



DYALOG

Elsinore 2019

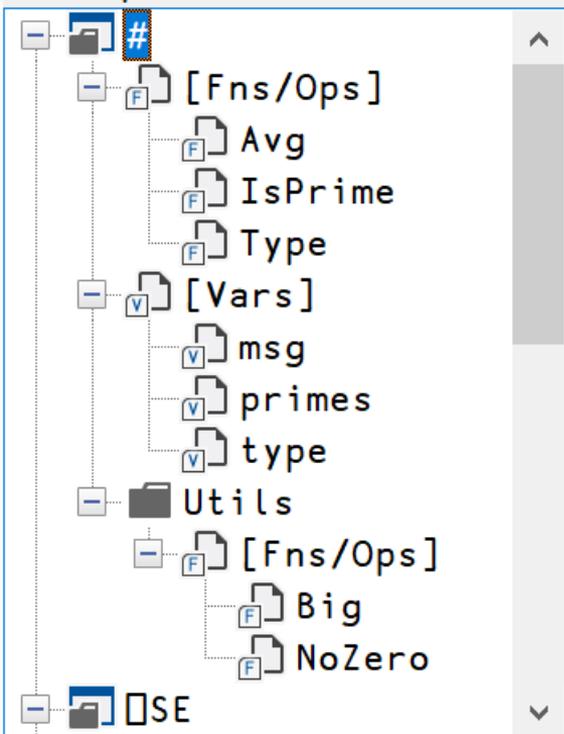
**Source  
Code  
Management  
with Git, SVN & Dyalog APL**

*Morten Kromberg  
Adám Brudzewsky*

File Edit Options View Tools



Workspace Tree



Contents of # (Global Scope)

Name	Location	Type	Description	Size
Avg	#	Function		976b
IsPrime	#	Function	r←IsPrime n	888b
Type	#	Function		904b
Utils	#	Namespace	Namespace	4.258Kb
msg	#	Variable	Matrix: 2x2 nested	232b
primes	#	Variable	Vector: 8 int	112b
type	#	Variable	Vector: 3 nested	176b

File Edit Options View Tools



Workspace Tree

- #
  - [Fns/Ops]
    - Avg
    - IsPrime
    - Type
  - [Vars]
    - msg
    - primes
    - type
  - Utils
    - [Fns/Ops]
      - Big
      - NoZero

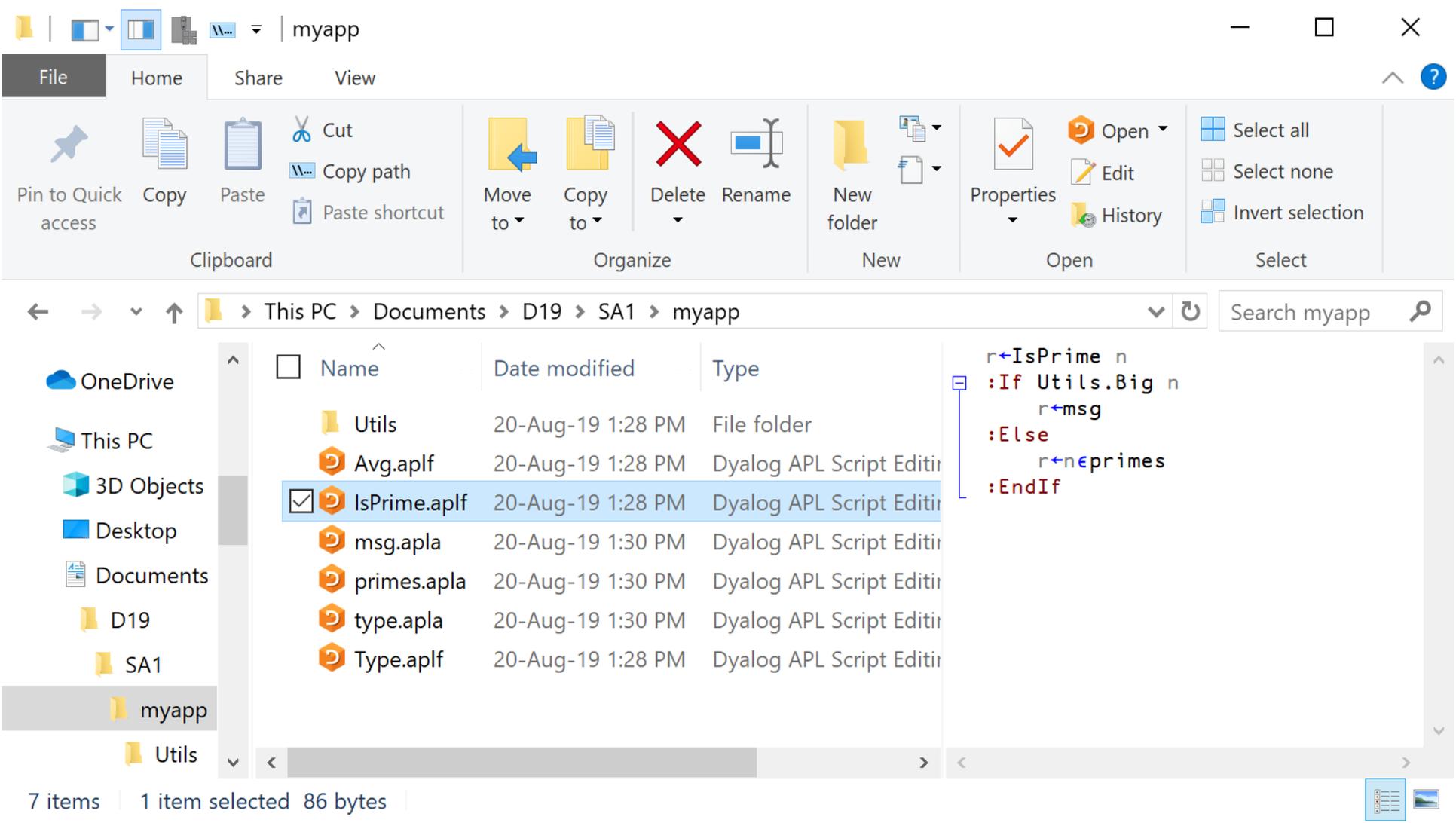
Contents of # (Global Scope)

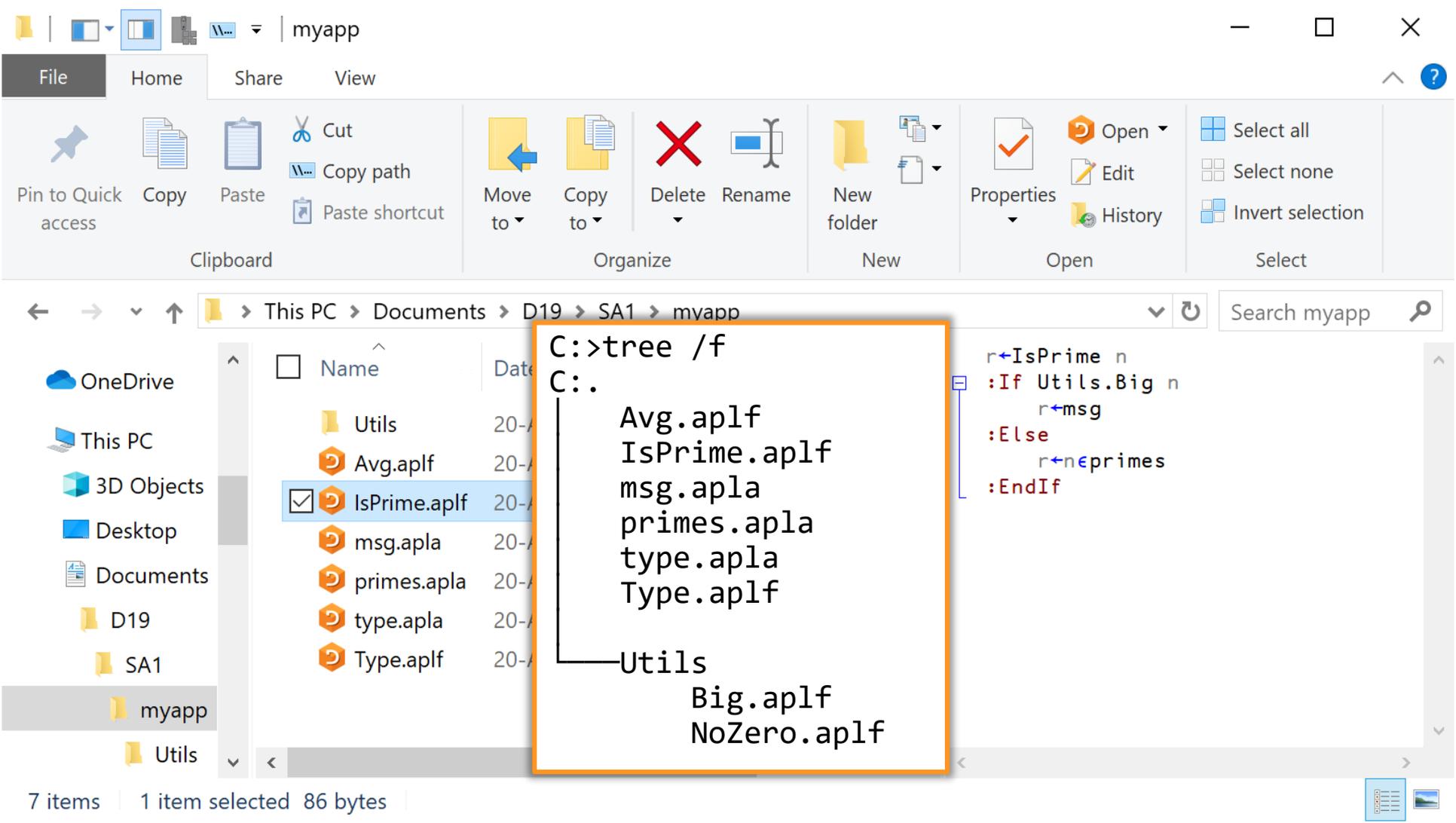
Name	Location	Type
Avg	#	Function
IsPrime	#	Function
Type	#	Function
Utils	#	Namespace
msg	#	Variable
primes	#	Variable
type	#	Variable

