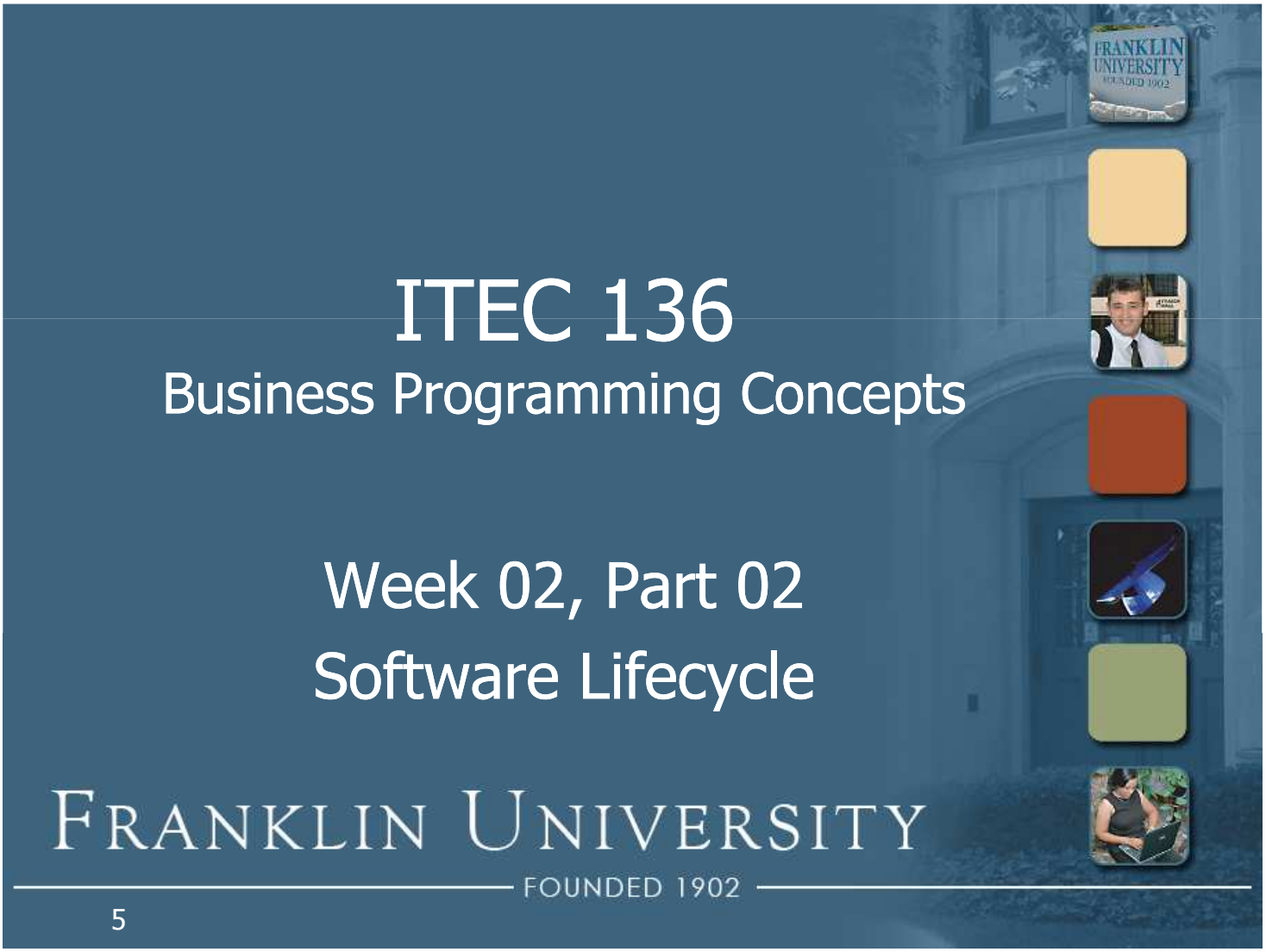
- Week 1 review
  - HTML documents
    - Document Type Definition
    - Elements (tags)
    - Attributes
    - Entities
  - Inline and external JavaScript

Week 2 Overview

- Outcomes
  - Describe the inputs, activities, and outputs of each step in the software development life cycle.
  - Describe arithmetic, relational, and logical operators in terms of their input and output data types.

