An Introduction to STEM Programming with Python 3

Bonus Chapter 1
Problem Solving

James M. Reneau Ph.D.
Shawnee State University, Portsmouth OH

ISBN: In Process

James M. Reneau Ph. D.
P. O. Box 278
Russell, KY 41169-0278

Version 2019-06-30a

Table of Contents

Illustration Index

Index of Tables

Bonus Chapter One — Problem Solving

Introduction

Intro text

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Define what an algorithm is.
- Explain the five processes within the IDEAL Framework.
- Analyze the completeness of an algorithm using a checklist of questions.
- Produce a step-by-step set of instructions for a simple problem using the IDEAL framework.

Prerequisites

This bonus chapter stands alone and may be optionally included anywhere in your studies.

B1A — What is a Problem

Solving problems algorithmically is one of the most difficult things we have to do as humans, but we do it all the time. This brief paper will introduce a method for creating step-by-step solutions to problems. This paper is specifically geared toward creating algorithms that will be implemented as computer programs, but the general method can be used to create algorithms in many domains.

First a few definitions, a problem is "something difficult to deal with" (Problem, 2015) and an algorithm is "a set of steps that are followed in order to solve a ... problem" (Algorithm, 2015). Problems can be broken into tree main parts: 1) the initial conditions; 2) the goals that you want at the end of the solution; and 3) the barriers that are in the way of achieving the solution.

B1B — The IDEAL Framework

There are many methods used to solve a problem. The method proposed in this paper is a combination of the IDEAL framework originally by Bransford (1984) combined with the “Basic Strategy” by Dr.

Vanconcelos (2007) of Johns Hopkins University. This framework will help you to create an algorithm to solve a problem, not the final implementation.

Think of the acronym IDEAL as a way to remember the 5 steps to solving a problem algorithmically.

I — Identify the problem.
D — Define and represent the problem.
E — Explore possible strategies.
A — Act on your strategy.
L — Look back and evaluate at your solution.

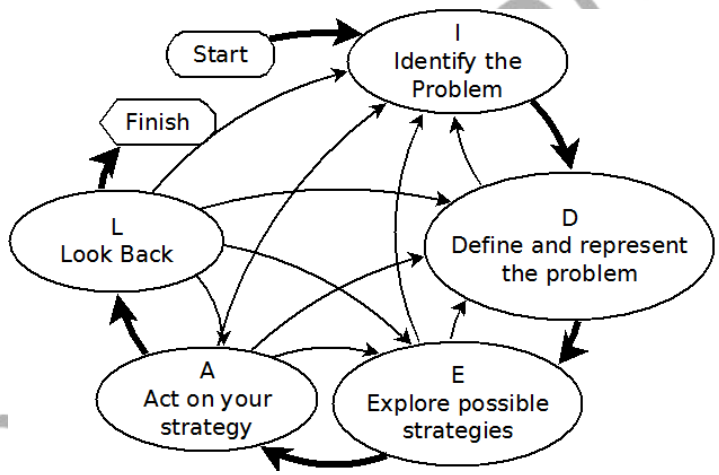

Illustration 1: IDEAL Framework

This process is not a waterfall type of method, where you move directly from one phase to another in a single direction. We will be moving both forward and backwards through the phases until we can find a solution that really works.

As we move from one phase to the next we need to ask ourselves a group of questions. These questions are suggested to help us make sure that we are ready to move on. It is virtually impossible to create a solution for a problem that you do not understand.

The IDEAL framework can also be represented as a flow chart. After each step use the question guide (below) and decide if the step was fully understood and works to solve the problem. If we are "good" then move on to the next step. If we have not found a successful solution or do not fully understand the problem then we need to go back to prior steps and gather more information. Remember stepping back is not a failure but an intentional refinement that makes the process work.

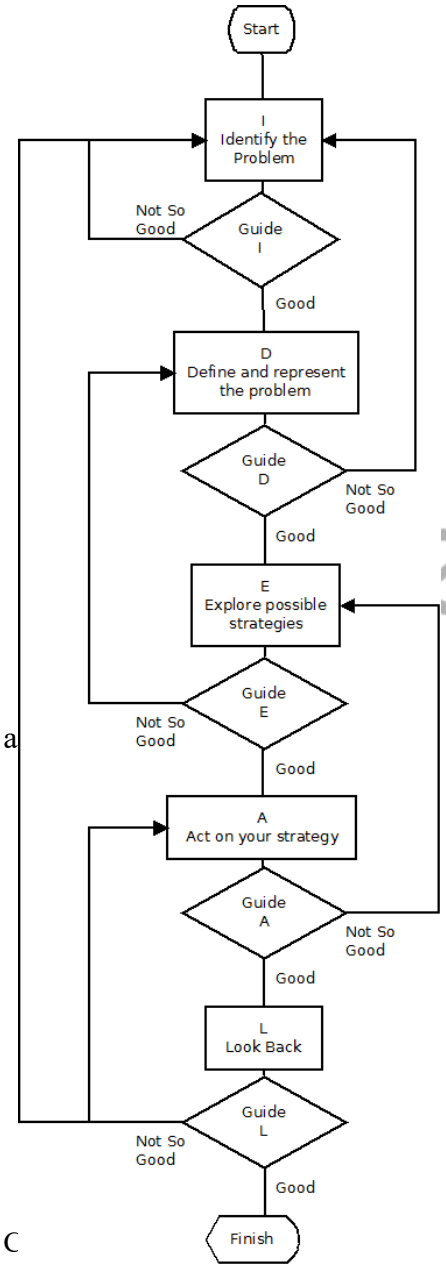

Illustration 2: IDEAL Framework Flow Chart

The following discussion about the five phases and the questions that you need to ask yourself in the process. The questions have been tailored for the computer programming domain but can easily be restated to guide in the creation of almost any algorithm.