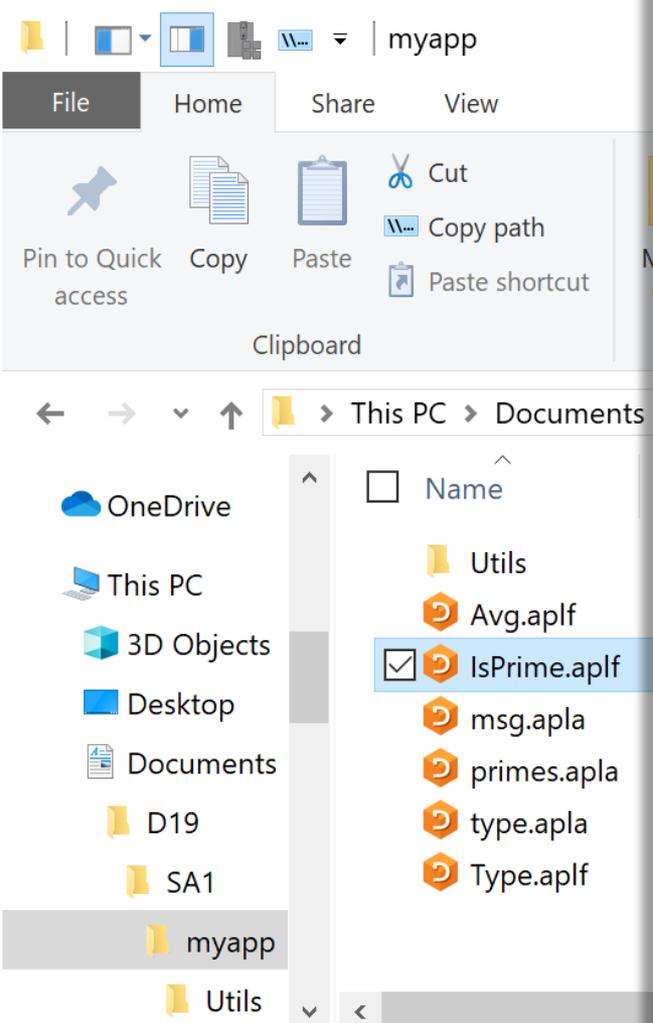
]Map

```
#
~ msg primes type
  ▾ Avg IsPrime Type
    Utils
      ▾ Big NoZero
```

Namespace	Size
Utils	4.258Kb
msg	Matrix: 2x2 nested 232b
primes	Vector: 8 int 112b
type	Vector: 3 nested 176b







### CLEAR WS Exploring [#]

File Edit Options View Tools

#### Workspace Tree

- #
  - [Fns/Ops]
    - Avg
    - IsPrime
    - Type
  - [Vars]
    - msg
    - primes
    - type
  - Utils
    - [Fns/Ops]
      - Big
      - NoZero

#### Contents of # (Global Scope)

Name	Location	Type
Avg	#	Function
IsPrime	#	Function
Type	#	Function
Utils	#	Namespace
msg	#	Variable
primes	#	Variable
type	#	Variable

7 items | 1 item selected | 86 bytes

7 object(s). 1.998Gb free | 7648 bytes used (0 bytes selected)

```
C:>tree /f
C:..
  Avg.aplf
  IsPrime.aplf
  msg.apla
  primes.apla
  type.apla
  Type.aplf
  Uutils
    Big.aplf
    NoZero.aplf
```

CLEAR WS Exploring [#]  
File Edit Options View Tools



Workspace Tree

- #
  - [Fns/Ops]
    - Avg
    - IsPrime
    - Type
  - [Vars]
    - msg
    - primes
    - type
  - Utils
    - [Fns/Ops]
      - Big
      - NoZero

Contents of # (Global Scope)

Name	Location	Type
Avg	#	Function
IsPrime	#	Function
Type	#	Function
Utils	#	Namespace
msg	#	Variable
primes	#	Variable
type	#	Variable

```
]Map
#
· ~ msg primes type
· ∇ Avg IsPrime Type
· Utils
· · ∇ Big NoZero
```

# ]LINK

]link -?

LINK	User commands for namespace-directory synchronisation (see <a href="https://github.com/dyalog/link/w">https://github.com/dyalog/link/w</a> )
Add	Associate item in linked namespace with new file/directory in corresponding directory
Break	Break link between namespace and corresponding directory
Create	Link a namespace with a directory (create one or both if absent)
Export	Export a namespace to a directory (create the directory if absent); does not create a link
Expunge	Erase item and associated file
GetFileName	Return name of file associated with item
GetItemName	Return name of item associated with file
Import	Import a namespace from a directory (create the namespace if absent); does not create a link
List	List active namespace-directory links
Refresh	Fully synchronise namespace-directory content



# ]LINK.Create

```
]LINK.Create #.namespace /path
```

```
Linked: #.namespace ↔ /path
```



# ]LINK.Create

```
]LINK.Create #.namespace /path
```

```
Linked: #.namespace ↔ /path
```

```
3 □MKDIR '/tmp/myapp/'
```

```
'dup←{ω ω}' □NPUT '/tmp/myapp/dup.aplf'
```

```
]link.create myapp /tmp/myapp
```

```
Linked: #.myapp ↔ /tmp/myapp
```



# ]LINK.Add

```
]LINK.Add item
```

```
Added: #.item
```



# ]LINK.Add

```
]LINK.Add item
```

```
Added: #.item
```

```
myapp.myarray←ι2 3
```

```
]link.add myapp.myarray
```



# ]LINK.Break

```
]LINK.Break #.namespace
```

```
Unlinked N items: #.namespace
```



# ]LINK.Break

```
]LINK.Break #.namespace
```

```
Unlinked N items: #.namespace
```

```
]link.break myapp
```



# ]LINK.Break

```
]LINK.Break #.namespace
```

```
Unlinked N items: #.namespace
```

```
]link.break myapp
```

```
]link.create myapp /tmp/newapp
```



# ]LINK.Break

```
]LINK.Break #.namespace
```

```
Unlinked N items: #.namespace
```

```
]link.break myapp
```

```
]link.create myapp /tmp/newapp
```

```
Source directory not found: C:\tmp\newapp
```



**Link wiki: [github.com/Dyalog/link/wiki](https://github.com/Dyalog/link/wiki)**



# Link wiki: [github.com/Dyalog/link/wiki](https://github.com/Dyalog/link/wiki)

## Home

Adám Brudzewsky edited this page 7 days ago · 6 revisions

*Link* facilitates the use of text files for APL source code through the creation of one or more *links* between *namespaces* inside the active workspace and *directories* containing source code. Functionality provided by links include:

- **Keeping Source Files Up-to-date:** Typically, links are configured to replicate any changes made using the Dyalog editor and tracer to external source files. As a result, the source files are kept up-to-date without further action by the developer.
- **Integrating External Changes into the workspace:** Links can be also be configured to use a "file system watcher" to replicate changes made

► Pages 18

1. [Home](#)
2. [Overview](#)
3. [API](#)
  - i. [Add](#)
  - ii. [Break](#)
  - iii. [CaseCode](#)
  - iv. [Create](#)
  - v. [Export](#)



# Link wiki: [github.com/Dyalog/link/wiki](https://github.com/Dyalog/link/wiki)

## Home

Adám Brudzewsky edited this page 7 days ago · 6 revisions

*Link* facilitates the use of text files for APL source code through the creation of one or more *links* between *namespaces* inside the active workspace and *directories* containing source code. Functionality provided by links include:

- **Keeping Source Files Up-to-date:** Typically, links are configured to replicate any changes made using the Dyalog editor and tracer to external source files. As a result, the source files are kept up-to-date without further action by the developer.
- **Integrating External Changes into the workspace:** Links can be also be configured to use a "file system watcher" to replicate changes made

► Pages 18

1. [Home](#)
2. [Overview](#)
3. [API](#)
  - i. [Add](#)
  - ii. [Break](#)
  - iii. [CaseCode](#)
  - iv. [Create](#)
  - v. [Export](#)



# Link wiki: [github.com/Dyalog/link/wiki](https://github.com/Dyalog/link/wiki)

```
]LINK.Create #.ns /path/ns  
□SE.Link.Create '#/ns' '/path/ns'
```

*Link* facilitates the use of text files for APL source code through the creation of one or more *links* between *namespaces* inside the active workspace and *directories* containing source code. Functionality provided by links include:

- **Keeping Source Files Up-to-date:** Typically, links are configured to replicate any changes made using the Dyalog editor and tracer to external source files. As a result, the source files are kept up-to-date without further action by the developer.
- **Integrating External Changes into the workspace:** Links can be also be configured to use a "file system watcher" to replicate changes made

► Pages 18

1. [Home](#)
2. [Overview](#)
3. [API](#)
  - i. [Add](#)
  - ii. [Break](#)
  - iii. [CaseCode](#)
  - iv. [Create](#)
  - v. [Export](#)



# Link wiki: [github.com/Dyalog/link/wiki](https://github.com/Dyalog/link/wiki)

```
]LINK.Create #.ns /path/ns
```

```
[]SE.Link.Create '#/ns' '/path/ns'
```

```
[]SE.UCMD 'LINK.Create #.ns /path/ns'
```

of one or more *links* between *namespaces* inside the active workspace and *directories* containing source code. Functionality provided by links include:

- **Keeping Source Files Up-to-date:** Typically, links are configured to replicate any changes made using the Dyalog editor and tracer to external source files. As a result, the source files are kept up-to-date without further action by the developer.
- **Integrating External Changes into the workspace:** Links can be also be configured to use a "file system watcher" to replicate changes made