Week 2 Overview

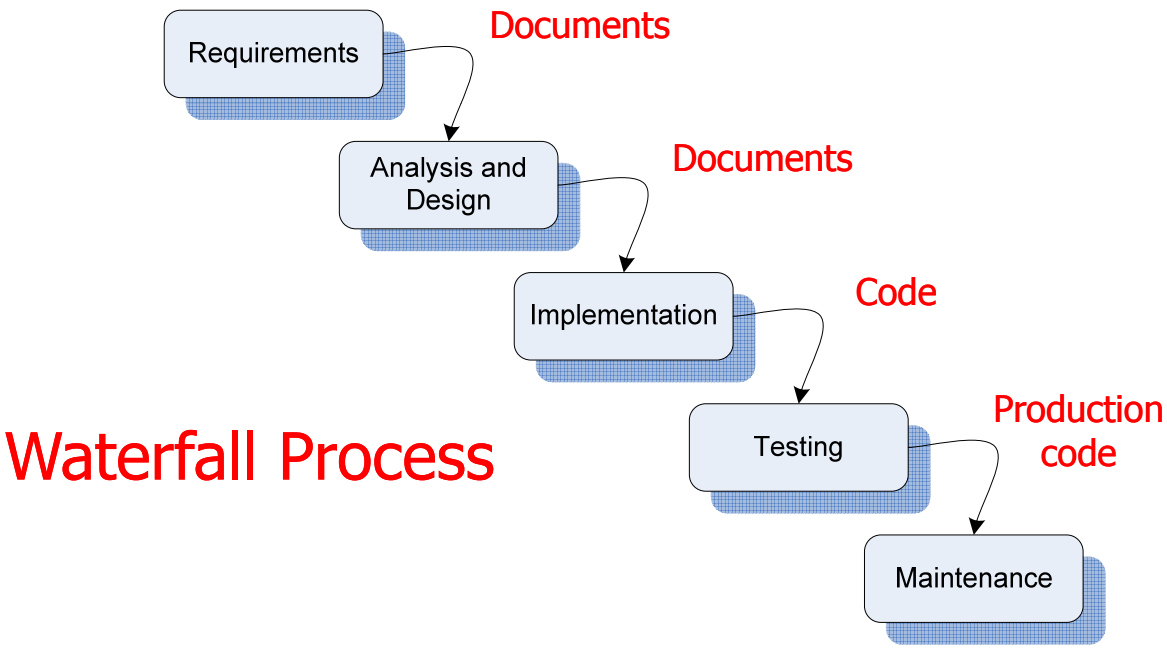
- Outcomes
  - Declare, define, and use variables in a script.



Software Lifecycle

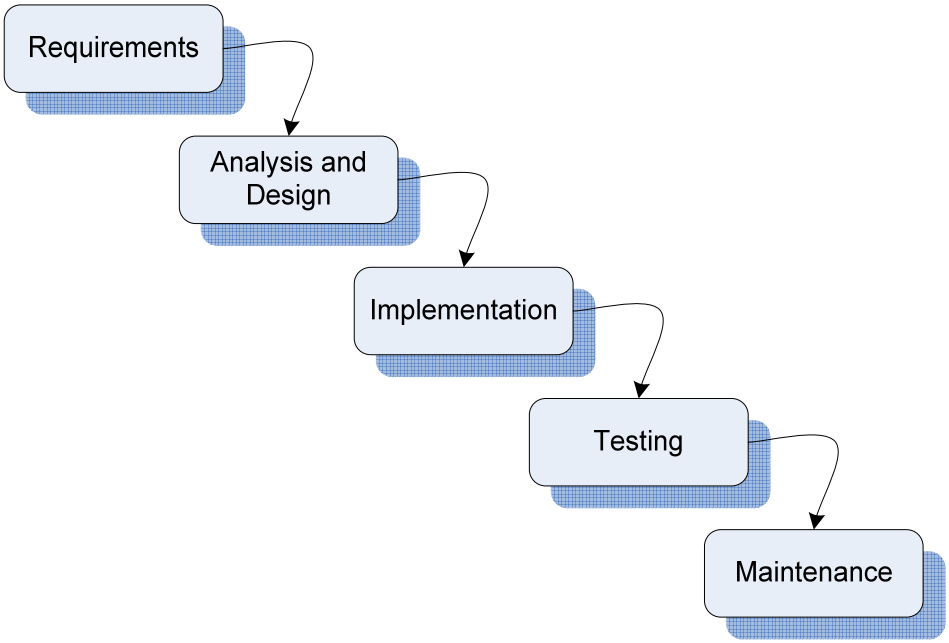
- Outcome
  - Describe the inputs, activities, and outputs of each step in the software development life cycle.

Software Lifecycle



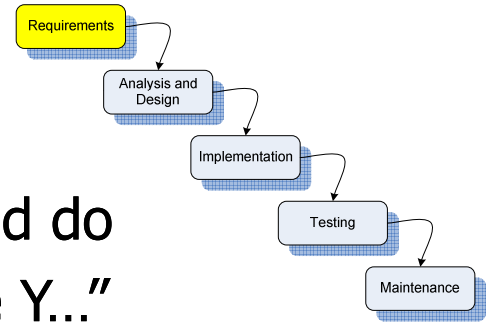
Waterfall Process

Software Lifecycle



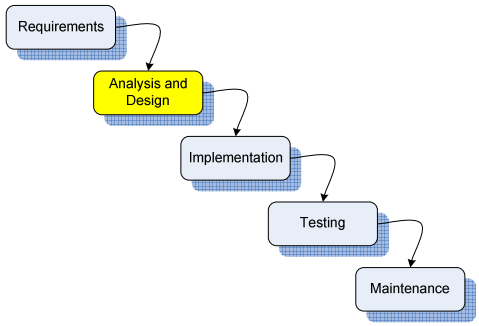
Software Lifecycle

- Requirements
  - Functional
    - What the software should do
    - e.g. "on input X produce Y…"
  - Non-functional (qualitative)
    - Criteria against which the system is measured
    - e.g. "…within 2 seconds or less"



