DYALOG

APL Germany

News from Dyalog

Gitte Christensen, CEO

Morten Kromberg, CTO

Status

- For some years now we have declared ourselves ready to start marketing Dyalog and APL
- There is a growing demand for new APL'ers to take over where the old ones left (or are now leaving/retiring)
- But young people are not what they used to be ☺

Status

- One of our insights is that maybe we need young people to talk to young people
- - and besides we also need a generation change at Dyalog like so many other APL shops

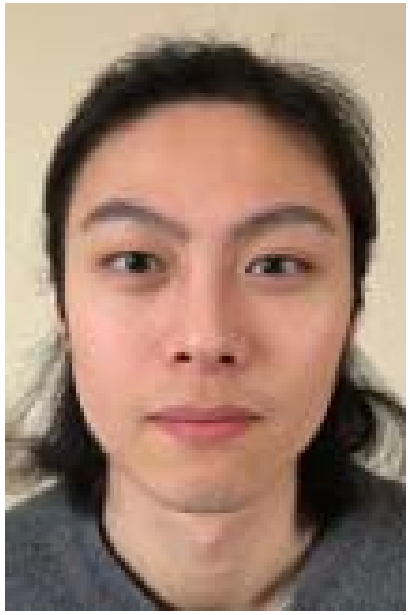
New faces

- Stine Kromberg,
  Msc Business Administration and Information Management With a Minor in Maritime Business
- Stine is managing the accounts at Dyalog and is also a competent project manager

New faces

- Shuhao Yang
  Msc Quantum Technologies

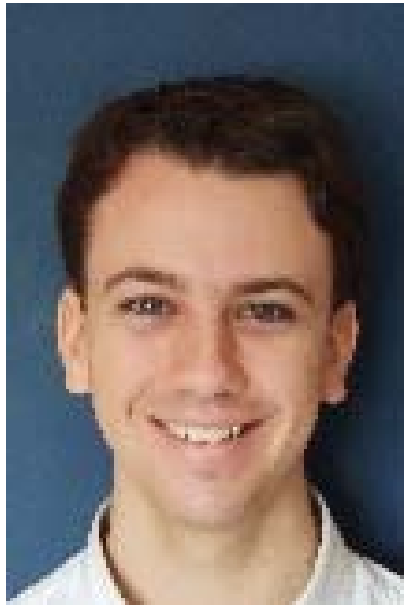
- Shuhao is part of the interpreter development team at Dyalog



New faces

- Rodrigo Girão Serrão
  Msc Mathmatics and Informatics

  Rodrigo has a great passion for teaching and is currently working on a rework of Bernard Legrand's tutorial on Dyalog APL



New faces

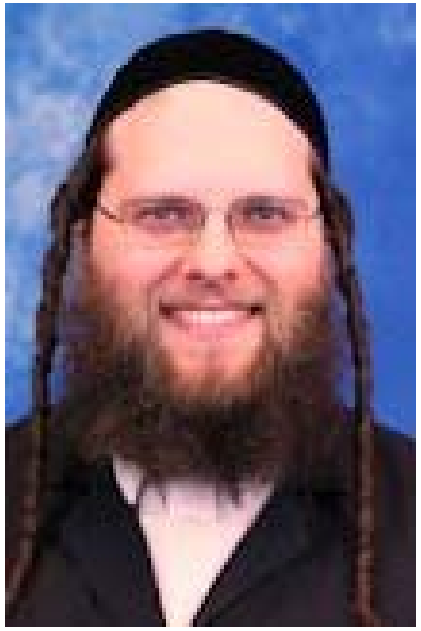
- Karta Kooner
  PhD Physics, String theory

  Karta has joined the interpreter development team

You already know

- Adam Brudzewsky
  Fell in the APL pot as a kid

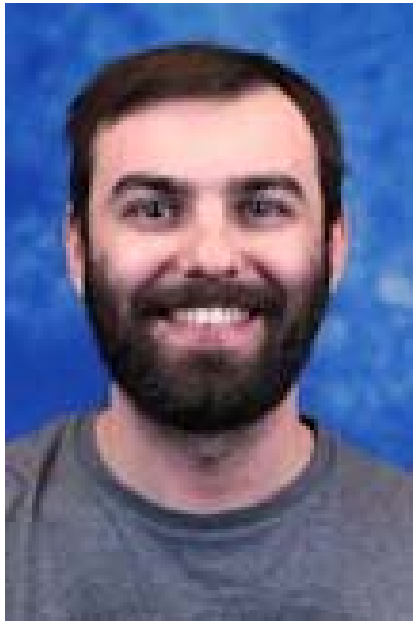
  Adam is participating in language design and is a vigorous APL evangelist in particular through the APL Orchard



