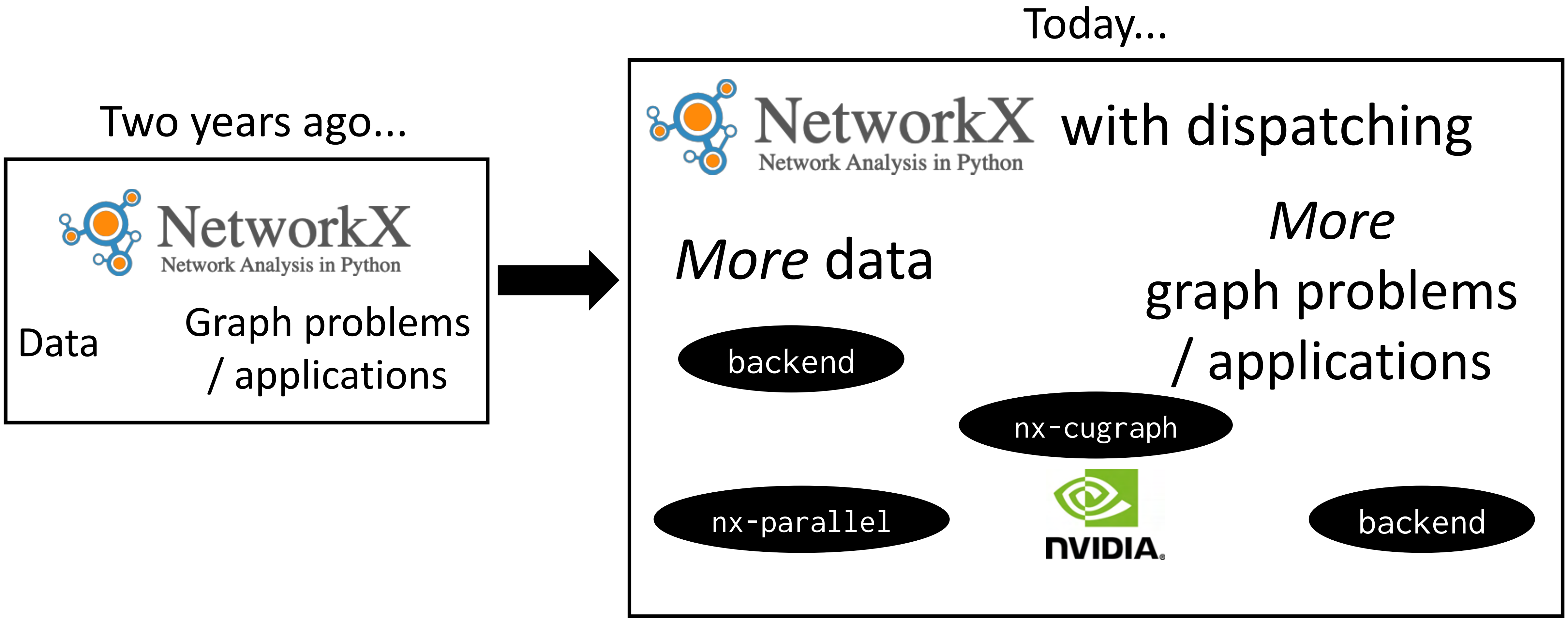
Understanding NetworkX's API Dispatching with a parallel backend

-By Erik Welch and
Aditi Juneja

Erik Welch, core NetworkX dev, OSS engineer on NVIDIA RAPIDS team



Two years ago...

NetworkX
Network Analysis in Python

Data

Graph problems
/ applications

Today...