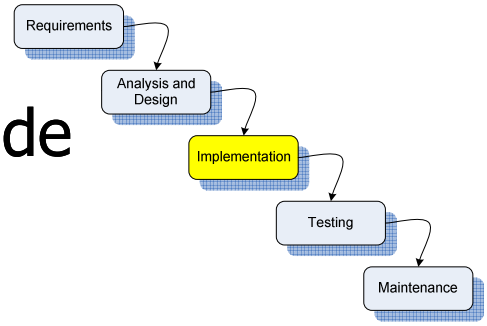
Software Lifecycle

- Analysis and Design
  - Determine architecture
    - System
    - Software
  - Determine what is done in software
  - Create abstract models
  - High-level and low-level
  - Generally, "what" not "how"



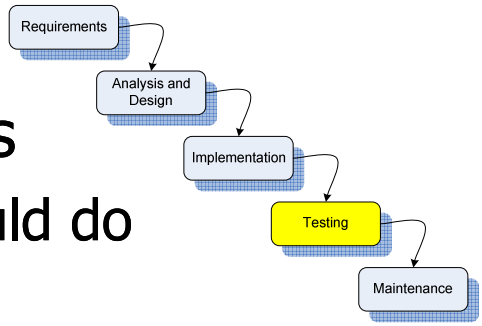
Software Lifecycle

- Implementation
  - Translate design into code
    - Algorithms
    - Objects
    - Functions
    - Control structures
  - i.e. what is generally considered to be "programming"



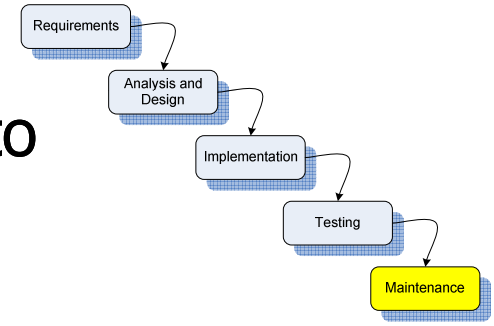
Software Lifecycle

- Testing
  - Validates code two ways
    - That it does what it should do
      - Functional requirements
      - Non-funcitonal requirements
    - That it doesn't do what it shouldn't do
      - Graceful failure
      - Error recovery



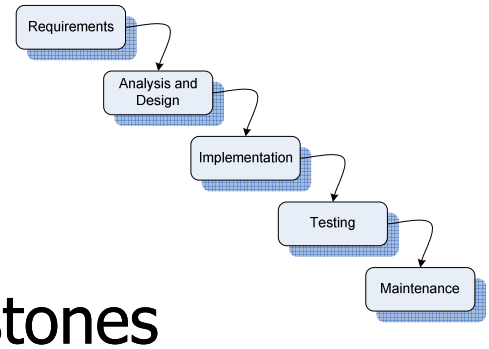
Software Lifecycle

- Maintenance
  - Ongoing development to
    - Fix bugs
    - Add new features
  - Can be more than 80% of man-hours
  - Typically what separates "academic" projects from "production" projects!



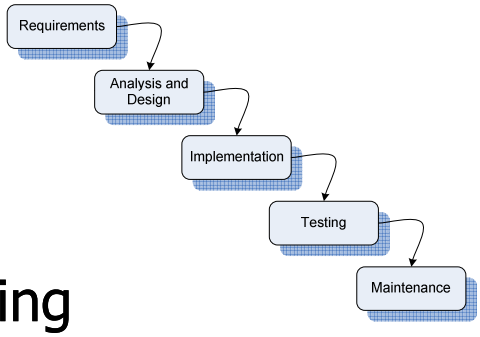
Software Lifecycle

- Waterfall advantages
  - Simplicity
  - Easy to benchmark
  - Clearly delineated milestones