1. [Home](#)
2. [Overview](#)
3. [API](#)
  - i. [Add](#)
  - ii. [Break](#)
  - iii. [CaseCode](#)
  - iv. [Create](#)
  - v. [Export](#)



# ]LINK.Create options

```
]LINK.Create #.namespace /path -options
```

```
]link.break myapp
```

```
]link.create myapp /tmp/newapp
```

```
Source directory not found: C:\tmp\newapp
```



# ]LINK.Create options

```
]LINK.Create #.namespace /path -options
```

```
]link.break myapp
```

```
]link.create myapp /tmp/newapp -source=ns
```

```
Linked: #.myapp ↔ /tmp/newapp
```



# ]LINK.Import

```
]LINK.Import #.namespace /path
```

```
Imported: #.namespace ← /path
```



# ]LINK.Import

```
]LINK.Import #.namespace /path
```

```
Imported: #.namespace ← /path
```

```
]link.import myapp /tmp/myapp
```



# ]LINK.List

```
]LINK.List
```

```
]LINK.List #.namespace
```



# ]LINK.List

```
]LINK.List
```

```
]LINK.List #.namespace
```

```
]link.list
```



# ]LINK.List

```
]LINK.List
```

```
]LINK.List #.namespace
```

```
]link.list
```

No active links



# ]LINK.List

```
]LINK.List
```

```
]LINK.List #.namespace
```

```
]link.list
```

Namespace	Directory	Items
#.test	C:/tmp/test	7





DYALOG

Elsinore 2019

**S**ource  
**C**ode  
**M**anagement  
with **Git[Hub]** & **Dyalog APL**

*Morten Kromberg*  
*Adám Brudzewsky*

# SA1 – Part 2: Git (and GitHub)



## SA1 – Part 2: Git (and GitHub)

- One of the most important benefits of storing APL code in text files is that standard **Source Code Managment** systems apply to our source



## SA1 – Part 2: Git (and GitHub)

- One of the most important benefits of storing APL code in text files is that standard **Source Code Managment** systems apply to our source
- Git** is the most popular **SCM** system today



## SA1 – Part 2: Git (and GitHub)

- One of the most important benefits of storing APL code in text files is that standard **Source Code Managment** systems apply to our source
- Git** is the most popular **SCM** system today
- GitHub** is a service that hosts Git repositories and provides a web front end



# Git vs other SCM systems

- All modern SCMs use a strategy of only storing changes each time a file is changed
- Git is a "distributed" SCM, different from "client-server" SCMs like Subversion:
  - Each user of a repository has a complete local copy of all changes ever made to the repository
  - (rather than a snapshot of the files at a point in time)
- This allows disconnected/remote use
- If more than one user modifies a file, files are "merged" if Git can figure out how to do that, which usually means:
  - The source files consists of lines of text
  - No two users modified the same line of text



# Git vs GitHub



# Git vs GitHub

- Git is a source code management system that works on files
  - Open source, invented by Linus Thorvalds (of Linux fame)



# Git vs GitHub

- Git is a source code management system that works on files
  - Open source, invented by Linus Thorvalds (of Linux fame)
- GitHub is a web server/service that ...
  - Hosts Git repositories
  - Provides a web front end for them
  - Allows local Git clients to synchronise cloud and local repos using "pull" and "push" commands
  - Manages Pull Requests (changes suggested by 3<sup>rd</sup> parties)
  - The largest collection of source code in the world
  - Now owned and run by Microsoft
  - Free: unlimited private repos and limited private repos (pay for more)



# Our Workspace

CLEAR WS Exploring [#]

File Edit Options View Tools

Workspace Tree

Contents of # (Global Scope)

Name	Location	Type	Description	Size
Avg	#	Function		976b
IsPrime	#	Function	r←IsPrime n	888b
Type	#	Function		904b
Utils	#	Namespace	Namespace	4.258Kb
msg	#	Variable	Matrix: 2x2 nested	232b
primes	#	Variable	Vector: 8 int	112b
type	#	Variable	Vector: 3 nested	176b

Workspace Tree

- [#]
  - [Fns/Ops]
    - Avg
    - IsPrime
    - Type
  - [Vars]
    - msg
    - primes
    - type
  - Utils
    - [Fns/Ops]



# Our Workspace

CLEAR WS Exploring [#]

File Edit Options View Tools

Workspace Tree

```

graph TD
    Root[Workspace Tree] --> FnsOps1[Fns/Ops]
    Root --> Vars[Vars]
    Root --> Utils1[Utils]
    FnsOps1 --> Avg1[Avg]
    FnsOps1 --> IsPrime1[IsPrime]
    FnsOps1 --> Type1[Type]
    Vars --> msg[msg]
    Vars --> primes[primes]
    Vars --> type[type]
    Utils1 --> FnsOps2[Fns/Ops]
    
```

Contents of

Name
Avg
IsPrime
Type
Utils
msg
primes
type

myapp

File Home Share View

Clipboard: Pin to Quick access, Copy, Paste, Cut, Copy path, Paste shortcut

Organize: Move to, Copy to, Delete, Rename

This PC > Documents > D19 > SA1 > myapp

Name	Date modified	Type
Utils	20-Aug-19 1:28 PM	File folder
Avg.aplf	20-Aug-19 1:28 PM	Dyalog APL
<input checked="" type="checkbox"/> IsPrime.aplf	20-Aug-19 1:28 PM	Dyalog APL
msg.apla	20-Aug-19 1:30 PM	Dyalog APL
primes.apla	20-Aug-19 1:30 PM	Dyalog APL
type.apla	20-Aug-19 1:30 PM	Dyalog APL
Type.aplf	20-Aug-19 1:28 PM	Dyalog APL



# Exercise 1



# Exercise 1

1. Save myapp.dws to a folder called "myapp" using `]link.create`



# Exercise 1

1. Save myapp.dws to a folder called "myapp" using `]link.create`
2. Move the `Utils` folder out of myapp, so that the two folders are siblings.



# Exercise 1

1. Save myapp.dws to a folder called "myapp" using `]link.create`
2. Move the `Utils` folder out of myapp, so that the two folders are siblings.
3. In a clear workspace, use `]link.create` to recreate the original workspace structure.



# Exercise 1

1. Save `myapp.dws` to a folder called "myapp" using `]link.create`
2. Move the `Utils` folder out of `myapp`, so that the two folders are siblings.
3. In a clear workspace, use `]link.create` to recreate the original workspace structure.
4. Write an APL function to call `⊞SE.Link.Create`, in place of using user commands



# Exercise 1 - Solutions



# Exercise 1 - Solutions

```
1. )load myapp  
   ]link.create # c:\wherever\myapp -source=ns
```



# Exercise 1 - Solutions

1. 

```
)load myapp  
]link.create # c:\wherever\myapp -source=ns
```
2. Use File Explorer to move Utils



# Exercise 1 - Solutions

1. 

```
)load myapp  
]link.create # c:\whatever\myapp -source=ns
```
2. Use File Explorer to move Utils
3. 

```
]link.create Utils c:\whatever\Utils  
]link.create # c:\whatever\myapp
```



# Exercise 1 - Solutions

1. 

```
)load myapp  
]link.create # c:\wherever\myapp -source=ns
```
2. Use File Explorer to move Utils
3. 

```
]link.create Utils c:\wherever\Utils  
]link.create # c:\wherever\myapp
```
4. 

```
▽ Load;folder;opts  
  (opts←[]NS ' ').source←'dir'  
  folder←'c:\wherever\  
  opts []SE.Link.Create 'Utils' (folder,'\Utils')  
  opts []SE.Link.Create # (folder,'\myapp')
```



# git version and git config

- Try these commands, in order to...
  - Check Git is installed
    - (if not go sit next to someone who has it)
  - Register your name & email (not mine!)
  - Test the Git help system

```
C:\ git --version
git version 2.14.1.windows.1

C:\ git config --global user.name "Morten Kromberg"
C:\ git config --global user.email mkrom@dyalog.com
C:\ git config -list
C:\ git --help
C:\ git config --help
```



# git init

`git init` turns a directory into a Git repository

- It does this by creating a hidden directory named `.git` within your directory
- To make it a normal repository again, simply delete the `.git` folder

Exercise 2:

- Use `git init` to turn your `myapp` directory into a git repository



```
Command Prompt
C:\D19\SA1>
C:\D19\SA1>CD C:\D19\SA1\myapp
C:\D19\SA1\myapp>git version
git version 2.14.1.windows.1
C:\D19\SA1\myapp>git status
fatal: Not a git repository (or any of the parent directories): .git
C:\D19\SA1\myapp>git init
Initialized empty Git repository in C:/D19/SA1/myapp/.git/
C:\D19\SA1\myapp>dir .git
Volume in drive C is Windows
Volume Serial Number is 624D-7B40

Directory of C:\D19\SA1\myapp\.git

01-09-2019  21:16                130 config
01-09-2019  21:16                73 description
01-09-2019  21:16                23 HEAD
01-09-2019  21:16      <DIR>        hooks
01-09-2019  21:16      <DIR>        info
01-09-2019  21:16      <DIR>        objects
01-09-2019  21:16      <DIR>        refs
                3 File(s)            226 bytes
                4 Dir(s)  253.297.889.280 bytes free
C:\D19\SA1\myapp>
```

# git status

`git status` gives a brief report about the current state of your repository

- ◆ The current branch (more about branches soon)
- ◆ Changes made to tracked files since the last commit
- ◆ Any *untracked files* in the directory

## Exercise 3

- ◆ Try `git status` on your new repo



```
C:\D19\SA1\myapp>git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
Avg.aplf
```

```
IsPrime.aplf
```

```
Type.aplf
```

```
msg.apla
```

```
primes.apla
```

```
type.apla
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
C:\D19\SA1\myapp>
```

# git add (and git rm)

`git add <file>` *stages* a file, so that it will be in the next commit.