NetworkX
Network Analysis in Python
with dispatching

More data

More
graph problems
/ applications

backend

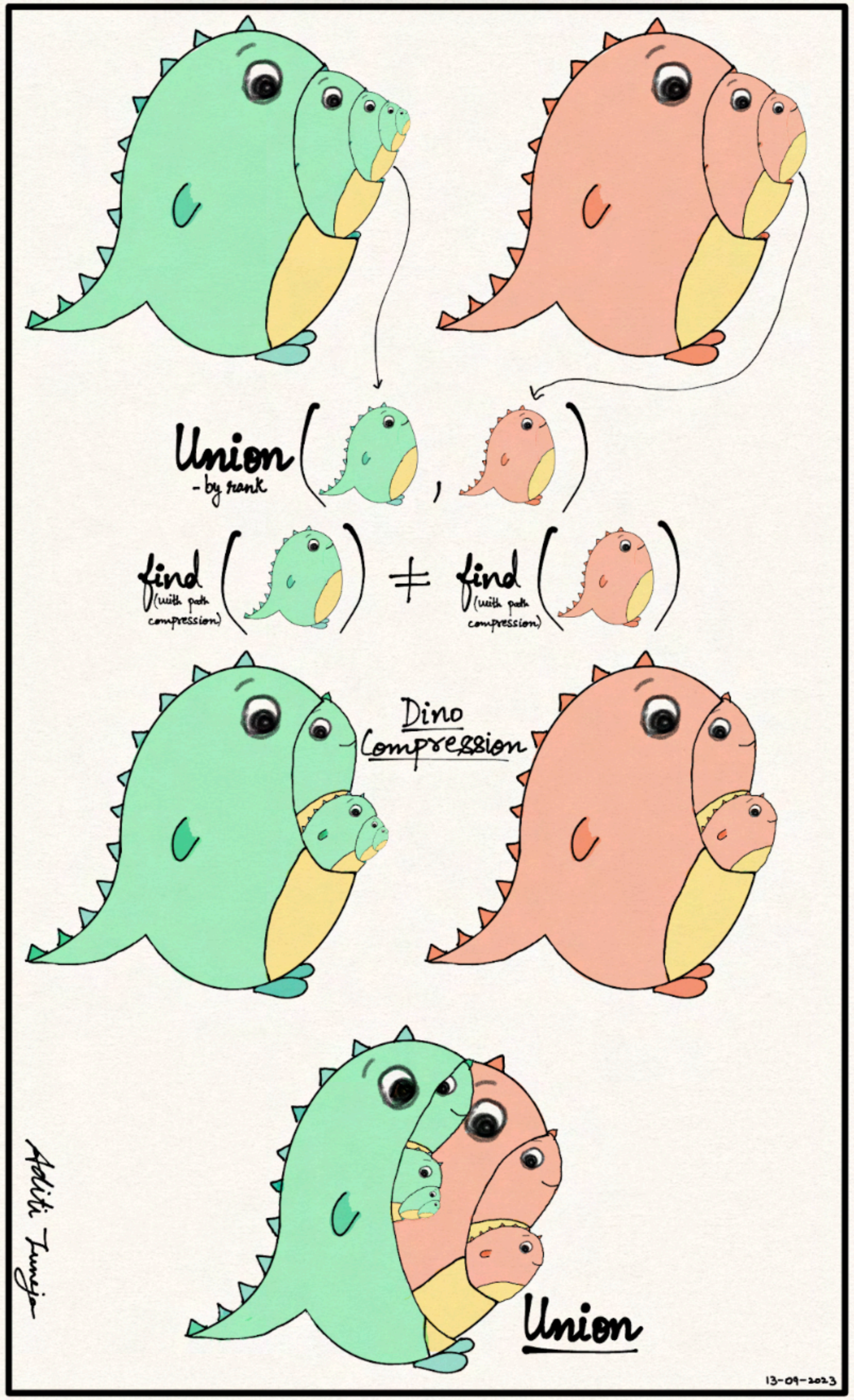
nx-cugraph

nx-parallel

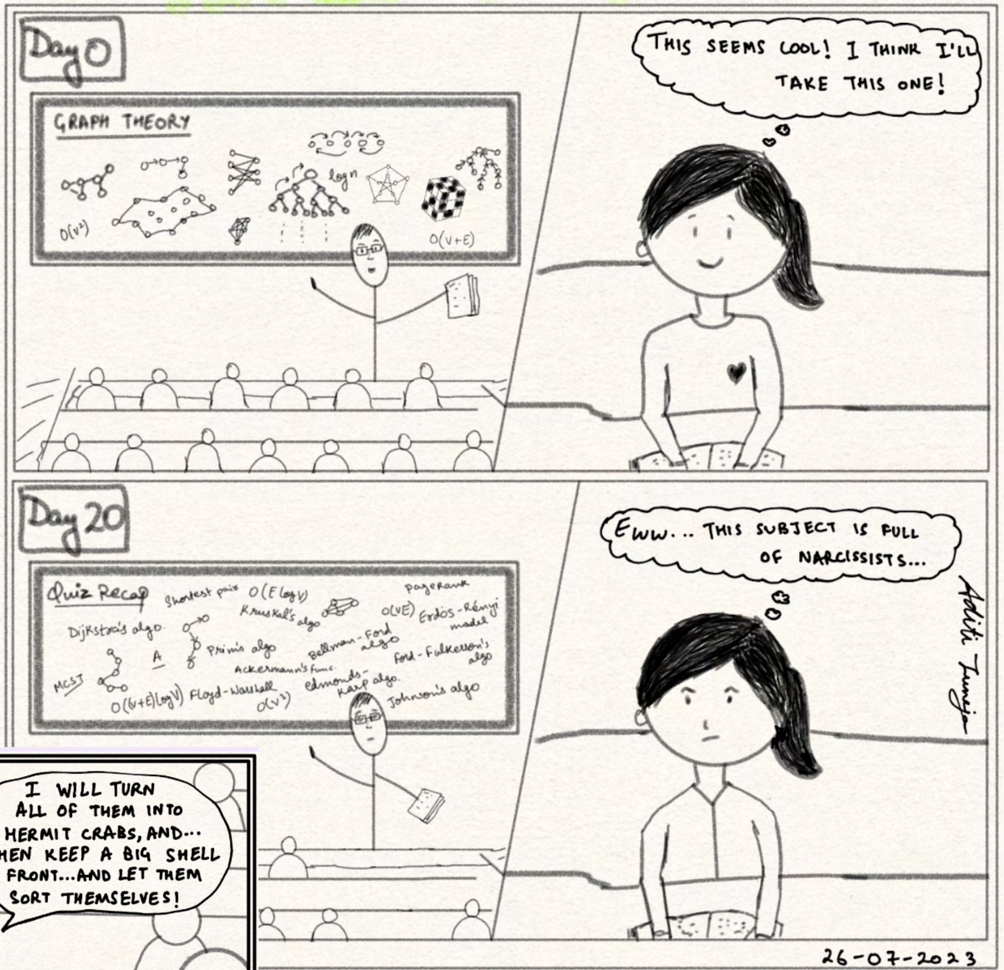
NVIDIA

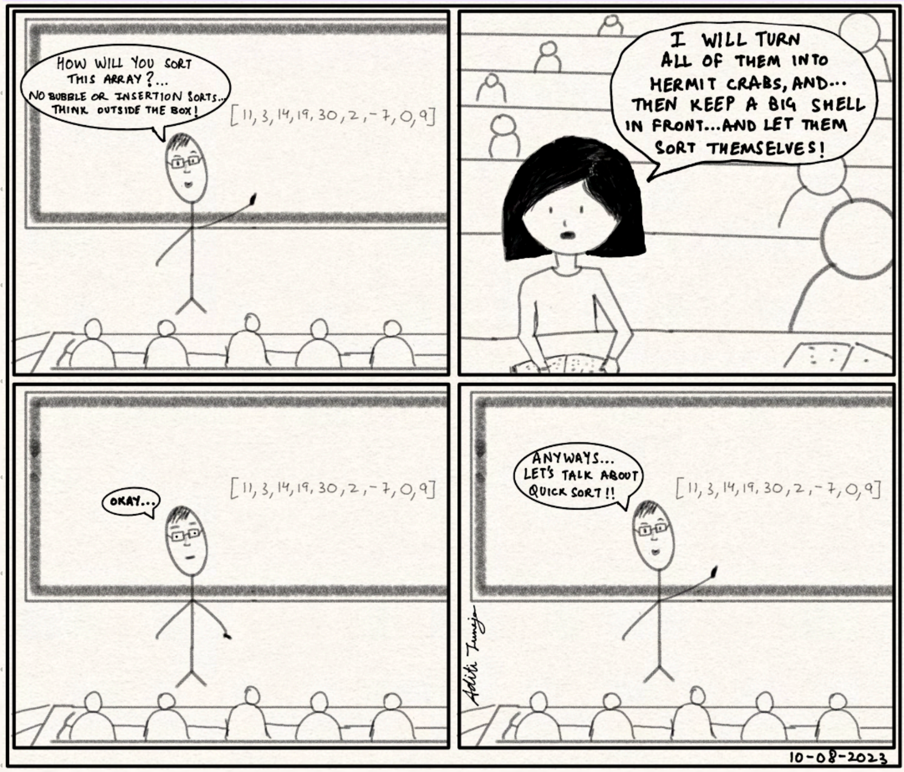
backend

See also: "Dispatching, Backend Selection, and Compatibility APIs"
Tomorrow 13:20, Room 5