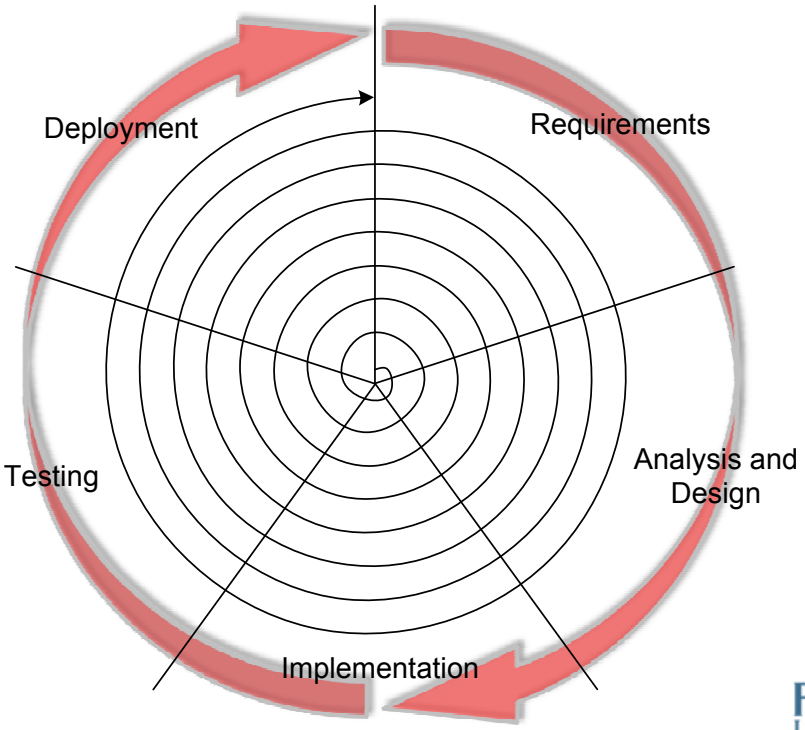
Software Lifecycle

- Waterfall problems
  - Assumptions
    - Stable requirements
    - Stable technologies/staffing
    - Early risk identification
    - Familiarity with the problem domain
    - No need for *feedback* in the system
  - Result: early mistakes are costly