You already know

- Richard Park
  Msc Physics

  Richard is developing training material and webinars on APL and is now looking for opportunities to learn application development



You already know

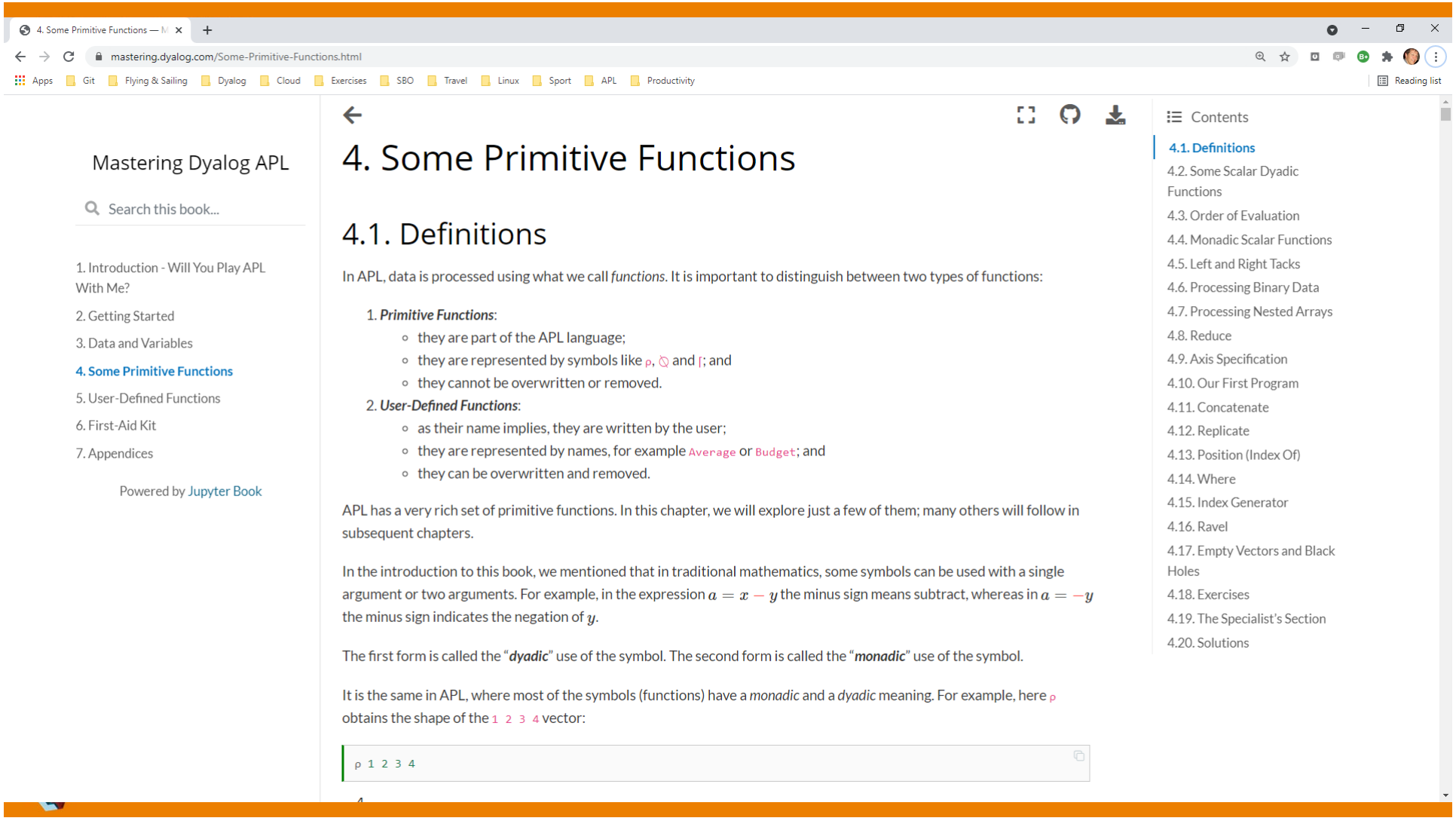
- Aaron Hsu
  PhD Information Technology