“ Back in 2023... ”



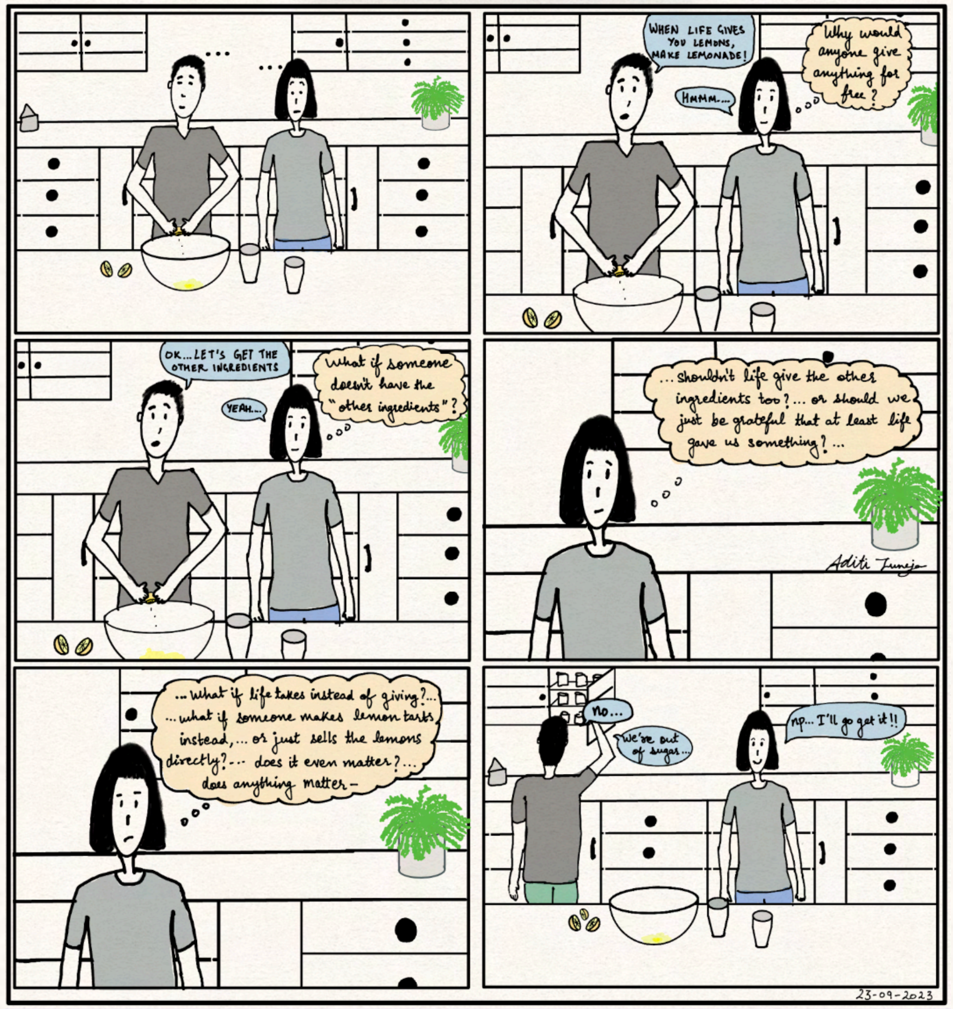


The best introduction to numbers I've ever seen—the clearest and funniest explanation of what they are and why we need them—appears in a *Sesame Street* video called *123 Count with Me*. Humphrey, an amiable but dimwitted fellow with pink fur and a green nose, is working the lunch shift at the Furry Arms Hotel when he takes a call from a roomful of penguins. Humphrey listens carefully and then calls out their order to the kitchen: "Fish, fish, fish, fish, fish, fish." This prompts Ernie to enlighten him about the virtues of the number six.

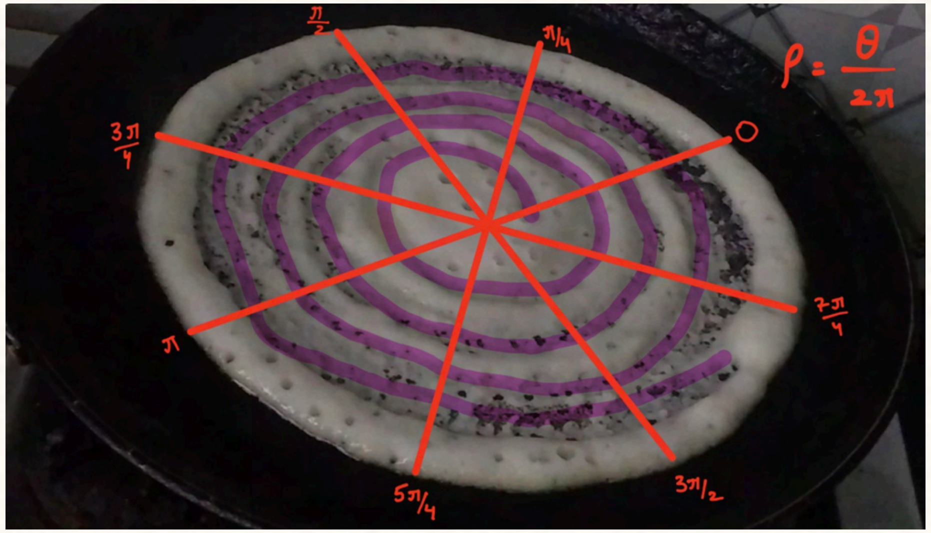
" Some more... "


