# Evidence-based Learning Strategies

Amaldev Manuel

School of Mathematics & Computer Science
IIT GOA

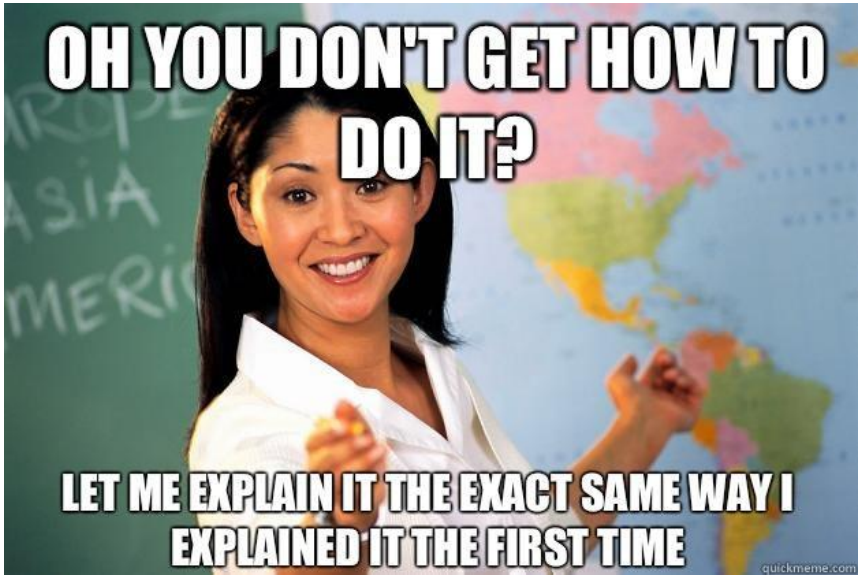
# The Many Plagues of Indian CFTIs

- Variegated Classrooms
- Student's apathy
- Limited Liberal Arts Portfolio
- Poor Caste and Gender Sensitisation
- Insufficient Mental Health infrastructure
- **Lack of Pedagogical Training of Faculty**


*Plague doctor from the time of Bubonic plague*

# The Many Functions of a Teacher

1. Repository of Knowledge
2. **Orchestrator of the Learning Process**
3. Inspire, Nurture, Mentor, Organize, Plan, Evaluate, Assess, …




'*Unhelpful High School Teacher*' meme

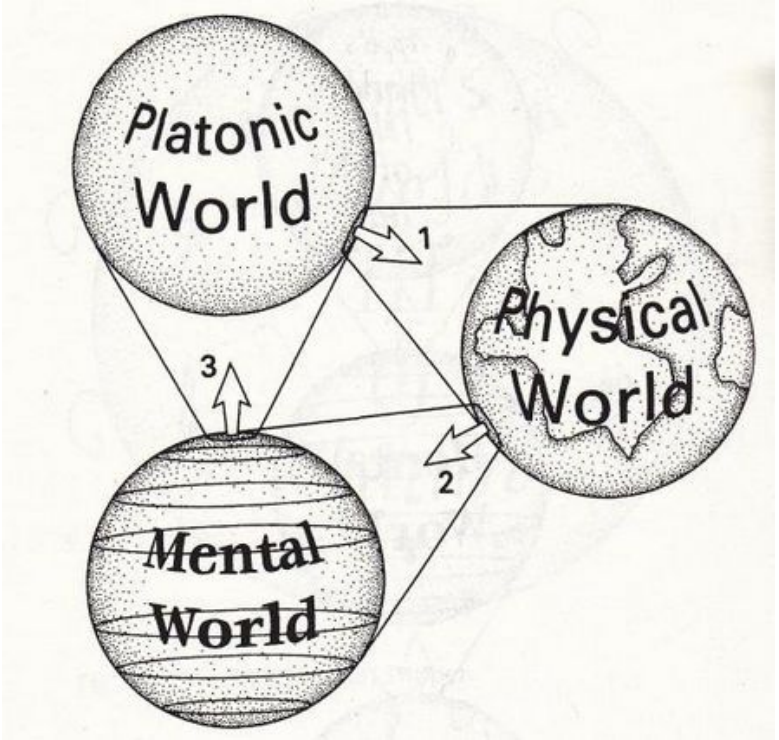
# ‘Teacher Effect’

- Teacher’s *Value-Added*
- **Increase in students’ stock of human capital**, however that may be achieved – better communication to students, classroom management, encouragement of greater effort by students or parents, etc. [Jackson, Rockoff, Staiger, 2014]
- Wide variation in Teacher Effectiveness
- Variation in Teacher Effect is greater in Mathematics than in English or in Reading.