- New / Changed files are NOT staged by default
  - (many GUI frontends will offer to do this for you)
- `git add .` will stage all files in the current directory
- `git status` will show you what you have done
- You can change files and repeatedly use `git add` to stage multiple changes

## Exercise 4

- Use `git add` at least twice, to stage first one, then all the files in `myapp`
- Monitor your progress using `git status`
- Try `git add --help`, and `git rm --help`



```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>git add Avg.aplf
```

```
C:\D19\SA1\myapp>git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   Avg.aplf
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
IsPrime.aplf
```

```
Type.aplf
```

```
msg.apla
```

```
primes.apla
```

```
type.apla
```

```
C:\D19\SA1\myapp>_
```

```
C:\D19\SA1\myapp>git add .
```

```
C:\D19\SA1\myapp>git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   Avg.aplf
new file:   IsPrime.aplf
new file:   Type.aplf
new file:   msg.apla
new file:   primes.apla
new file:   type.apla
```

```
C:\D19\SA1\myapp>
```

# git commit -m "commit message"

Eventually, you will want to use `git commit` to integrate a set of staged changes to the repository.

- You MUST provide a commit message using the `-m` switch as in the title of this slide.
- Use meaningful commit messages, you will thank yourself later. "First commit" *is* a meaningful message (this time)!

## Exercise 5

- Use `git commit` to make the first commit
- Return to APL and edit one of the functions in `#`
- Use `git status`, `git add`, `git commit` to do the right thing





```
C:\D19\SA1\myapp>git commit -m "First commit"
```

```
[master (root-commit) 5e1211e] First commit
```

```
6 files changed, 20 insertions(+)
```

```
create mode 100644 Avg.aplf
```

```
create mode 100644 IsPrime.aplf
```

```
create mode 100644 Type.aplf
```

```
create mode 100644 msg.apla
```

```
create mode 100644 primes.apla
```

```
create mode 100644 type.apla
```

```
C:\D19\SA1\myapp>git log
```

```
commit 5e1211e2aa5ea0401c30b100e6f85afa84c686fd (HEAD -> master)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:41:13 2019 +0200
```

```
First commit
```

```
C:\D19\SA1\myapp>
```

```
Command Prompt
C:\D19\SA1\myapp>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Avg.aplf

no changes added to commit (use "git add" and/or "git commit -a")

C:\D19\SA1\myapp>git add .

C:\D19\SA1\myapp>git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Avg.aplf

C:\D19\SA1\myapp>git commit -m "Added comment to Avg"
[master 118d87e] Added comment to Avg
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\D19\SA1\myapp>
```

# git log

`git log` allows you to view the commit log

- You can see the last n entries, filter by date/time, author, and many other criteria
- I found [www.thegeekstuff.com/2014/04/git-log](http://www.thegeekstuff.com/2014/04/git-log)

## Exercise 6

- Use `git log` to examine the log and filter it in various ways.
- Use `git log --stat` and `git log -p` to look at the actual changes (aka the "diffs")



```
C:\D19\SA1\myapp>git log
```

```
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:54:11 2019 +0200
```

```
Added comment to Avg
```

```
commit 5e1211e2aa5ea0401c30b100e6f85afa84c686fd
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:41:13 2019 +0200
```

```
First commit
```

```
C:\D19\SA1\myapp>git log --after "2019-09-01 21:45"
```

```
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:54:11 2019 +0200
```

```
Added comment to Avg
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>git log --after "2019-09-01 21:45" --stat
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:54:11 2019 +0200
```

```
Added comment to Avg
```

```
Avg.aplf | 2 +-  
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
C:\D19\SA1\myapp>git log --after "2019-09-01 21:45" -p
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Sun Sep 1 21:54:11 2019 +0200
```

```
Added comment to Avg
```

```
diff --git a/Avg.aplf b/Avg.aplf
```

```
index 888e602..b4e45ce 100644
```

```
--- a/Avg.aplf
```

```
+++ b/Avg.aplf
```

```
@@ -1,4 +1,4 @@
```

```
- Avg{
```

```
+ Avg{ Compute the average of
```

```
good Utils.NoZero
```

```
sum + good
```

```
C:\D19\SA1\myapp>git log --after "2019-09-01 21:45" --stat
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
Author: Morten Kromberg <mkrom@dyalog.com>
Date: Sun Sep 1 21:54:11 2019 +0200
```

Added comment to Avg

```
Avg.aplf | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
C:\D19\SA1\myapp>git log --after "2019-09-01 21:45" -p
commit 118d87e56d518922aae209084d025bf1aa6c3d46 (HEAD -> master)
Author: Morten Kromberg <mkrom@dyalog.com>
Date: Sun Sep 1 21:54:11 2019 +0200
```

Added comment to Avg

```
diff --git a/Avg.aplf b/Avg.aplf
index 888e602..b4e45ce 100644
--- a/Avg.aplf
+++ b/Avg.aplf
@@ -1,4 +1,4 @@
```

E2	86	90	]tohex 'UTF-8' []UCS '←'
E2	8D	9D	]tohex 'UTF-8' []UCS '␣'
E2	8D	B5	]tohex 'UTF-8' []UCS 'ω'

```
- Avg<E2><86><90>{
+ Avg<E2><86><90>{ <E2><8D><9D> Compute the average of <E2><8D><B5>
  good<E2><86><90>Utils.NoZero <E2><8D><B5>
  sum<E2><86><90>+<E2><8C><BF>good
```