I — Identify the Problem

In the first step we need to fully understand the problem as it was given to us. We need to figure out what the initial conditions are when the problem starts, what information we will be given (data), what computations need to be made, and what the final results will be.

We DO NOT NEED a finished solution here. Just a complete understanding of the problem. Sometimes this is the hardest thing to do.

We need to ask ourselves the following questions:

1. Do I fully understand every word in the problem and why they are there?
2. Can I rewrite the problem in my own words (not paraphrasing) or sketch it out?
3. What future problems may be caused or happen as a result of the current problem?
4. Have I seen a problem like this one in the past and do I already have a “good” solution for it?
5. What is the problem's domain (computer programming, manufacturing, play, business, algebra, geometry…)?

D — Define and represent the problem

In this phase, you need to take our identification of the problem and now define the problem in terms that we can act upon. We are not arriving at a solution, at this point, but actually developing a formal definition of the problem. All the gray areas need to be resolved.

Questions you need to as yourself:

1. What are the initial conditions (givens) for the problem?
2. What inputs, from the user or other source, are defined in the problem?
3. What calculations or other actions are described in the problem?
4. What outputs (messages, drawings, data) does the problem ask for?
5. What is the terminal condition (when is the problem finished)?
6. What constraints or barriers (external or internal) exist that will make solving this problem more difficult?
7. What existing algorithms can be used to solve this?
8. Can we simplify this problem by breaking all or part into smaller problems that can be solved independently (functions, subroutines, classes...)?

E — Explore possible strategies

Now that you completely understand the problem and have eliminated all the "fuzzy" ideas about the problem, it is now time to create strategies to solve it. During this phase we may develop several algorithms or methods to solve the problem. The final product of this step is one "good" algorithm that we can implement.

Questions to ask:

1. Can we solve this problem by hand (on paper) and describe the process that we used to solve it?
2. How am I going to store the data (in simple variables, in arrays, in tables…)?
3. Assign your own words (variables) to the data elements in the problem. Have you identified ALL data elements?
4. How can we apply sequence (a step by step progression, one step after another without skipping a step) to the problem?
5. How can we apply selection (parts of an algorithm that only happens under certain conditions — logic) to the problem?
6. How do we need to apply iteration (looping through an algorithm or part of one a specific number of times or while/until a condition is met) to the problem?
7. Can we describe several combinations of sequence, selection, and iteration to completely solve the problem?

A — Act on your strategy

In the previous phase you have explored many possible strategies and have identified an algorithm that will work. It is now time to formally describe the algorithm to solve the problem. This is done in different manners for different domains.

1. Can we draw flow charts or other diagrams to completely model the algorithm?
2. Can you write the algorithm in pseudocode or a set of step by step instructions?
3. If you run-through your algorithm (pretending to be a computer or person with no prior knowledge of the problem), does it create the desired result?
4. Have you looked at your solution critically:
    - Is it efficient?
    - Is it easily understood?
    - Is it translatable?

L — Look back and evaluate at your solution

No human is perfect, and rarely do we create a "perfect" algorithm. In this last phase of the IDEAL framework, we need to continue to look critically at our algorithm and make needed adjustments to it.

In the domain of computer programming, we will be evaluating the algorithm as we write the computer code, as we test the program, and as the program is being used. By identifying issues and problems and going back through this framework we can refine and correct the algorithm.

Questions to ask ourselves:

1. Can I actually program this algorithm using the tools at my disposal?
2. Are there existing modules/functions/objects that I can use to more easily and reliably implement the code?
3. Can I test the code with real word cases? Does it work? Completely?
4. Is the user satisfied that the problem is actually solved?
5. Is the environment changing and will the algorithm require updates and changes to continue to work?

Summary

Goes here

Important Terms

- IDEAL
- act
- algorithm
- barrier
- define
- domain
- explore
- identify
- look back
- method
- problem
- solution
- step-by-step
- waterfall

Exercises

Here

Word Search

```
h o t y r i - j u m t a l t
o t m e t h o d n i s c o u
- n i d e n t i f y t t e d
n t f y e y r l m o e a b e
```

```
i p b l t w f b a t p l s f
t r e b n a r a b e - g o i
f o x r s t · r i c b o l n
i b p l d e t r z x y r u e
d l l s o r u i r i - i t i
e e o p m f r e l p s t i i
a m r u a a a r v o t h o b
l - e f i l r m t i e m n l
r y o d n l v c f m p a b r
a f l o o k · b a c k m t o
```

IDEAL, act, algorithm, barrier, define, domain, explore, identify, look back, method, problem, solution, step-by-step, waterfall

References

Bransford, J. (1984). The ideal problem solver. W. H. Freeman and Company.

Vasconcelos, J. (2007). Basic Strategy for Algorithmic Problem Solving. (2007). . Unpublished class notes. Retrieved from http://www.cs.jhu.edu/~jorgev/cs106/ProblemSolving.html on 2016-03-01.

Algorithm. (2015). Definition from Merriam-Webster Dictionary. Retrieved from http://www.merriam-webster.com/dictionary/algorithm on 2015-03-01.

Problem. (2015). Definition from Merriam-Webster Dictionary. Retrieved from http://www.merriam-webster.com/dictionary/problem on 2015-03-01.