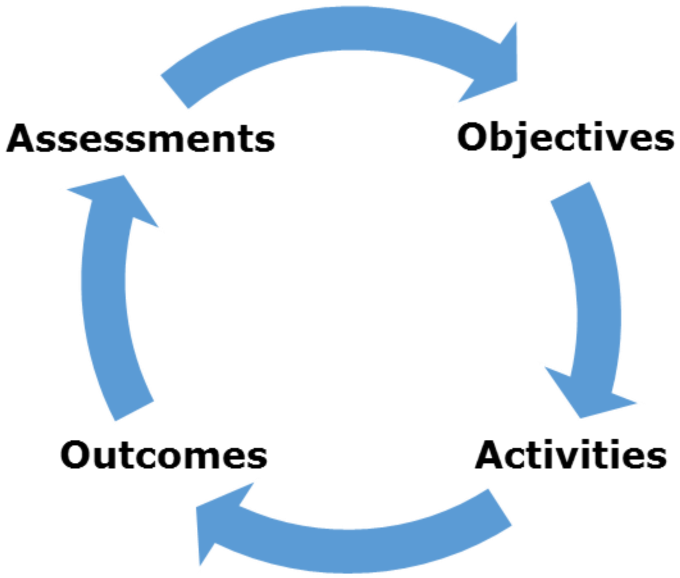
Developing Courses

This is a short overview of developing courses. I try to make it as easy as possible while making sure to plan for entire cycle of the course.

Computer Information Systems needs an introductory programming course that is application focused, rather than mathematical algorithms as Computer Science.

Here are the steps:

The Educational Process



Learning: Unless by accident, learning occurs in four stages: objectives, activities, outcomes, and assessments. This happens anywhere: on a personal basis, in a business, in marketing, and, most importantly, in education.

Objectives: Objectives are goals you want to get out of the process. They are dreams.

Activities: Activities are the individual steps required to achieve each objective. These require both input (direction) and output (results). Also called Course Content.

Outcomes: Outcomes are the measures of completion or competence of each activity. They are grades.

Assessments: Assessments are the comparison of the results of the outcomes to the original objectives. You can complete every activity but not obtain your objective. In this instance, either the objectives need to be changed or the activities.

Creating Objectives, Activities, and Outcomes
Most of the time, academic courses follow requirements from universities and industry certifications. In these cases, the objectives are already provided.

For ITIS, there is already a C-ID course that follows the recommendations of the Association for Computing Machinery, ITIS-130 Introduction to Programming Concepts and Methodologies. I will use this outline to develop the course:

The C-ID specification provides all of the course requirements except the outcomes. I make a table aligning the three areas:

Spec Course Objectives	Spec Course Content	My Course Outcomes
		Over the course, each student will create a project application with:
• identify basic programming concepts	• Programming concepts • Program development lifecycle	Document the purpose, audience, platform, timeline, and goal.
• choose an appropriate data structure for modeling a simple problem	• Requirements determinants and analysis • Modular design	Document elements and modules, along with their purposes. Create a UML diagram of the project.
• use primitive data types and data structures offered by the development environment	• Techniques for modeling program structures	Create pseudocode of the program flow and activities.
• write simple applications that relate to a specific domain	• Program design	Create simple programs using each of the basic constructs.
• design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions	• Coding • Variables • Literals • Types • Expressions • Procedures • Functions • Parameters • Operators and operations • Decision logic • Looping • Sub-procedures • Passing parameters	Complete a master project encompassing all of the simple constructs covered in the course.
• test applications with sample data	• Unit testing	Verify that all constructs are covered and working.
• apply core program control structures	• Control structures	Provide a start, pause/save/resume, and finish to the application

Activities

Now that you know what students need to know, the activities that you want to focus on, and the measurements you will use to gauge their success, you can put together the actual class. My greatest difficulty is coming up with exercises for each item, but why re-invent the wheel? Someone has surely already done this.

The first stop is http://Google.com/books. When using Google Search, be sure to check out Tools below the magnifying glass on the right-side of the search box. You can use Advanced Search to focus searches, even down to specific sites. Most books have a table of contents you can check out.

Once you find some possibilities, jump over to http://vitalsource.com. They provide e-books for many of the publishers. Once you register as an instructor, you can search for your book and get an e-book instantly.

Check the books against your course requirements table. Once you have found some good possibilities, go to the publishers' websites and find your representative. Making regular contact with the publisher reps will make it easy to get desk copies, pre-release copies, and even take part in the development of new books.

Most textbooks provide more exercises than you really need, so be sure to weed out any that do not support your course objectives, content, or outcomes. Students are adults, and any adult does not like to waste their time with irrelevant activities.

Exams

Exams or projects are the best way to verify that students have mastered the necessary knowledge or skills for the course. They should align with the course outcomes to make it easy to assess the student, the module, and the course as a whole.

In my course specification table previously, I aligned the objectives, activities, and outcomes to make it easy to do this. Try not to mix outcomes in a single step in the activities. Also, try to make each activity self-contained so that failure of a previous step does not effect the outcome of a later step. In project-based assessments, this is difficult to do, so I offer to complete the parts of a step that a student cannot do, in exchange for points that it is worth, so that the students can complete all that they can do. This allows me to be precise at exactly what they can and cannot do.

Analyzing Outcomes

Here is where the little table I created for the course really comes in handy.

Assessment Results		
Course Outcome	Criteria	Results:
Method: Over the course, each student will create a project application with:		# of projects submitted
Document the purpose, audience, platform, timeline, and goal.	Student documented a valid purpose, audience, platform, timeline, and goal.	# of students completing all
Document elements and modules, along with their purposes.	Student listed all of the elements and modules with purpose.	# complete
Create a UML diagram of the project.	Accurate UML diagram	# complete
Create pseudocode of the program flow and activities.	Logical pseudocode of the program flow and activities	# complete
Create simple programs using each of the basic constructs.	Complete each section exercise.	# complete
Complete a master project encompassing all of the simple constructs covered in the course.	Final project contains all constructs	# complete
Verify that all constructs are covered and working.	All constructs have been verified to work on their own.	# complete
Provide a start, pause/save/resume, and finish to the application	The final project has working start, pause/save, and resume.	# complete

Note: The dotted line before UML indicates where I realized that this was a separate outcome and needed its own row.

Finalization

From the Results, you can calculate the percent of success for each outcome and you have a simple reference of where students may need some help.