Hey!
Hi...
LONGTIME?.. UH..
yeah..
HOW IS--
- LISTEN, MY CAB IS WAITING--
oh-np... We'll catch-up later
Aditi Juneja
BYE..
THANKS
BYE..
"CATCH-UP"?!? AM I OLD NOW?!
20-06-2023


'NECESSITY' IS THE MOTHER OF INVENTIONS. WHO KNOWS WHO THE FATHER IS?
ISN'T THAT OBVIOUS? THE INVENTOR...
WOOOAAA.....
FREAKING PATRIARCHY...
WAIT... BUT.. ISN'T 'INVENTOR' GENDER NEUTRAL?
.....
WHATEVER DUDE, WE ALL KNOW, YOU THOUGHT ABOUT A 'MALE' INVENTOR, WHEN I SAID IT.
WELL... NOT NECESSARILY.
ALSO, IT'S 'MOTHER' NOT 'PARENT'!!
WELL, A GUY CAN ALSO BE A MOTHER!
NOT BACK THEN!
BY THE WAY, I AM NOT A 'DUDE'!
WELL... 'INVENTOR' IS GENDER NEUTRAL
SORRY, GRL....
Aditi Juneja
WHY DOESN'T 'GRL' SOUND AS COOL AS 'DUDE'?
WELL... I GUESS, WE BOTH KNOW THE ANSWER TO THAT ONE!
FREAKING PATRIARCHY!
FREAKING PATRIARCHY!
30/04/22

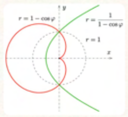

...
...
WHEN LIFE GIVES YOU LEMONS, MAKE LEMONADE!
HMMM....
Why would anyone give anything for free?
OK...LET'S GET THE OTHER INGREDIENTS
YEAH...
What if someone doesn't have the "other ingredients"?
...shouldn't life give the other ingredients too?... or should we just be grateful that at least life gave us something?...
Aditi Juneja
...What if life takes instead of giving?... ...what if someone makes lemon tarts instead,... or just sells the lemons directly?... does it even matter?... does anything matter-
No...
We're out of sugar...
np... I'll go get it!!
23-09-2023

Math-food play

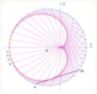


inflating

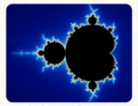
just everywhere....



parabola inverted about a circle



As an envelope curve



In the mandelbrot set

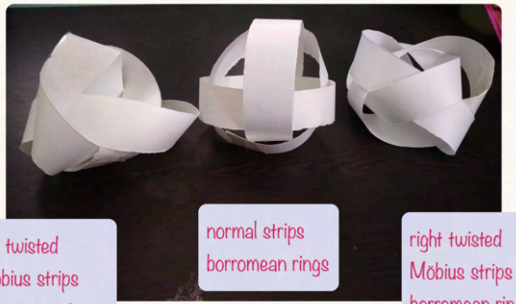
cardioid mic pick-up pattern

https://en.wikipedia.org/wiki/Cardioid

the perfect evening...

how can someone not burn sanitizer when there are burning candles around?! ...

...and here are some Borromean rings...



left twisted Möbius strips borromean rings

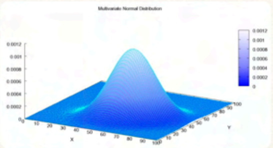
normal strips borromean rings

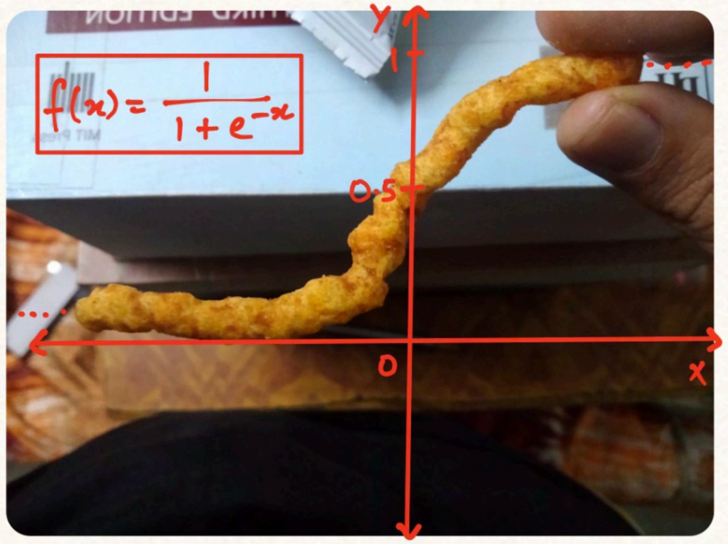
right twisted Möbius strips borromean rings

https://en.wikipedia.or

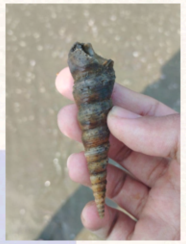
...and make a cute little ladoo!!

...looks like a gaussian distribution in 3d (Bivariate normal joint density)