# Time to Consider GUI FrontEnds?

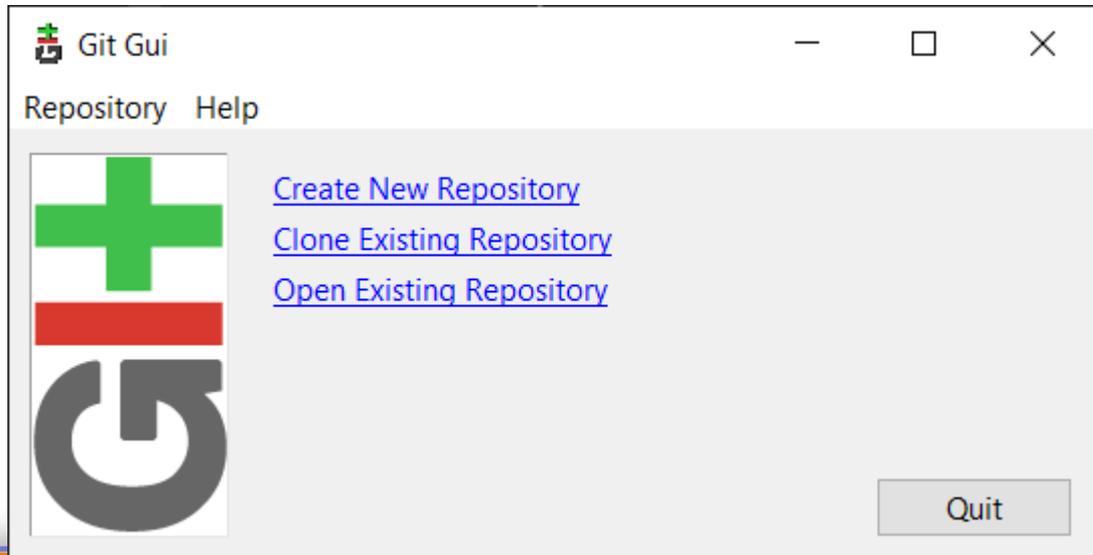
- Git GUI
- VS Code
  
- There are lots of others



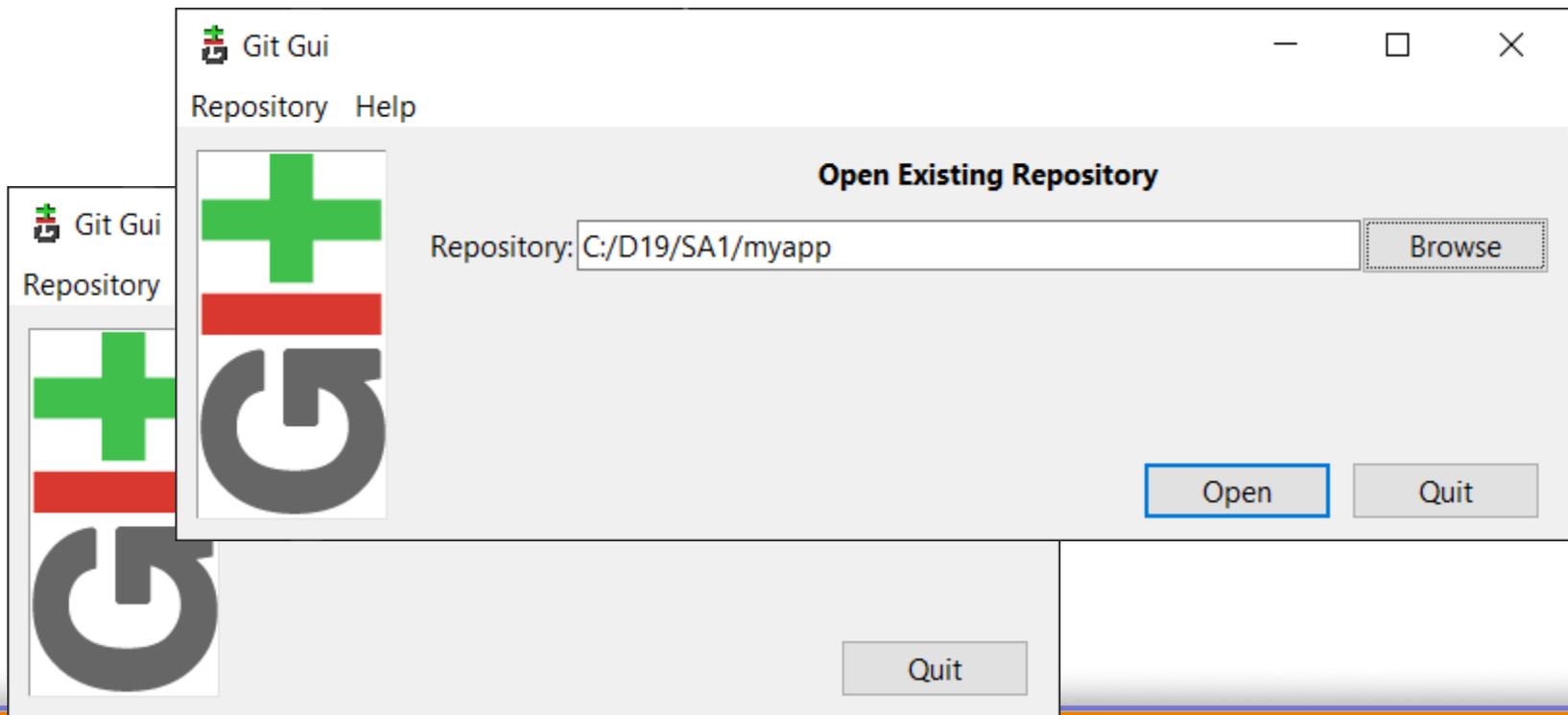
# Git GUI



# Git GUI

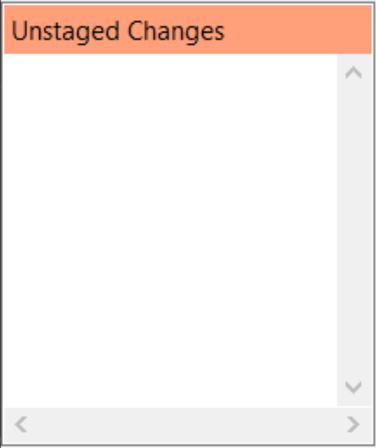
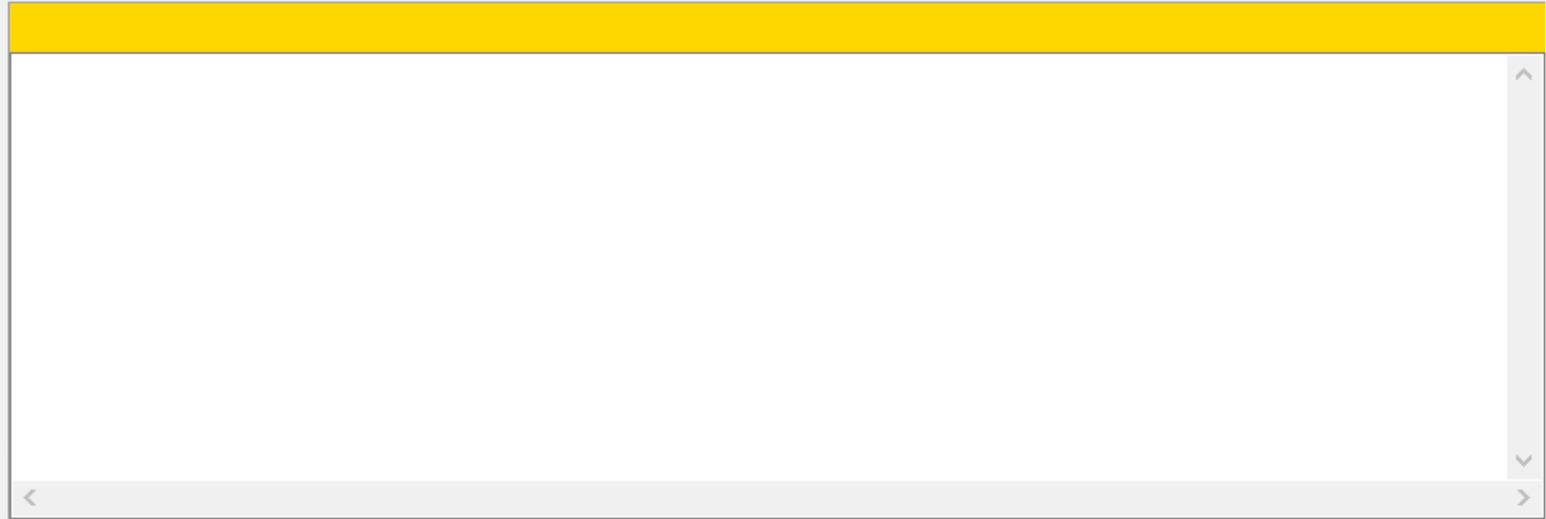


# Git GUI

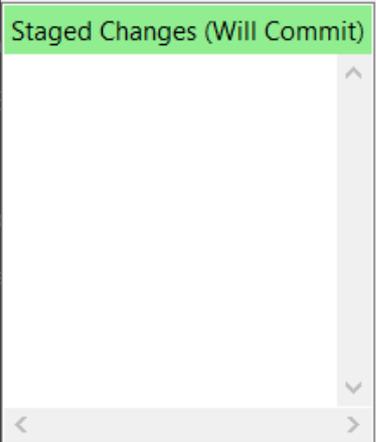


Current Branch: master

Unstaged Changes

A vertical panel with an orange header and a scrollable area. It is currently empty. Navigation arrows are visible at the bottom.A large yellow horizontal bar at the top, followed by a large white scrollable area. This area is currently empty.

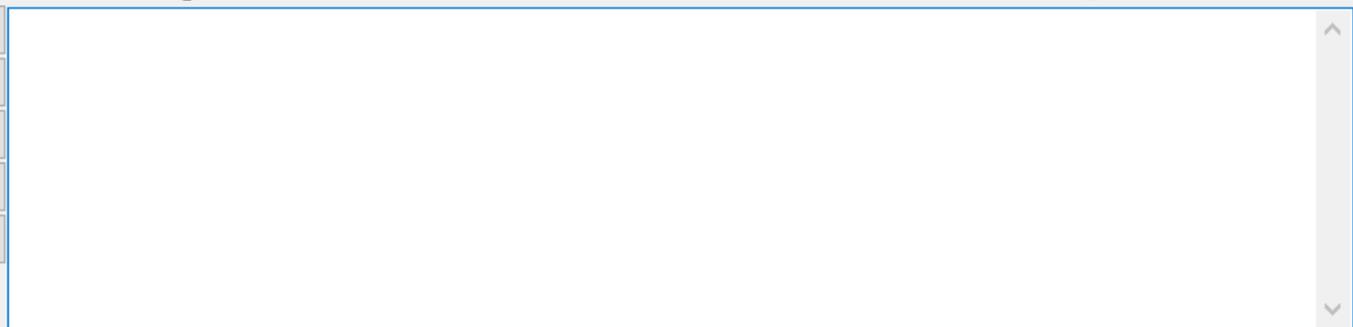
Staged Changes (Will Commit)

A vertical panel with a green header and a scrollable area. It is currently empty. Navigation arrows are visible at the bottom.

Commit Message:

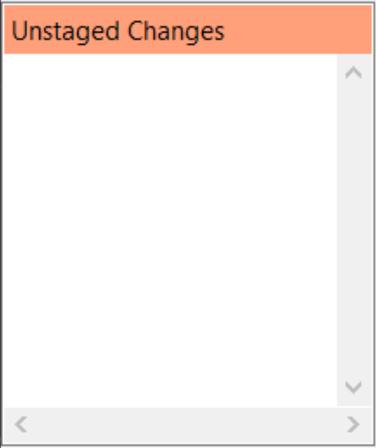
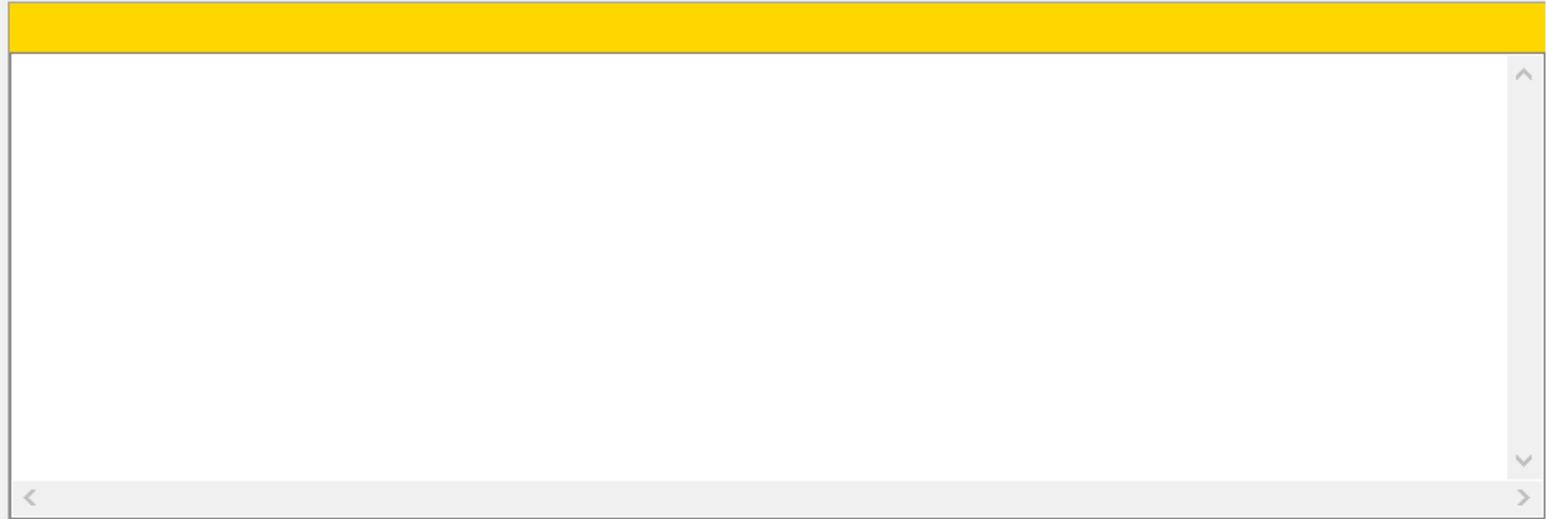
New Commit  Amend Last Commit

- Rescan
- Stage Changed
- Sign Off
- Commit
- Push

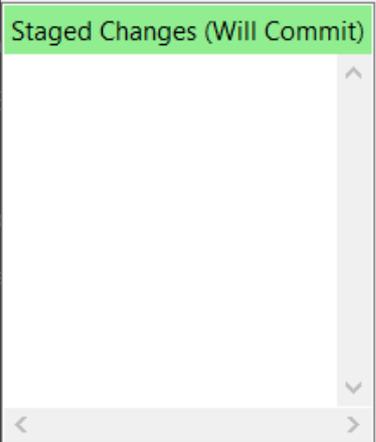
A large white text area with a blue border, intended for entering a commit message. It is currently empty.