*The Anatomy Lesson of Dr. Nicolaes Tulp – Rembrandt*

# When ‘Teacher Effect’ is crucial

- Not all mathematics are the same!
- ‘Empirical’ theories in Mathematics are easier to understand
- Difficulty seems to be proportional to how far removed the abstractions are from human experience




*Penrose’s three world model*

# A Case Study : Monads in Functional Programming

In functional programming, a **monad** is a structure that combines program fragments (functions) and wraps their return values in a type with additional computation.

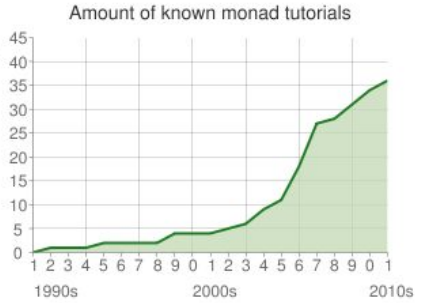
Given any well-defined, basic types *T*, *U*, a monad consists of three parts:

- A type constructor *M* that builds up a monadic type *M T*
- A type converter, often called **unit** or **return**, that embeds an object *x* in the monad:
  `unit : T → M T`
- A combinator, typically called **bind** and represented with an infix operator `>>=` that unwraps a monadic variable, then inserts it into a monadic function/expression, resulting in a new monadic value:
  `(>>=) : (M T, T → M U) → M U` so if `mx : M T` and `f : T → M U`, then `(mx >>= f) : M U`

# Monad tutorials timeline

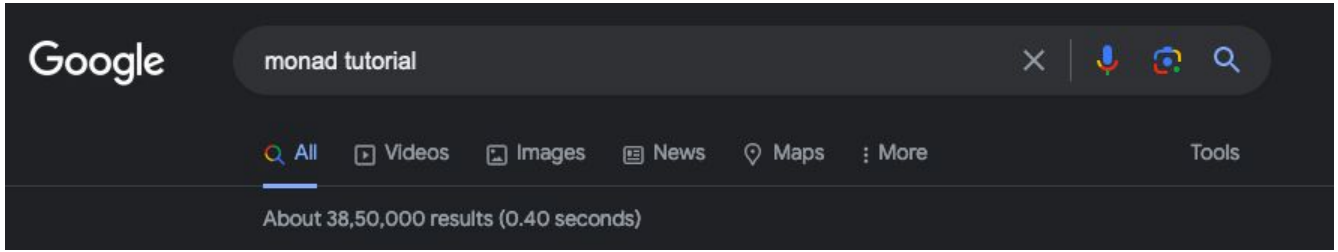
This is a comprehensive timeline of monad tutorials and related articles.

Please update this list as it becomes outdated! If you find a tutorial, article, post, comment, or message that stands on its own as an explanation of monads, then please take a moment to paste the link somewhere on this page (register a throwaway account, if you prefer). The date, author, and blurb can be added later. This will greatly help others who are using this list as a resource for learning about monads.




**Contents** [hide]





# Story of a Monad Tutorial

Joe Haskeller is trying to learn about monads. After struggling to understand them for a week, looking at examples, writing code, reading things other people have written, he finally has an “aha!” moment: everything is suddenly clear, and Joe understands Monads!

What has really happened, of course, is that Joe’s brain has fit all the details together into a higher-level abstraction, a metaphor which Joe can use to get an intuitive grasp of monads; suppose that Joe’s metaphor is that Monads are like Burritos.

Here is where Joe badly misinterprets his own thought process: “Of course!” Joe thinks. “It’s all so simple now. The key to understanding monads is that they are Like Burritos. If only I had thought of this before!”

The problem, of course, is that if Joe HAD thought of this before, it wouldn’t have helped: the week of struggling through details was a necessary and integral part of forming Joe’s Burrito intuition, not a sad consequence of his failure to hit upon the idea sooner.

But now Joe goes and writes a monad tutorial called “Monads are Burritos,” under the well-intentioned but mistaken assumption that if other people read his magical insight, learning about monads will be a snap for them. “Monads are easy,” Joe writes. “Think of them as burritos.” Joe hides all the actual details about types and such because those are scary, and people will learn better if they can avoid all that difficult and confusing stuff.

Of course, exactly the opposite is true, and all Joe has done is make it *harder* for people to learn about monads, because now they have to spend a week thinking that monads are burritos and getting utterly confused, and then a week trying to forget about the burrito analogy, before they can actually get down to the business of learning about monads.

***Brent Yorgey*** *(https://byorgey.wordpress.com/2009/01/12/abstraction-intuition-and-the-monad-tutorial-fallacy/)*

# “Young man, in mathematics you don’t understand things. You just get used to them”

Reply to a physicist friend who had said "I'm afraid I don't understand the method of characteristics".

Source: en.wikiquote.org/wiki/John_von_Neumann

**“When we talk mathematics, we may be discussing a secondary language built on the primary language of the nervous system”**

Source: en.wikiquote.org/wiki/John_von_Neumann