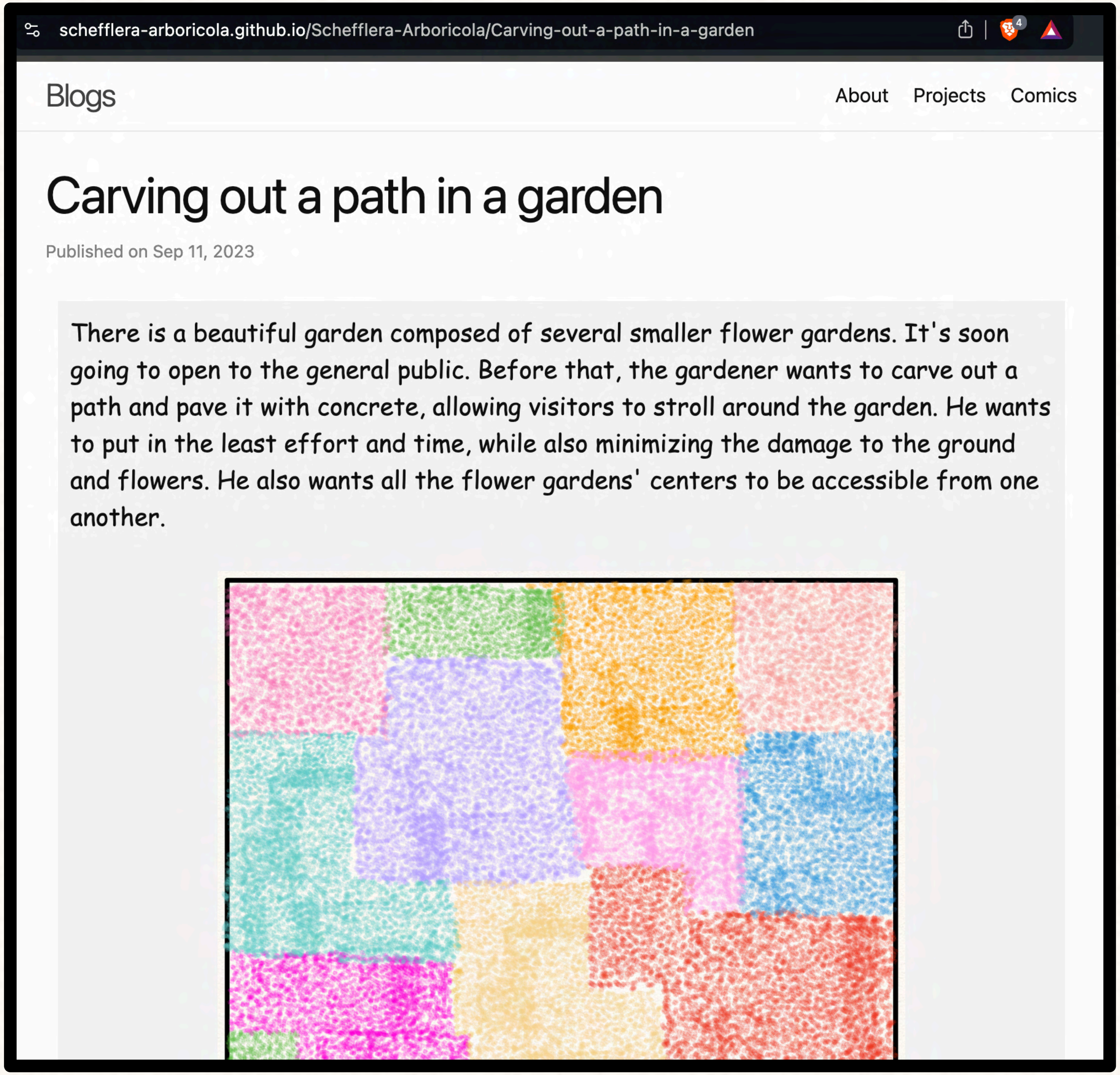


...that's a good-sigmoid-looking Kurkura.



These should be eaten the way they were built - fry by fry, layer by layer, like a seashell on the shore, formed over the years, layer by layer building up to that beautiful, intricate structure.

My last contribution
to the ”graph world”...



schefflera-arboricola.github.io/Schefflera-Arboricola/Carving-out-a-path-in-a-garden

Blogs

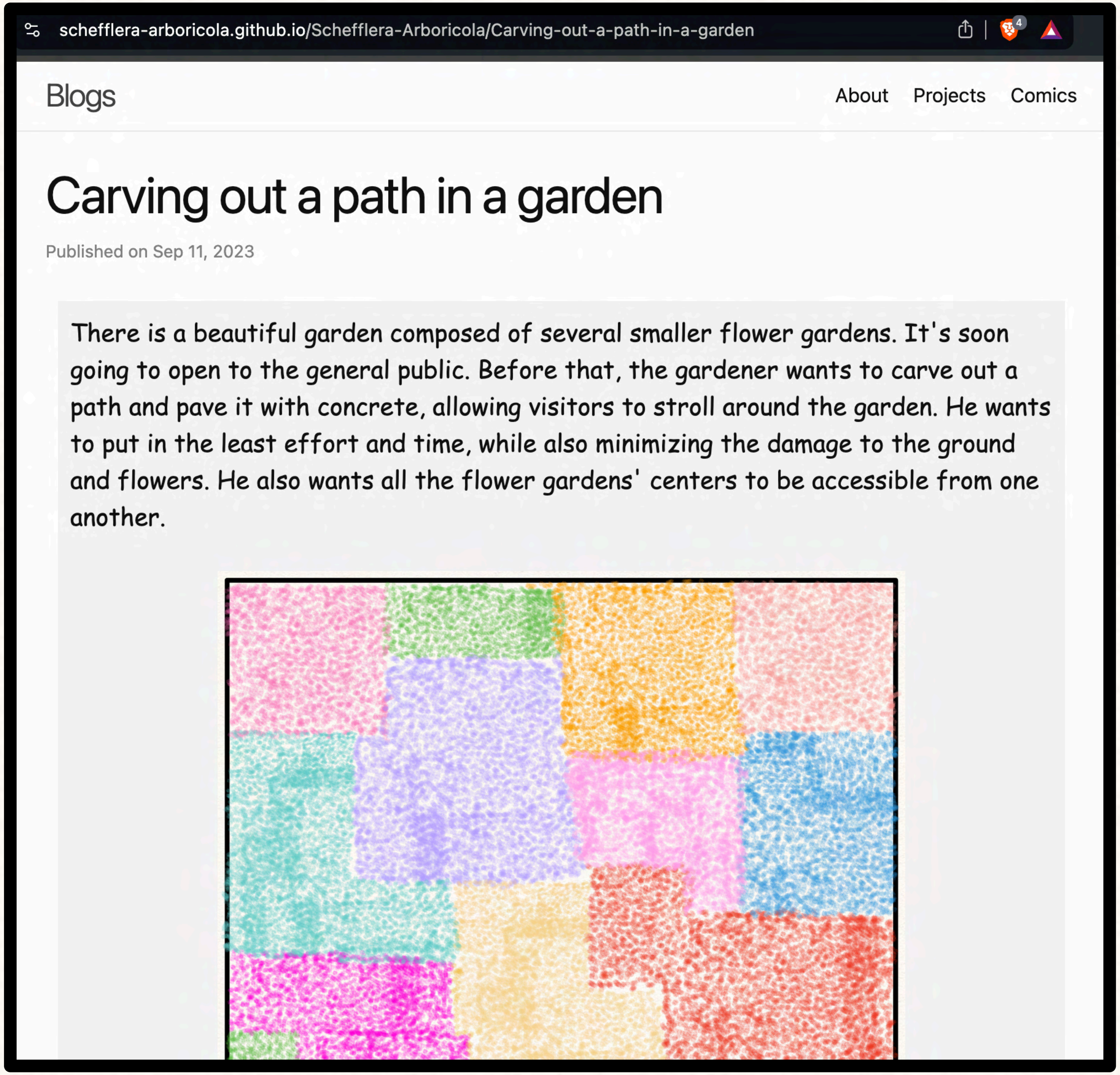
About Projects Comics

Carving out a path in a garden

Published on Sep 11, 2023

There is a beautiful garden composed of several smaller flower gardens. It's soon going to open to the general public. Before that, the gardener wants to carve out a path and pave it with concrete, allowing visitors to stroll around the garden. He wants to put in the least effort and time, while also minimizing the damage to the ground and flowers. He also wants all the flower gardens' centers to be accessible from one another.