Current Branch: master

Unstaged Changes

A vertical panel with an orange header and a scrollable area. It is currently empty. Navigation arrows are visible at the bottom.A large yellow header bar above a large white scrollable area. The scrollable area is currently empty.

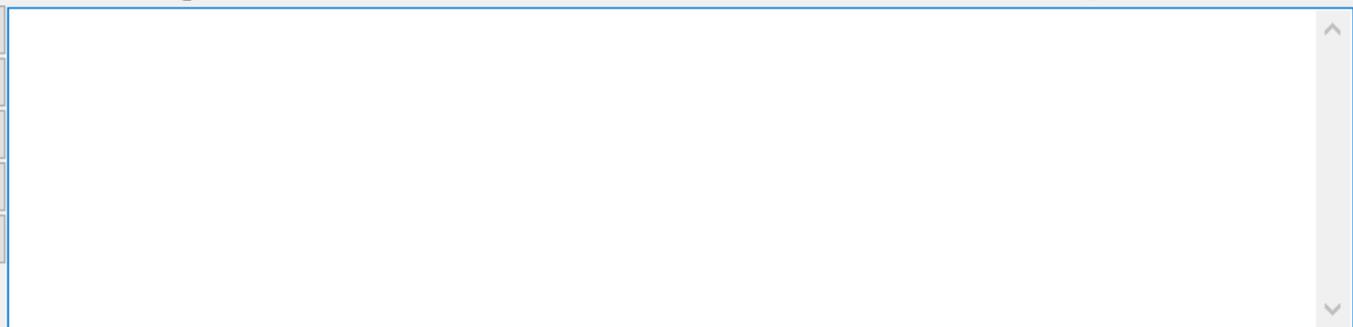
Staged Changes (Will Commit)

A vertical panel with a green header and a scrollable area. It is currently empty. Navigation arrows are visible at the bottom.

Commit Message:

New Commit  Amend Last Commit

- Rescan
- Stage Changed
- Sign Off
- Commit
- Push

A large white text area for entering a commit message. It is currently empty.

Repository Edit Branch Commit Merge Remote Tools Help

Explore Working Copy

Git Bash

Browse master's Files

Browse Branch Files...

Visualize master's History

Visualize All Branch History

Database Statistics

Compress Database

Verify Database

Create Desktop Icon

Quit

Ctrl-Q

Commit Message:

 New Commit  Amend Last Commit

Rescan

Stage Changed

Sign Off

Commit

Push

● **master** Added comment to Avg  
● First commit

Morten Kromberg <mkrom@dy 2019-09-01 21:54:11  
 Morten Kromberg <mkrom@dy 2019-09-01 21:41:13

SHA1 ID: 118d87e56d518922aae209084d025bf1aa6c3d46

Row 1 / 2

Find  commit containing: Exact  All fields Search  Patch  Tree

Diff  Old version  New version Lines of context: 3   
 Author: Morten Kromberg <mkrom@dyalog.com> 2019-09-01  
 Committer: Morten Kromberg <mkrom@dyalog.com> 2019-09-01  
 Parent: [5e1211e2aa5ea0401c30b100e6f85afa84c686fd](#) (First  
 Branch: [master](#)  
 Follows:  
 Precedes:

Added comment to Avg

----- Avg.aplf -----

```

index 888e602..b4e45ce 100644
@@ -1,4 +1,4 @@
- Avg{
+ Avg{
   Compute the average of
   good+Utils.NoZero
   sum+good
   sum- good
  
```

Comments  
 Avg.aplf

● **master** Added comment to Avg  
● First commit

Morten Kromberg <mkrom@dy 2019-09-01 21:54:11  
 Morten Kromberg <mkrom@dy 2019-09-01 21:41:13

SHA1 ID: 118d87e56d518922aae209084d025bf1aa6c3d46

Row 1 / 2

Find  commit containing: Exact  All fields Search  Patch  Tree

Diff  Old version  New version Lines of context: 3   
 Author: Morten Kromberg <mkrom@dyalog.com> 2019-09-01  
 Committer: Morten Kromberg <mkrom@dyalog.com> 2019-09-01  
 Parent: [5e1211e2aa5ea0401c30b100e6f85afa84c686fd](#) (First  
 Branch: [master](#)  
 Follows:  
 Precedes:

Added comment to Avg

----- Avg.aplf -----

index 888e602..b4e45ce 100644

@@ -1,4 +1,4 @@

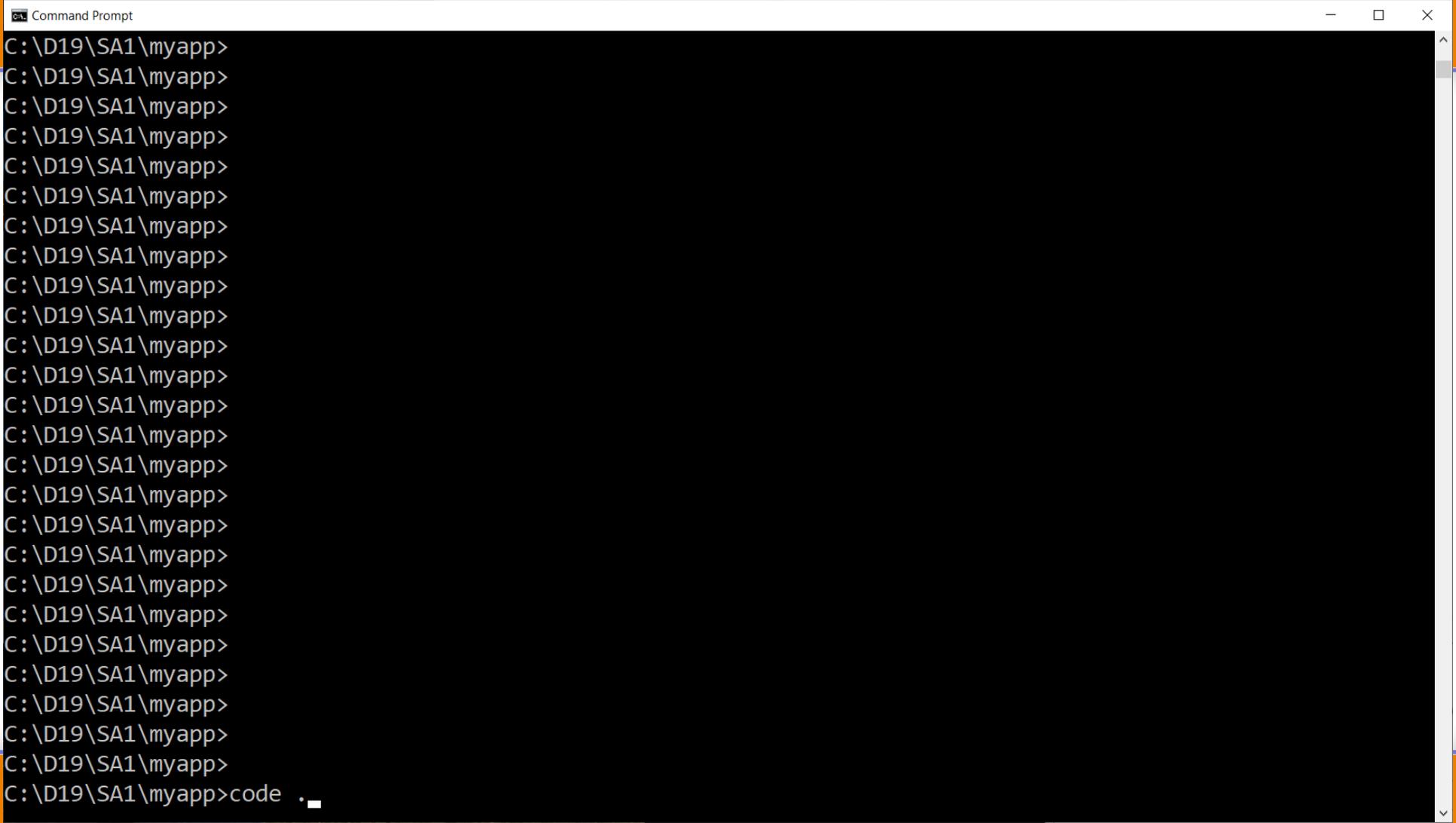
```

- Avg{
+ Avg{
  Compute the average of
  good+Utils.NoZero
  sum+good
  sum*good
  
```

Ugh – "raw" UTF-8

 Comments

Avg.aplf



C:\D19\SA1\myapp>

C:\D19\SA1\myapp>code .