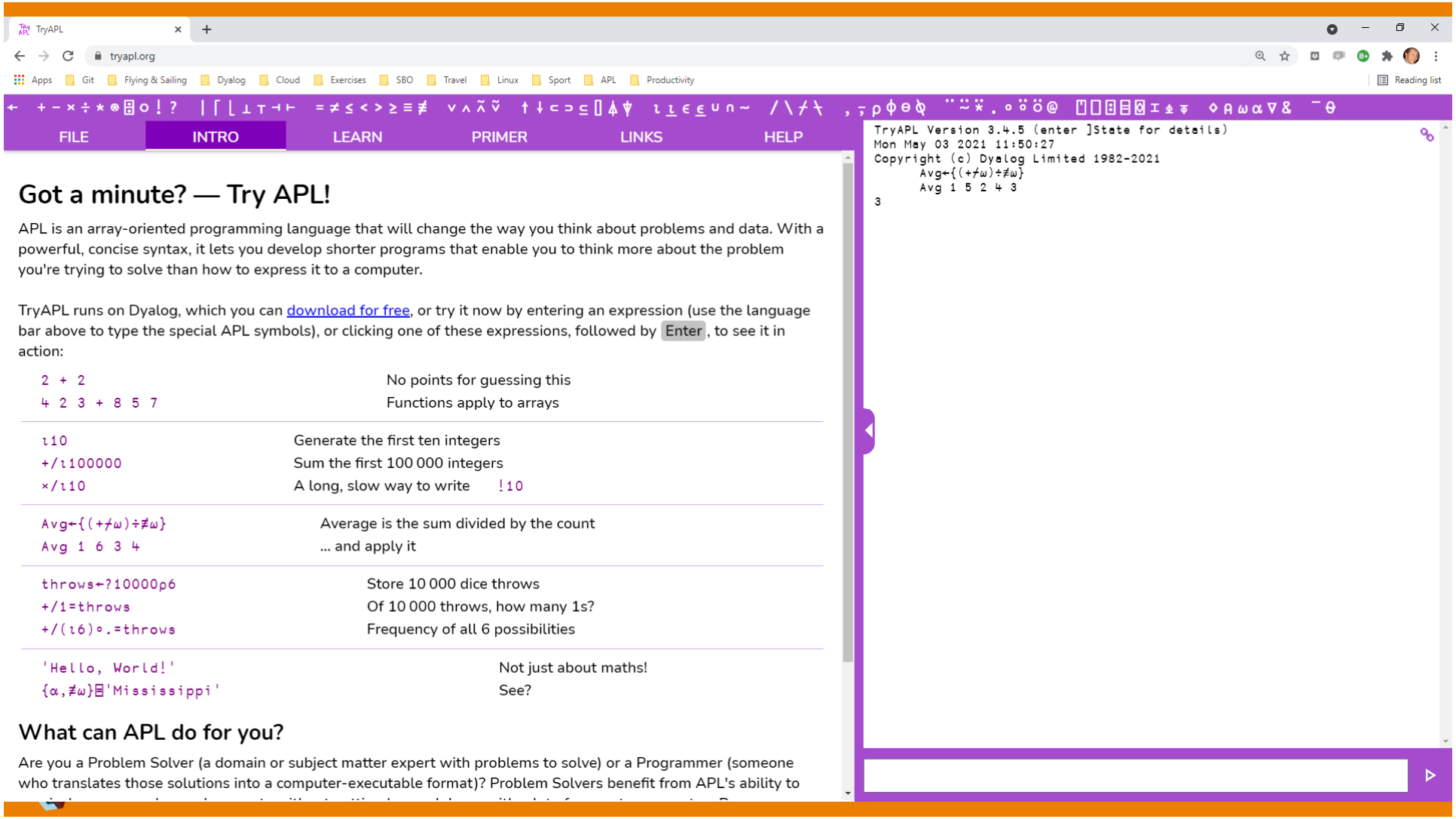
  Aaron is working on the co-dfns compiler and a fierce APL evangelist especially targeting the Functional computing community and HPC