My last contribution to the ”graph world”...



schefflera-arboricola.github.io/Schefflera-Arboricola/Carving-out-a-path-in-a-garden

Blogs

About Projects Comics

Carving out a path in a garden

Published on Sep 11, 2023

There is a beautiful garden composed of several smaller flower gardens. It's soon going to open to the general public. Before that, the gardener wants to carve out a path and pave it with concrete, allowing visitors to stroll around the garden. He wants to put in the least effort and time, while also minimizing the damage to the ground and flowers. He also wants all the flower gardens' centers to be accessible from one another.

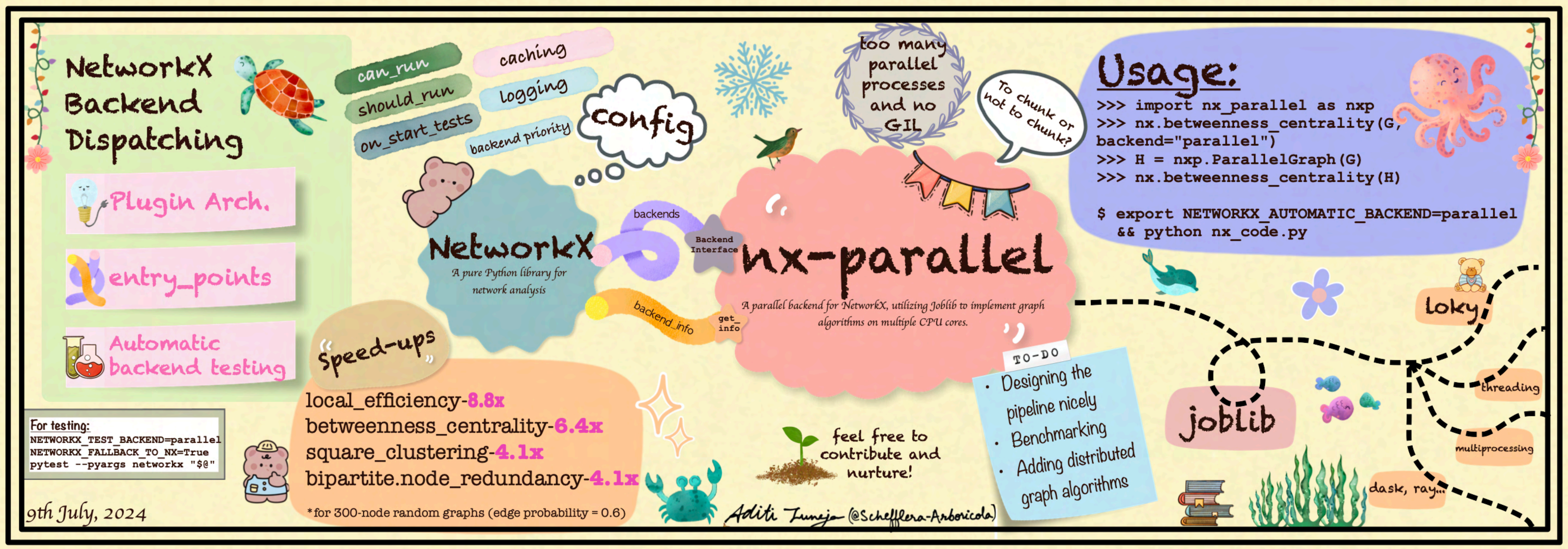
But....

Outreachy

NetworkX

nx-parallel

SciPy 2024 Poster



NetworkX Backend Dispatching

- Plugin Arch.
- entry_points
- Automatic backend testing

For testing:

```
NETWORKX_TEST_BACKEND=parallel
NETWORKX_FALLBACK_TO_NX=True
pytest --pyargs networkx "$@"
```

9th July, 2024

can_run
should_run
on_start_tests
caching
logging
backend priority
config

too many parallel processes and no GIL

To chunk or not to chunk?

NetworkX

A pure Python library for network analysis

backends

Backend Interface

backend_info

get_info

nx-parallel

"A parallel backend for NetworkX, utilizing Joblib to implement graph algorithms on multiple CPU cores."

Usage:

```
>>> import nx_parallel as nxp
>>> nx.betweenness_centrality(G,
backend="parallel")
>>> H = nxp.ParallelGraph(G)
>>> nx.betweenness_centrality(H)

$ export NETWORKX_AUTOMATIC_BACKEND=parallel
  && python nx_code.py
```

"Speed-ups"

local_efficiency-**8.8x**
betweenness_centrality-**6.4x**
square_clustering-**4.1x**
bipartite.node_redundancy-**4.1x**

*for 300-node random graphs (edge probability = 0.6)

feel free to contribute and nurture!