GITLENS

REPOSITORIES

- myapp master
  - master
    - Compare master (working) with <branch, tag, ...
  - Branches
    - master
      - Less than a week ago
      - Added comment to Avg +0 ~1 -0 • You, 17 ...
        - Avg.aplf**
        - First commit +6 ~0 -0 • You, 17 hours ago
      - Contributors
      - Remotes
      - Stashes
      - Tags

FILE HISTORY

LINE HISTORY

COMPARE

SEARCH COMMITS

⌵ Avg.aplf (5e1211e) ↔ Avg.aplf (118d87e) ×

c: > D19 > SA1 > myapp > Avg.aplf

```

1 - Avg+{
2   good+Utils.NoZero ω
3   sum++/good
4   sum÷#good
5 }
6

```

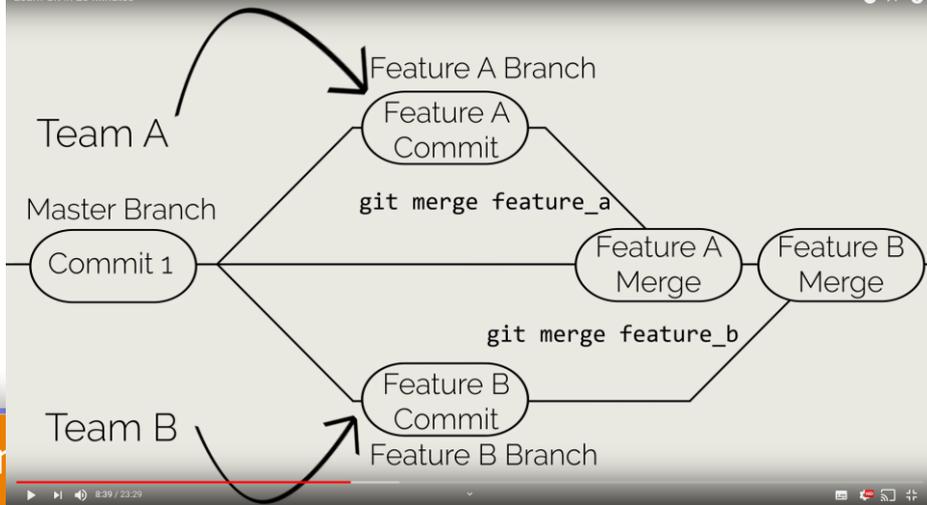
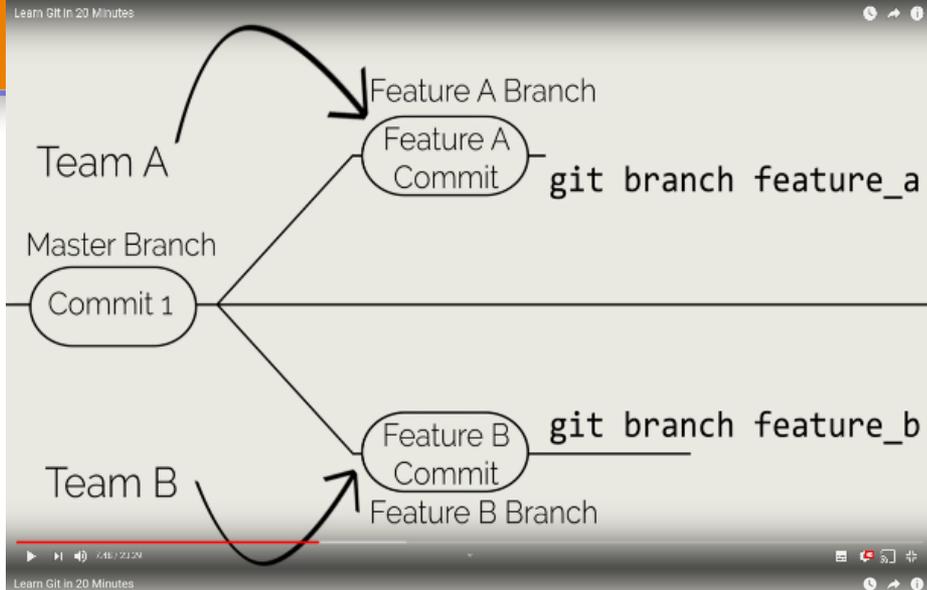
```

1 + Avg+{ A Compute the average of ω
2   good+Utils.NoZero ω
3   sum++/good
4   sum÷#good
5 }
6

```

# Branches

Branches allow you to create environments in which changes can be made and tracked – without interfering with other uses of the code.



# Git Tutorials

- Learn Git in 20 Minutes

[youtu.be/IHaTbJPdB-s](https://youtu.be/IHaTbJPdB-s) by *Web Dev Simplified*

- Git Tutorial for Beginners: Command-Line Fundamentals

[youtu.be/HVsySz-h9r4](https://youtu.be/HVsySz-h9r4) by Corey Schafer



# git branch | git checkout | git merge



# git branch | git checkout | git merge

`git branch branchname` creates a branch, in which you can develop new features or fixes outside the "master"



# git branch | git checkout | git merge

`git branch branchname` creates a branch, in which you can develop new features or fixes outside the "master"

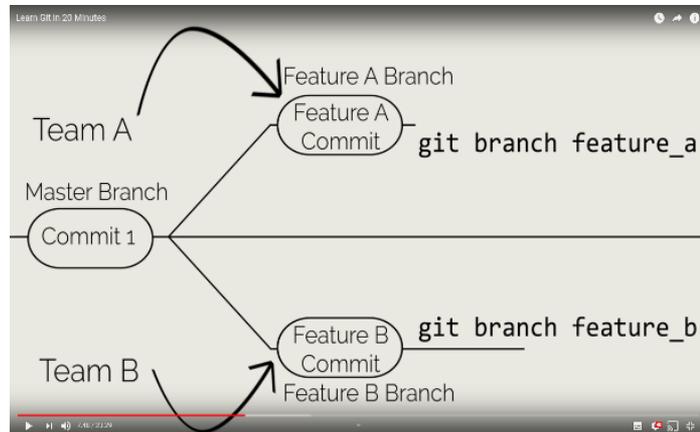
- ◆ `git checkout -b branchname` is shorthand for: `git branch branchname` followed by: `git checkout branchname`



# git branch | git checkout | git merge

`git branch branchname` creates a branch, in which you can develop new features or fixes outside the "master"

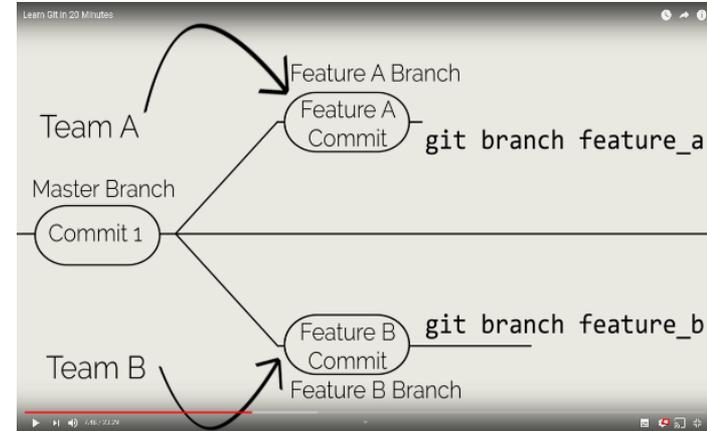
- `git checkout -b branchname` is shorthand for: `git branch branchname` followed by: `git checkout branchname`



# git branch | git checkout | git merge

`git branch branchname` creates a branch, in which you can develop new features or fixes outside the "master"

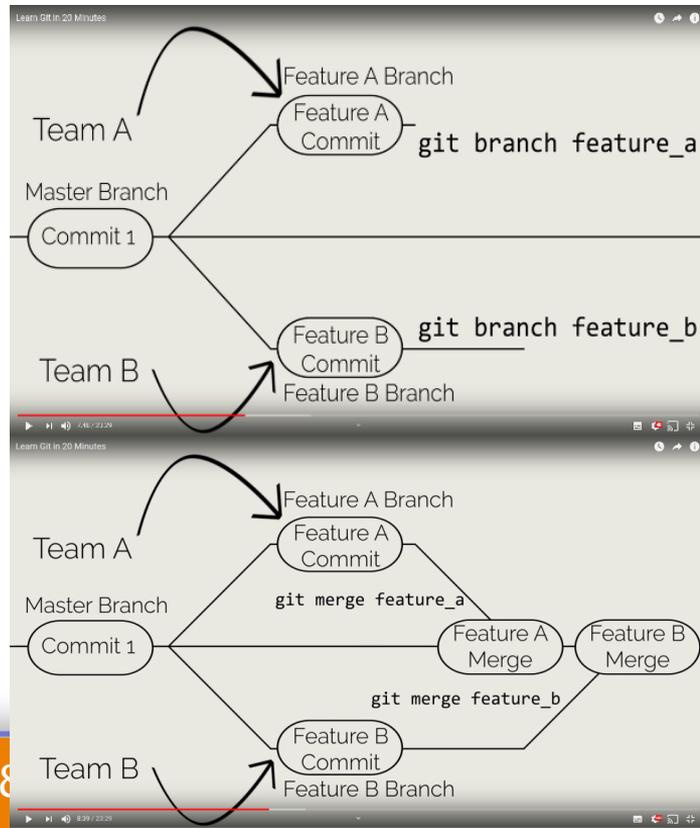
- `git checkout -b branchname` is shorthand for: `git branch branchname` followed by: `git checkout branchname`
- When the new feature is complete, `git merge branchname` is used to fold it back (after switching back to master with `git checkout master`)



# git branch | git checkout | git merge

`git branch branchname` creates a branch, in which you can develop new features or fixes outside the "master"

- `git checkout -b branchname` is shorthand for: `git branch branchname` followed by: `git checkout branchname`
- When the new feature is complete, `git merge branchname` is used to fold it back (after switching back to master with `git checkout master`)



# git branch | git checkout | git merge

- ◆ Before you start the exercise: if you have VS Code or similar, keep it open and watch what happens as you make changes on the command line
- ◆ Also watch what happens inside the workspace as you checkout different branches (assuming you have a link)