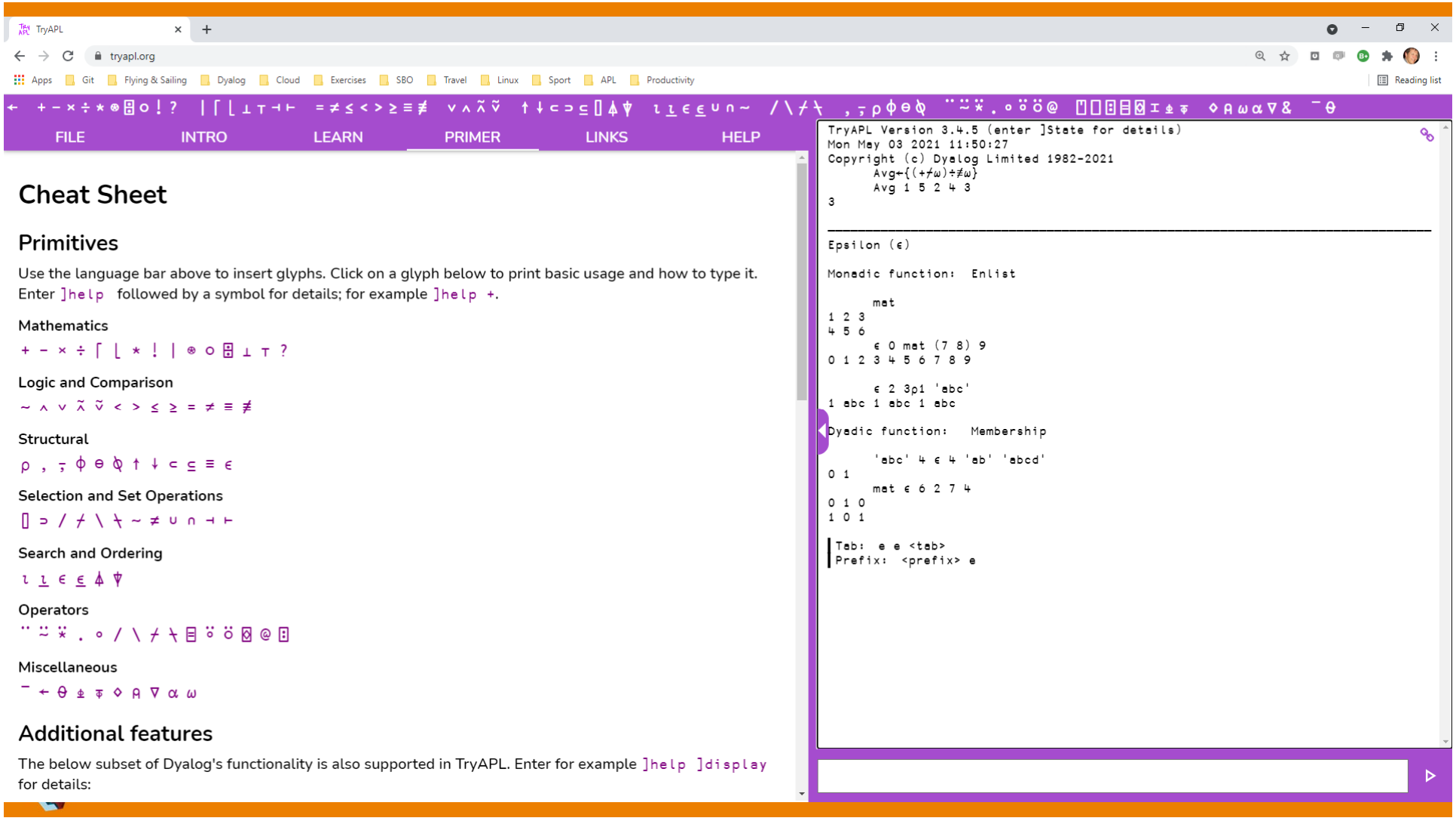
- So what are all these 35- up to?