TO-DO

- Designing the pipeline nicely
- Benchmarking
- Adding distributed graph algorithms

joblib

loky

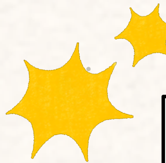
threading

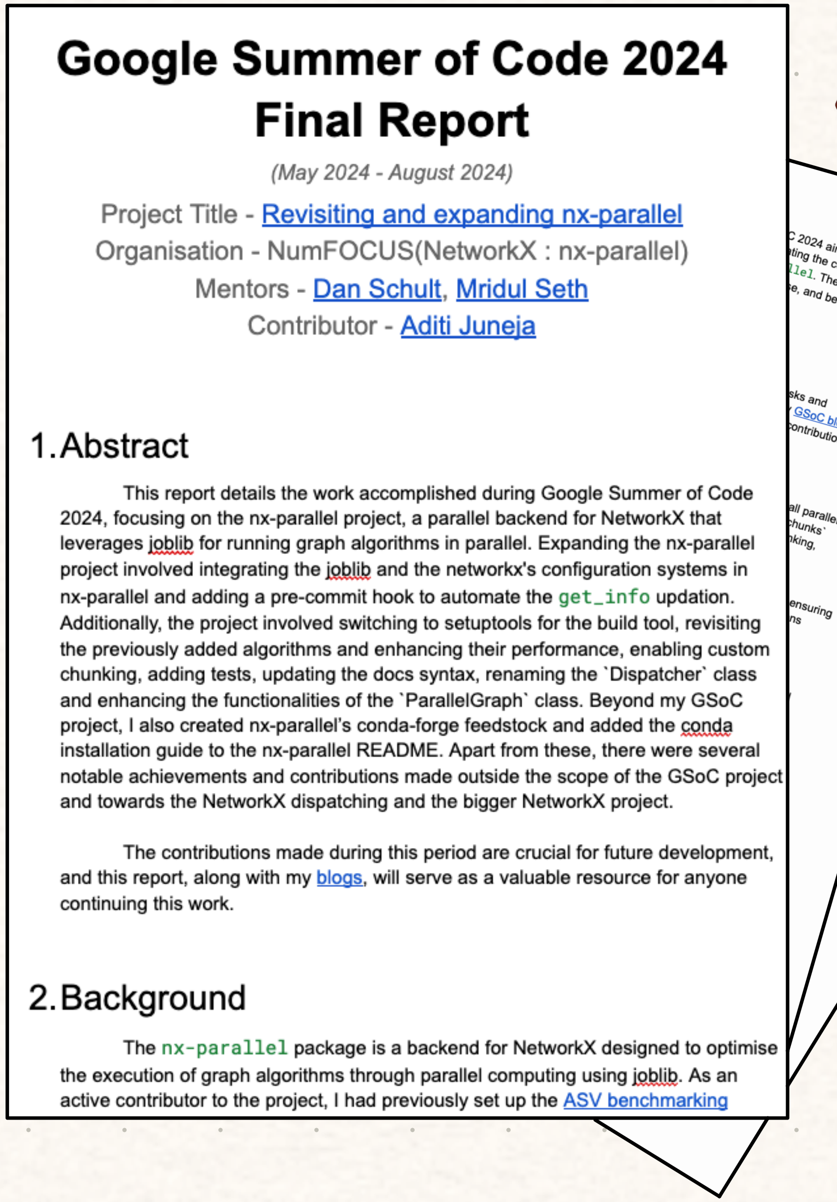
multiprocessing

dask, ray...

Aditi Juneja (@Schefflera-Arboricola)

2 days ago...





Google Summer of Code 2024 Final Report

(May 2024 - August 2024)

Project Title - Revisiting and expanding nx-parallel
Organisation - NumFOCUS(NetworkX : nx-parallel)
Mentors - Dan Schult, Mridul Seth
Contributor - Aditi Juneja

1. Abstract

This report details the work accomplished during Google Summer of Code 2024, focusing on the nx-parallel project, a parallel backend for NetworkX that leverages joblib for running graph algorithms in parallel. Expanding the nx-parallel project involved integrating the joblib and the networkx's configuration systems in nx-parallel and adding a pre-commit hook to automate the `get_info` updation. Additionally, the project involved switching to setuptools for the build tool, revisiting the previously added algorithms and enhancing their performance, enabling custom chunking, adding tests, updating the docs syntax, renaming the `Dispatcher` class and enhancing the functionalities of the `ParallelGraph` class. Beyond my GSoC project, I also created nx-parallel's conda-forge feedstock and added the conda installation guide to the nx-parallel README. Apart from these, there were several notable achievements and contributions made outside the scope of the GSoC project and towards the NetworkX dispatching and the bigger NetworkX project.

The contributions made during this period are crucial for future development, and this report, along with my blogs, will serve as a valuable resource for anyone continuing this work.

2. Background

The `nx-parallel` package is a backend for NetworkX designed to optimise the execution of graph algorithms through parallel computing using joblib. As an active contributor to the project, I had previously set up the ASV benchmarking