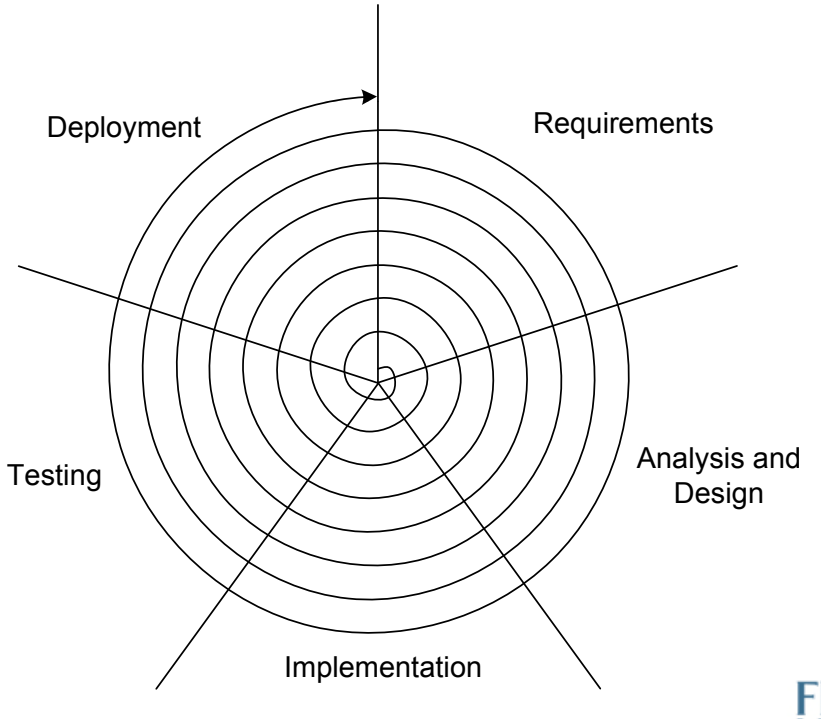


Software Lifecycle

Spiral Process

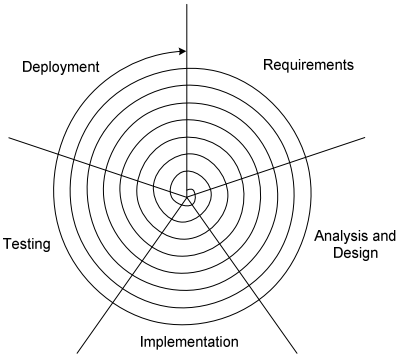


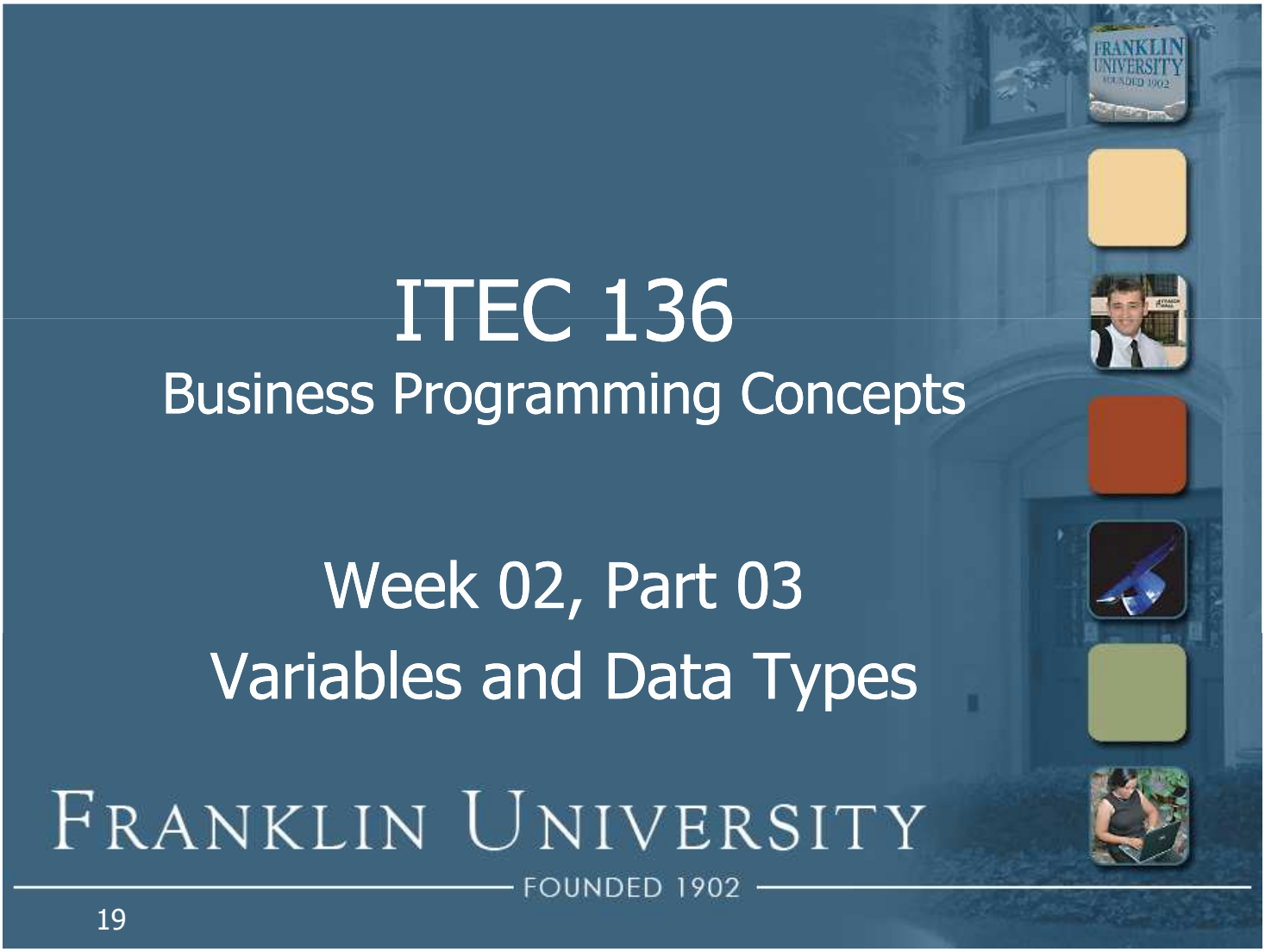
Software Lifecycle



Software Lifecycle

- Advantages
  - Changing requirements
  - Early problem discovery
  - Always working software
  - Extensive feedback
- Problems
  - Architecture suffers





Variables and Data Types

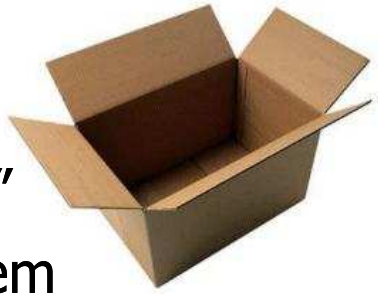
- Outcome
  - Declare, define, and use variables in a script.

Variables and Data Types

- Boxes
  - In the real world:
    - Some boxes are empty
    - Some boxes hold things
    - Box contents can be replaced
    - Boxes can hold more than one thing

Variables and Data Types

- Variables
  - In programming:
    - Some variables are "empty"
    - Some variables hold one item
    - Variable contents can be replaced
    - Variables can hold more than one thing (an array)



Variables and Data Types

- Variables have 4 key properties
  - Have a *name*
  - Have a *value*
  - Have a *data type*
  - Have a *scope*
- Can have operations performed on them

Variables and Data Types

- Declaring a variable

```
var myVariable;
```

var: keyword to create a "box" to hold data.

myVariable: an *identifier*. The name of the variable being created. You invent your own descriptive name for variables.

Variables and Data Types

- Declaring a variable

```
var myVariable;
```

- Declaring multiple variables

```
var myVariable, yetAnotherVariable;
```


Variables and Data Types

- Defining a variable – an initial value

```
var myVariable;
myVariable = 5;
```

The variable name is an l-value (something that can appear on the left hand side of an assignment statement)

Variables and Data Types

- Defining a variable – an initial value

```
var myVariable;
myVariable = 5;
```

Assignment operator: puts the contents on the right hand side into the "box" on the left hand side.

Variables and Data Types

- Defining a variable – an initial value

```
var myVariable;
myVariable = 5;
```

5 is an r-value (something that can appear on the right hand side of an assignment statement).