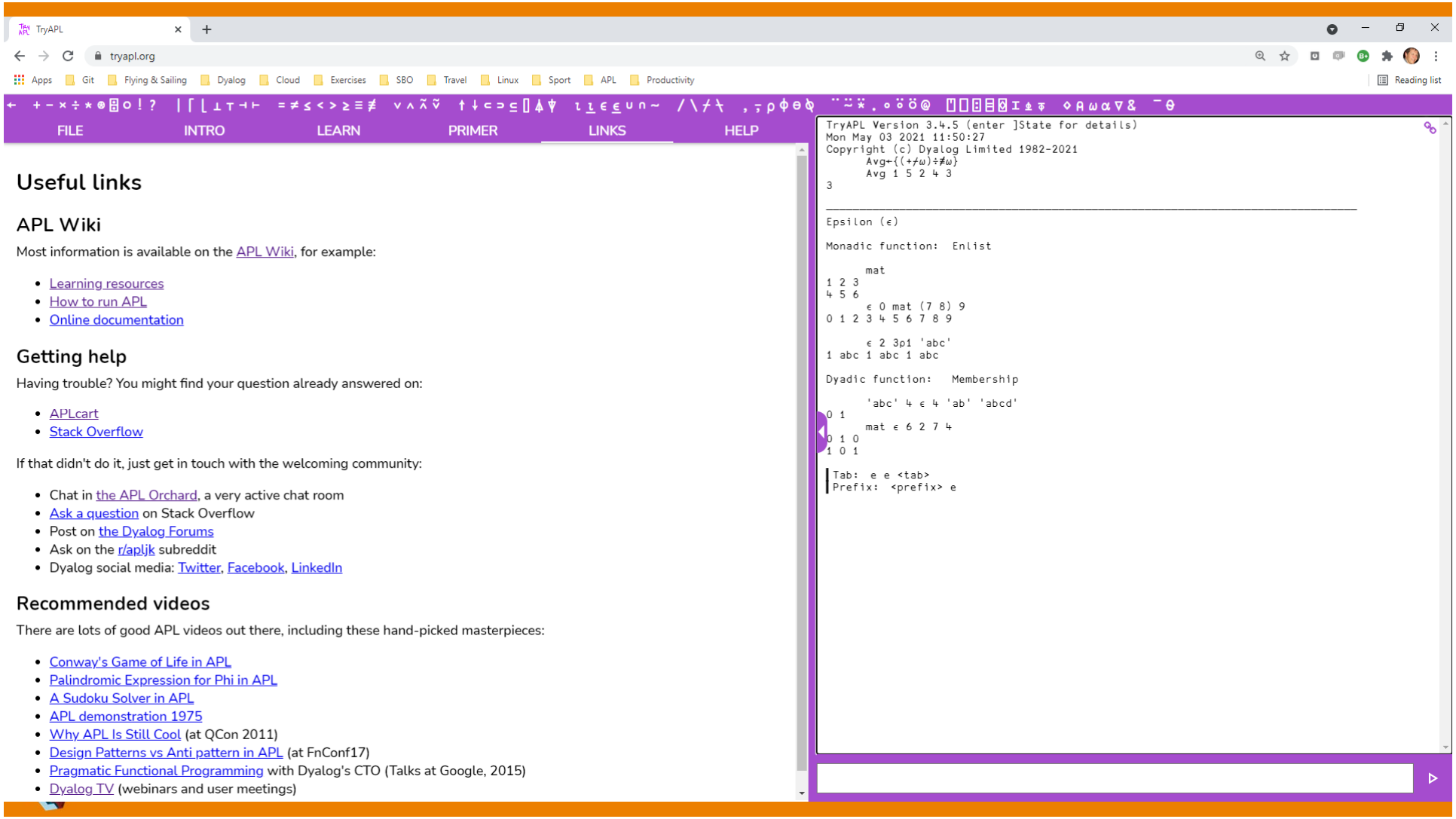
Documentation and Training Materials

- Reworked Mastering Dyalog APL
- Maintain high frequency of webinars
  - And webcasts
- New and improved TryAPL
- Open Source Project Templates
  - Running APL applications as services / daemons under Windows and Linux
  - Deploying and operating secure and load-balanced solutions in Dyalog APL
  - Continuous integration, generating containers from APL code in GitHub, deploying to the cloud