## Exercise 6

- ◆ Use `git checkout -b branchname` to create a branch, and make a couple of changes and commits.
- ◆ Merge the commits back into the master.
- ◆ Use `git branch` to verify the selected branch.
- ◆ Delete the branch using `git branch -d branchname`.



```
C:\D19\SA1\myapp>git checkout -b moreprimes
```

```
Switched to a new branch 'moreprimes'
```

```
C:\D19\SA1\myapp>git status
```

```
On branch moreprimes
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   primes.apla
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
C:\D19\SA1\myapp>git add .
```

```
C:\D19\SA1\myapp>git commit -m "Added 23 to list of primes"
```

```
[moreprimes ee5ff8e] Added 23 to list of primes
```

```
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
C:\D19\SA1\myapp>git add .
```

```
C:\D19\SA1\myapp>git commit -m "Fixed comparison ot avoid Utils.Big"
```

```
[moreprimes c7516d7] Fixed comparison ot avoid Utils.Big
```

```
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>git checkout master
```

```
Switched to branch 'master'
```

```
C:\D19\SA1\myapp>git log ..moreprimes
```

```
commit c7516d72fe82ca1cda91070cb3bade0afe4c808a (moreprimes)
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Mon Sep 2 21:26:47 2019 +0200
```

```
Fixed comparison ot avoid Utils.Big
```

```
commit ee5ff8e014a3010e90bd568183cf4bc0f66f65dd
```

```
Author: Morten Kromberg <mkrom@dyalog.com>
```

```
Date: Mon Sep 2 21:25:28 2019 +0200
```

```
Added 23 to list of primes
```

```
C:\D19\SA1\myapp>git merge moreprimes
```

```
Updating 118d87e..c7516d7
```

```
Fast-forward
```

```
IsPrime.aplf | 2 +-  
primes.apla | 2 +-  
2 files changed, 2 insertions(+), 2 deletions(-)
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>
```

```
C:\D19\SA1\myapp>
```

GITLENS

REPOSITORIES

- myapp master
  - master
    - Compare master (working) with <branch, tag, or ref>
  - Branches
    - master
      - Less than a week ago
        - Fixed comparison of avoid Utils.Big +0 ~1 -0 • You, an hour ago
          - IsPrime.aplf**
        - Added 23 to list of primes +0 ~1 -0 • You, an hour ago
          - primes.apla
        - Added comment to Avg +0 ~1 -0 • You, a day ago
          - Avg.aplf
        - First commit +6 ~0 -0 • You, a day ago
      - Contributors
      - Remotes
      - Stashes
      - Tags

FILE HISTORY

LINE HISTORY

COMPARE

SEARCH COMMITS

```

c: > D19 > SA1 > myapp > IsPrime.aplf
1  r+IsPrime n
2 - :If Utils.Big n
3      r+msg
4  :Else
5      r+neprimes
6  :EndIf
7

1  r+IsPrime n
2 + :If n>~1tprimes
3      r+msg
4  :Else
5      r+neprimes
6  :EndIf
7
    
```

You, an hour ago

Ln 2, Col 1 Spaces: 4 UTF-8 APL

# git clone

- 🟠 `git init` turns a local folder into a "repo"
- 🟠 `git clone` makes a copy of a remote repo

```
C:\D19\SA1> git clone https://github.com/Dyalog19/SA1-Utills Utills
Cloning into 'Utills'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
```

Exercise: Clone the SA1-Utills repo



# git pull and push

- After cloning, you work with the repo as if it were local.
- Note that `git commit` only changes your local copy of the repository!
- Every now and again, you should
  - `git pull` to merge changes made by others into your local copy
  - `git push` to make your own commits available to others
- You should **ALWAYS** pull before you push!!!  
(Many GUIs will not let you push before you pull.)



# git pull and push

- The syntax is:

```
git pull origin master  
git push origin master
```

- `origin` is a reference to the remote repository (the *origin* of the local copy)
- `master` is the name of the branch we are working on
- All of this is easier if you use a GUI like **VS Code**
- You will need a GitHub user id to `push`



# Merge conflicts

- Eventually, you will change the same line of code that someone else did



Dyalog19 / SA1-Utills

Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master SA1-Utills / NoZero.aplf Find file Copy path

mkromberg Remove all 42's 1b1f472 now  
1 contributor

2 lines (1 sloc) | 19 Bytes Raw Blame History  
1 NoZero←{ω~42}

```
C:\D19\SA1>cd Utils
```

```
C:\D19\SA1\Utils>git status
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   NoZero.aplf
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
C:\D19\SA1\Utils>git diff
```

```
diff --git a/NoZero.aplf b/NoZero.aplf
```

```
index 4dc5a8b..87f8ca8 100644
```

```
--- a/NoZero.aplf
```

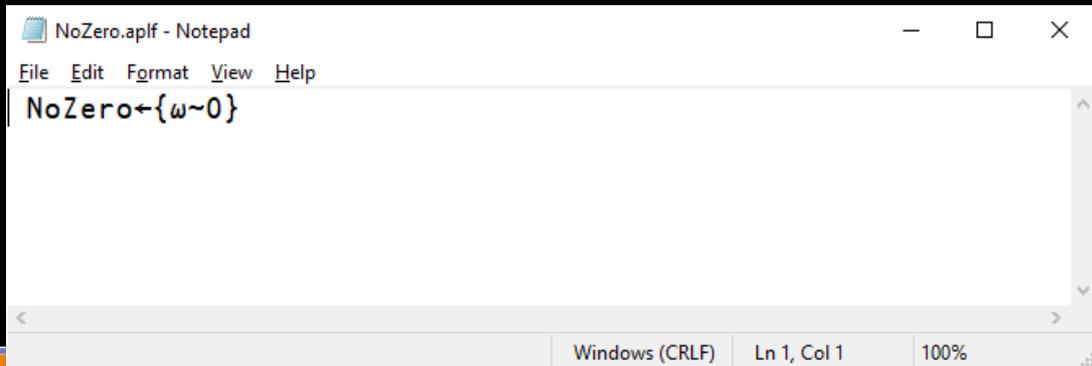
```
+++ b/NoZero.aplf
```

```
@@ -1,1 @@
```

```
- NoZero<E2><86><90>{<E2><8D><B5>~1}
```

```
+ NoZero<E2><86><90>{<E2><8D><B5>~0}
```

```
C:\D19\SA1\Utils>notepad NoZero.aplf
```



```
NoZero←{ω~0}
```

```
Command Prompt
C:\D19\SA1\Utils>notepad NoZero.aplf

C:\D19\SA1\Utils>
C:\D19\SA1\Utils>git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Dyalog19/SA1-Utills
* branch                master       -> FETCH_HEAD
   2b233fc..1b1f472     master       -> origin/master
error: Your local changes to the following files would be overwritten by merge:
       NoZero.aplf
Please commit your changes or stash them before you merge.
Aborting
Updating 2b233fc..1b1f472
```



```
Command Prompt
Aborting
Updating 2b233fc..1b1f472
C:\D19\SA1\Utils>git add .
C:\D19\SA1\Utils>git commit -m "Fixed NoZero to remove zeros"
[master 9ea733b] Fixed NoZero to remove zeros
1 file changed, 1 insertion(+), 1 deletion(-)
C:\D19\SA1\Utils>git pull origin master
From https://github.com/Dyalog19/SA1-Utills
* branch          master      -> FETCH_HEAD
Auto-merging NoZero.aplf
CONFLICT (content): Merge conflict in NoZero.aplf
Automatic merge failed; fix conflicts and then commit the result.
C:\D19\SA1\Utils>_
```



# Resolving Merge Conflicts

- You would usually do one of:
  - Delete lines <<<<<<< to =====
  - Delete lines ===== to >>>>>>>
- ... then commit and push



```
NoZero.aplf - Notepad
File Edit Format View Help
<<<<<<< HEAD
NoZero←{ω~0}
=====
NoZero←{ω~42}
>>>>>>> 1b1f472be2a0ac95ac683354b428e89f59b710e4
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```



SOURCE CONTROL: GIT ✓ ↻ ...

Message (press Ctrl+Enter to commit)

MERGE CHANGES 1

☰ NoZero.aplf 📄 + C

CHANGES 0

☰ NoZero.aplf ×

☰ NoZero.aplf

```

1 <<<<<< HEAD You, a few seconds ago • Uncommitted changes
2 NoZero+{ω-0}
3 =====
4 NoZero+{ω-42}
5 >>>>>> 1b1f472be2a0ac95ac683354b428e89f59b710e4 (Incoming Change)
6

```

SOURCE CONTROL: GIT ✓ ↻ ...

Message (press Ctrl+Enter to commit)

**MERGE CHANGES** 1

☰ NoZero.aplf 📄 + C

**CHANGES** 0

```

NoZero.aplf X
☰ NoZero.aplf
You, a few seconds ago | 1 author (You) | Accept Current Change | Accept Incoming Change | Accept Both Changes |
1 <<<<<< HEAD You, a few seconds ago • Uncommitted changes
2 NoZero+{ω-0}
3 =====
4 NoZero+{ω-42}
5 >>>>>> 1b1f472be2a0ac95ac683354b428e89f59b710e4 (Incoming Change)
6

```

**If the file is being watched by Link, "anything can happen" at this point**

# Exercises

- Add a new function to Utils and push it to GitHub
- Collaborate with someone next to you, to make conflicting changes to the same function
- Deal with the merge conflicts
- Make changes to two different lines of code and note that auto-merge "usually works"
  - (you may need to create a function with several lines of code before this is reliable)



# Git Tutorials

- Learn Git in 20 Minutes

[youtu.be/IHaTbJPdB-s](https://youtu.be/IHaTbJPdB-s) by *Web Dev Simplified*

- Git Tutorial for Beginners: Command-Line Fundamentals

[youtu.be/HVsySz-h9r4](https://youtu.be/HVsySz-h9r4) by Corey Schafer