Variables and Data Types

- Defining a variable – an initial value

```
var myVariable;
myVariable = 5;
```

- Doing both at once

```
var myVariable = 5;
```


Variables and Data Types

- Getting user input into variables

```
var firstName = prompt("Enter your first name");
var age = prompt("Enter your age");
```

prompt: a method of the **window** object that opens an input dialog box with the string parameter as a visual queue. Always returns a string.

Variables and Data Types

- Rules for variable names (identifiers)
  - Cannot be a reserved word (Gosselin, p. 60)
  - Must start with [A-Z, a-z, _, $]
  - Subsequent characters can also include [0-9]
  - No spaces allowed

Variables and Data Types

- Rules for variable names (identifiers)

Example	Valid or invalid?	If invalid, why?
`aSampleID`		
`First_Name`		
`1forTheMoney`		
`$big&tall`		
`document`		
`class`		
`my age`		

Variables and Data Types

- Informal rules for variable names
  - Should not conflict with another built-in identifier.
  - Should use `camelCaseConventions`
  - Should be descriptive of their purpose
    - Exceptions: i, j, k, etc., used as counting loop variables

Variables and Data Types

- Data types
  - Each variable has a *type* that determines which operations can be performed on it.
  - e.g. numbers can have arithmetic performed on them, strings can be concatenated, etc.

Variables and Data Types

- Data types

Data Type	Example	Description
Integers	`42`	A whole number -2^{53} through 2^{53}
Reals	`6.023E23`	A number with a decimal point
Boolean	`true`	Either `true` or `false`
String	`"lorem ipsum"`	A sequence of character data
Undefined		Declared but uninitialized variable
Null	`null`	The "empty" object
Object	`new Date()`	Any user defined object

Variables and Data Types

- **typeof** operator

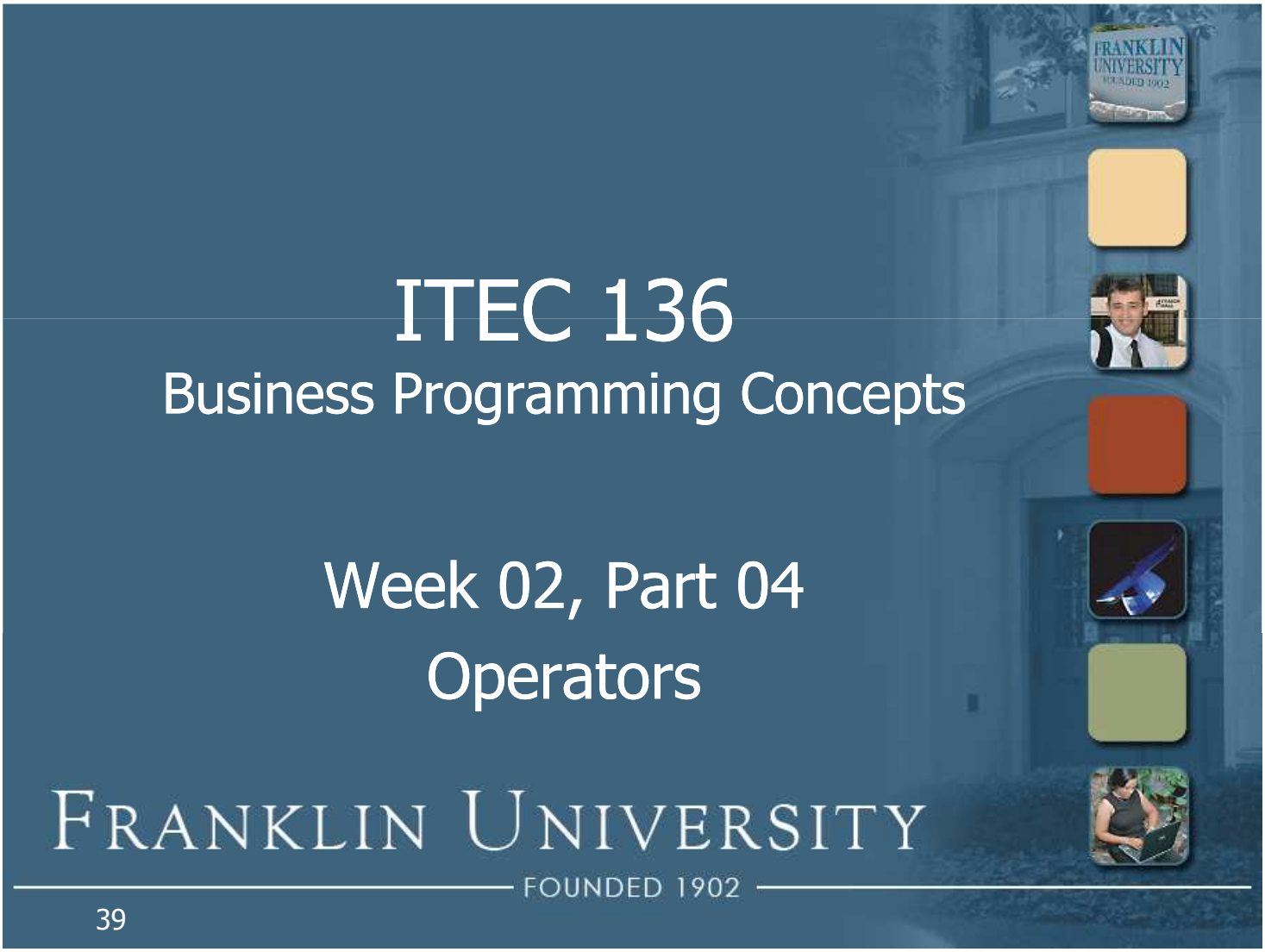
```
var lastName = "Smith";
var numDependents = 3;
var dateOfBirth = new Date(1973, 11, 29);
var canVote = true;
document.writeln(typeof lastName);       // string
document.writeln(typeof numDependents); // number
document.writeln(typeof dateOfBirth);   // object
document.writeln(typeof canVote);        // boolean
document.writeln(typeof (typeof 42));   // string
```