4. Some Primitive Functions — M
mastering.dyalog.com/Some-Primitive-Functions.html
Apps Git Flying & Sailing Dyalog Cloud Exercises SBO Travel Linux Sport APL Productivity
Reading list
Mastering Dyalog APL
Search this book...
1. Introduction - Will You Play APL With Me?
2. Getting Started
3. Data and Variables
4. Some Primitive Functions
5. User-Defined Functions
6. First-Aid Kit
7. Appendices
Powered by Jupyter Book
4. Some Primitive Functions
4.1. Definitions
In APL, data is processed using what we call functions. It is important to distinguish between two types of functions:
1. Primitive Functions:
- they are part of the APL language;
- they are represented by symbols like ⍴, ⍬ and ⌈; and
- they cannot be overwritten or removed.
2. User-Defined Functions:
- as their name implies, they are written by the user;
- they are represented by names, for example Average or Budget; and
- they can be overwritten and removed.
APL has a very rich set of primitive functions. In this chapter, we will explore just a few of them; many others will follow in subsequent chapters.
In the introduction to this book, we mentioned that in traditional mathematics, some symbols can be used with a single argument or two arguments. For example, in the expression $a = x - y$ the minus sign means subtract, whereas in $a = -y$ the minus sign indicates the negation of y.
The first form is called the "dyadic" use of the symbol. The second form is called the "monadic" use of the symbol.
It is the same in APL, where most of the symbols (functions) have a monadic and a dyadic meaning. For example, here ⍴ obtains the shape of the 1 2 3 4 vector:
⍴ 1 2 3 4
Contents
4.1. Definitions
4.2. Some Scalar Dyadic Functions
4.3. Order of Evaluation
4.4. Monadic Scalar Functions
4.5. Left and Right Tacks
4.6. Processing Binary Data
4.7. Processing Nested Arrays
4.8. Reduce
4.9. Axis Specification
4.10. Our First Program
4.11. Concatenate
4.12. Replicate
4.13. Position (Index Of)
4.14. Where
4.15. Index Generator
4.16. Ravel
4.17. Empty Vectors and Black Holes
4.18. Exercises
4.19. The Specialist's Section
4.20. Solutions


TryAPL
tryapl.org
Apps Git Flying & Sailing Dyalog Cloud Exercises SBO Travel Linux Sport APL Productivity Reading list
FILE INTRO LEARN PRIMER LINKS HELP
Got a minute? — Try APL!
APL is an array-oriented programming language that will change the way you think about problems and data. With a powerful, concise syntax, it lets you develop shorter programs that enable you to think more about the problem you're trying to solve than how to express it to a computer.
TryAPL runs on Dyalog, which you can download for free, or try it now by entering an expression (use the language bar above to type the special APL symbols), or clicking one of these expressions, followed by Enter, to see it in action:
2 + 2 No points for guessing this
4 2 3 + 8 5 7 Functions apply to arrays
⍳10 Generate the first ten integers
+/⍳100000 Sum the first 100 000 integers
×/⍳10 A long, slow way to write !10
Avg←{(+⌿⍵)÷≢⍵} Average is the sum divided by the count
Avg 1 6 3 4 ... and apply it
throws←?10000⍴6 Store 10 000 dice throws
+/1=throws Of 10 000 throws, how many 1s?
+/(⍳6)∘.=throws Frequency of all 6 possibilities
'Hello, World!' Not just about maths!
{⍺,≢⍵}⌸'Mississippi' See?
What can APL do for you?
Are you a Problem Solver (a domain or subject matter expert with problems to solve) or a Programmer (someone who translates those solutions into a computer-executable format)? Problem Solvers benefit from APL's ability to
TryAPL Version 3.4.5 (enter]State for details)
Mon May 03 2021 11:50:27
Copyright (c) Dyalog Limited 1982-2021
Avg←{(+⌿⍵)÷≢⍵}
Avg 1 5 2 4 3
3


TryAPL
tryapl.org
Apps Git Flying & Sailing Dyalog Cloud Exercises SBO Travel Linux Sport APL Productivity
Reading list
FILE INTRO LEARN PRIMER LINKS HELP
Cheat Sheet
Primitives
Use the language bar above to insert glyphs. Click on a glyph below to print basic usage and how to type it. Enter]help followed by a symbol for details; for example]help +.
Mathematics
Logic and Comparison
Structural
Selection and Set Operations
Search and Ordering
Operators
Miscellaneous
Additional features
The below subset of Dyalog's functionality is also supported in TryAPL. Enter for example]help]display for details:
TryAPL Version 3.4.5 (enter]State for details)
Mon May 03 2021 11:50:27
Copyright (c) Dyalog Limited 1982-2021
Avg←{(+⌿⍵)÷≢⍵}
Avg 1 5 2 4 3
3
Epsilon (∈)
Monadic function: Enlist
mat
1 2 3
4 5 6
∈ 0 mat (7 8) 9
0 1 2 3 4 5 6 7 8 9
∈ 2 3⍴1 'abc'
1 abc 1 abc 1 abc
Dyadic function: Membership
'abc' 4 ∈ 4 'ab' 'abcd'
0 1
mat ∈ 6 2 7 4
0 1 0
1 0 1
Tab: e e <tab>
Prefix: <prefix> e