Variables and Data Types

- Scope
  - A range of lines during which the variable is "live."
    - **Static scoping**: lifespan of a variable can be determined by inspecting the source code.
    - **Dynamic scoping**: lifespan of a variable can only be determined as the program is running.

Variables and Data Types

- Scope
  - JavaScript is (on the whole) *statically* scoped.
    - **Global scope**: any variable created (declared) outside of a function or without the **`var`** keyword.
    - **Local scope**: any variable created within a function and using the **`var`** keyword.



ITEC 136

Business Programming Concepts

Week 02, Part 04

Operators

FRANKLIN UNIVERSITY

FOUNDED 1902

JavaScript Operators

- Outcome
  - Describe arithmetic, relational, and logical operators in terms of their input and output data types.

JavaScript Operators

- Data types determine valid operators
  - Can add, subtract, multiply, and divide numbers but not Booleans
  - Can compare numbers and strings but not objects.
  - Can use *and*, *or*, and *not* on Booleans, but not strings

JavaScript Operators

- Arithmetic operators – math

Operator	Description
+	Adds two numbers yielding their sum
- (binary)	Subtracts two numbers yielding their difference
*	Multiplies two numbers yielding their product
/	Divides two numbers yielding their quotient
%	Divides two numbers yielding their remainder
- (unary)	Negates a single number

JavaScript Operators

- Operator precedence
  - Just like in math

```
var x = 4 + 2 * 3 - 1;   // x has value 9
var y = 4 + 2 * (3 - 1); // y has value 8
```

  - Complete table Gosselin p. 95-96

JavaScript Operators

- Relational Operators – comparison

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to
===	Equal to and of same type
!==	Not equal to or not of same type

JavaScript Operators

- Relational Operators – comparison

Expression	Value
`3 < 21`	
`"Fred" <= "Ginger"`	
`"3" < "21"`	
`3 >= 3`	
`"3" == 3`	
`"3" === 3`	
`"3" !== 3`	

JavaScript Operators

- Relational Operators – comparison

Expression	Value
`3 < 21`	true
`"Fred" <= "Ginger"`	true
`"3" < "21"`	false
`3 >= 3`	true
`"3" == 3`	true
`"3" === 3`	false
`"3" !== 3`	true

JavaScript Operators

- Logical Operators – join Booleans

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

JavaScript Operators

- Logical Operators – join Booleans

Expression	Value
`true || false`	
`true && !false`	
`true || !(!false)`	
`true || false && false`	
`true && false || false`	
`!(true || false && !false)`	

- Complete table – Gosselin p. 95-96

JavaScript Operators

- Logical Operators – join Booleans

Expression	Value
`true || false`	true
`true && !false`	true
`true || !(!false)`	true
`true || false && false`	true
`true && false || false`	false
`!(true || false && !false)`	false

- Complete table – Gosselin p. 95-96

JavaScript Operators

- Compound assignment

Operator	Shortcut for
`left += right`	`left = left + right`
`left -= right`	`left = left – right`
`left *= right`	`left = left * right`
`left /= right`	`left = left / right`
`left %= right`	`left = left % right`

JavaScript Operators

- Increment (++) and decrement (--)
  - Shortcut for adding 1 to a variable
  - Pre- versus post- operators
    - Pre- : ++ or -- operation on variable first, then yield the variable value
    - Post- : yield the variable value, ++ or -- operation on the variable last

JavaScript Operators

- Increment (++) and decrement (--)
  - Assume x is 10 initially

Example	New value of y	New value of x
`y = x++`		
`y = ++x`		
`y = x--`		
`y = --x`		

JavaScript Operators

- Increment (++) and decrement (--)
  - Assume x is 10 initially

Example	New value of y	New value of x
`y = x++`	10	11
`y = ++x`	11	11
`y = x--`	10	9
`y = --x`	9	9

JavaScript Operators

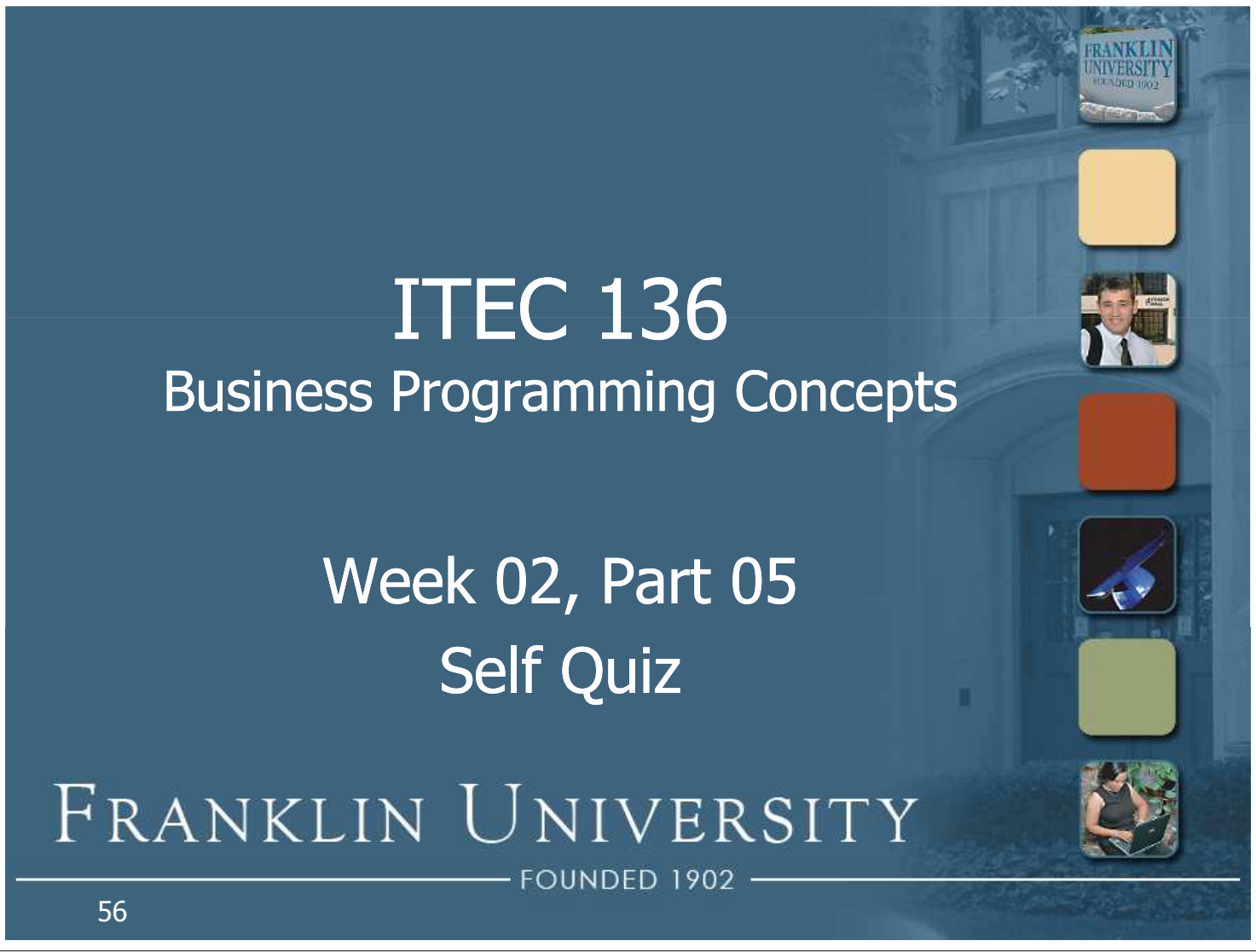
- Conditional operator
  - Syntax:

    <boolean_expression> ? <true_part> : <false_part>
  - Similar to Excel IF function

```
var number = prompt("Enter an integer");
document.writeln("The number was " +
   (number % 2 == 0 ? "even" : "odd"))
```

Questions?

FRANKLIN UNIVERSITY



ITEC 136

Business Programming Concepts

Week 02, Part 05

Self Quiz

FRANKLIN UNIVERSITY

FOUNDED 1902

Self Quiz

- What are the rules about how variables can be named?
- Which kind of operators combine Boolean expressions to create a Boolean result?
- Which kind of operators combine numbers to create a Boolean result?

Self Quiz

- What kinds of operators combine numbers to make a number result?
- What are the stages of the software development lifecycle?
- What do you do in each stage?
- What is the output of each stage?
- Which stage takes the most time?

Self Quiz

- How are the spiral-model and the waterfall-model of software development similar? Different?
- What is a l-value? R-value?

FRANKLIN UNIVERSITY

ITEC 136

Business Programming Concepts

Week 02, Part 06

Upcoming deadlines

FRANKLIN UNIVERSITY

FOUNDED 1902

Upcoming Deadlines

- Pre-class exercise 3 – Due Jan 19
- Homework 2 – Due Jan 19