TryAPL
tryapl.org
Apps Git Flying & Sailing Dyalog Cloud Exercises SBO Travel Linux Sport APL Productivity
Reading list
FILE INTRO LEARN PRIMER LINKS HELP
Useful links
APL Wiki
Most information is available on the APL Wiki, for example:
Learning resources
How to run APL
Online documentation
Getting help
Having trouble? You might find your question already answered on:
APLcart
Stack Overflow
If that didn't do it, just get in touch with the welcoming community:
Chat in the APL Orchard, a very active chat room
Ask a question on Stack Overflow
Post on the Dyalog Forums
Ask on the r/apljk subreddit
Dyalog social media: Twitter, Facebook, LinkedIn
Recommended videos
There are lots of good APL videos out there, including these hand-picked masterpieces:
Conway's Game of Life in APL
Palindromic Expression for Phi in APL
A Sudoku Solver in APL
APL demonstration 1975
Why APL Is Still Cool (at QCon 2011)
Design Patterns vs Anti pattern in APL (at FnConf17)
Pragmatic Functional Programming with Dyalog's CTO (Talks at Google, 2015)
Dyalog TV (webinars and user meetings)
TryAPL Version 3.4.5 (enter]State for details)
Mon May 03 2021 11:50:27
Copyright (c) Dyalog Limited 1982-2021
Avg←{(+⌿⍵)÷≢⍵}
Avg 1 5 2 4 3
3
Epsilon (∈)
Monadic function: Enlist
mat
1 2 3
4 5 6
∈ 0 mat (7 8) 9
0 1 2 3 4 5 6 7 8 9
∈ 2 3⍴1 'abc'
1 abc 1 abc 1 abc
Dyadic function: Membership
'abc' 4 ∈ 4 'ab' 'abcd'
0 1
mat ∈ 6 2 7 4
0 1 0
1 0 1
Tab: e e <tab>
Prefix: <prefix> e

youtube.com/results?search_query=dyalog+apl

YouTube DK

dyalog apl

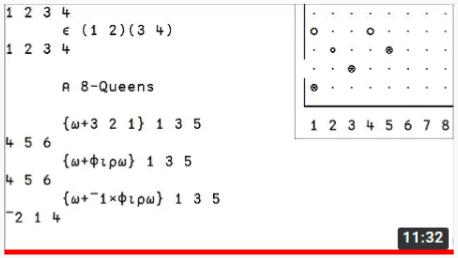
Home
Explore
Subscriptions
Library
History
Watch later
Liked videos

SUBSCRIPTIONS
CNN
The Majority Repor...
Palett
Browse channels

MORE FROM YOUTUBE
YouTube Premium
Movies
Gaming

FILTERS

Dyalog Modern Programming Language | Morten Kromberg | Talks at Google
12K views • 5 years ago
Talks at Google
APL is a member of the family of languages that are approaching middle age (Ken Iverson's book titled "A Programming ...
CC



Depth-first search in APL
6.9K views • 6 years ago
DyalogLtd
Step-by-step development of a depth-first tree-search operator in an APL session. NB: The concepts in this little video are fairly ...

Dyalog Webinars: Fast APL
272 views • Streamed 1 year ago
DyalogLtd
Richard Park lays out some basic guidance of fast (or rather 'not slow') APL code. He presents an overview of topics related to ...

The APL Competition and APL Seeds

- https://www.dyalog.com/student-competition.htm
- https://www.dyalog.com/apl-seeds-user-meetings/aplseeds21.htm

What you can do

- Use those resources to educate new APL'ers
- Admittedly it is all in English, but most young people are well versed in English
- Point young acquaintances of yours to the APL competition – you can even win a price if they do ☺