Understanding NetworkX's API Dispatching with a parallel backend

Understanding NetworkX's API Dispatching with a parallel backend

SPEC2 - API Dispatching

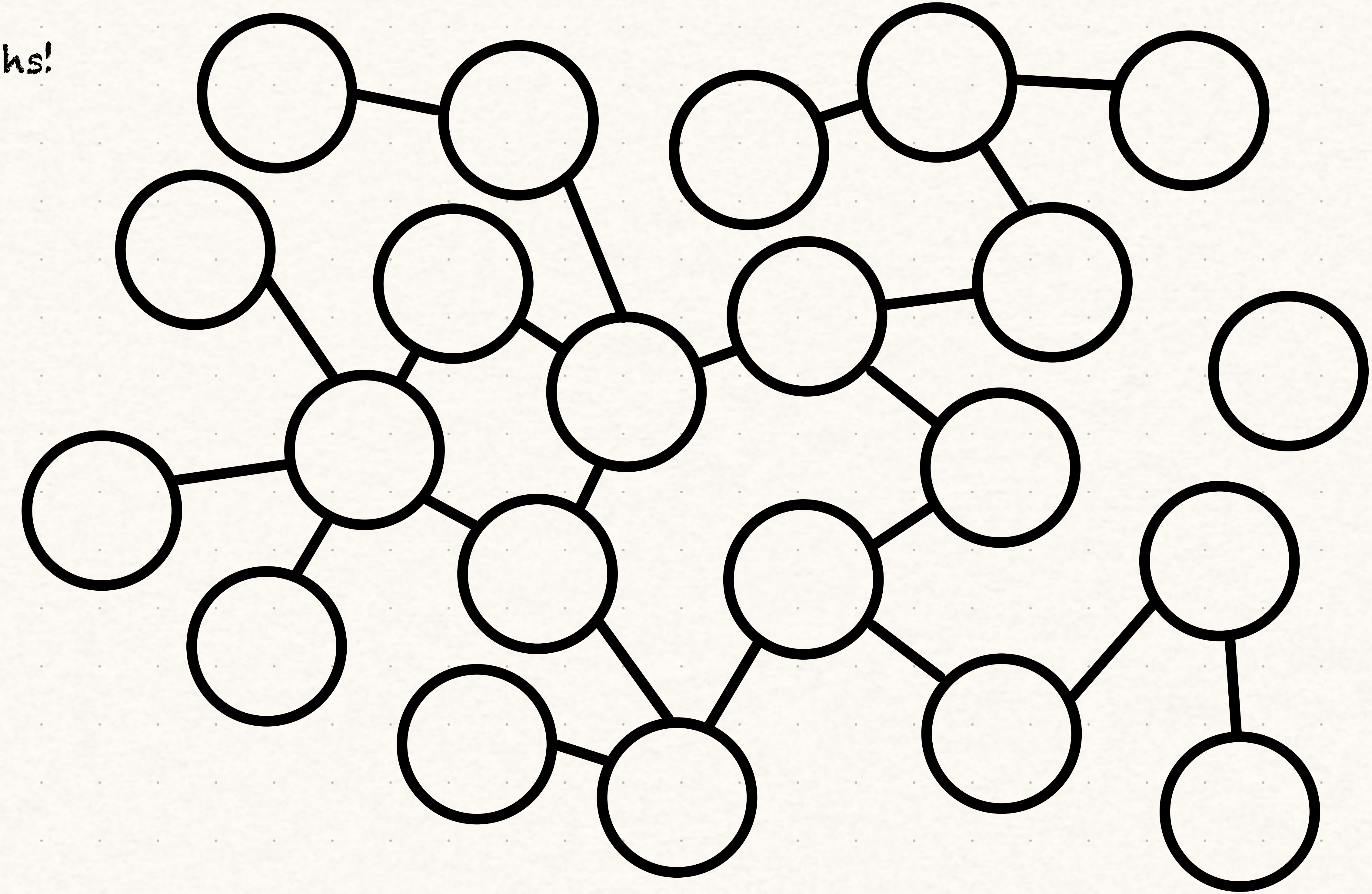
Understanding **NetworkX**'s API Dispatching with a parallel backend

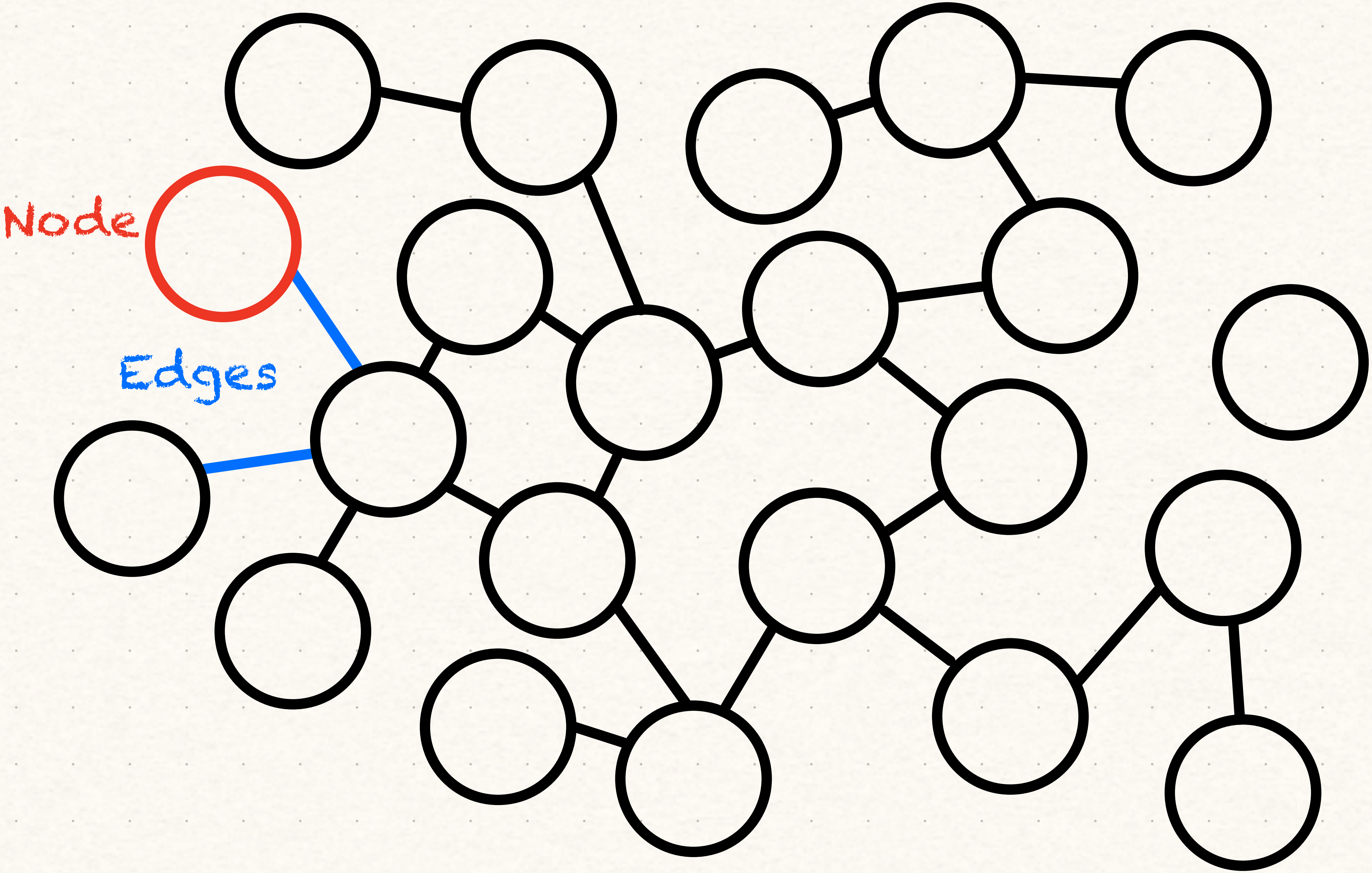
Understanding NetworkX's API Dispatching with a parallel backend

A graph (aka network) analysis Python library

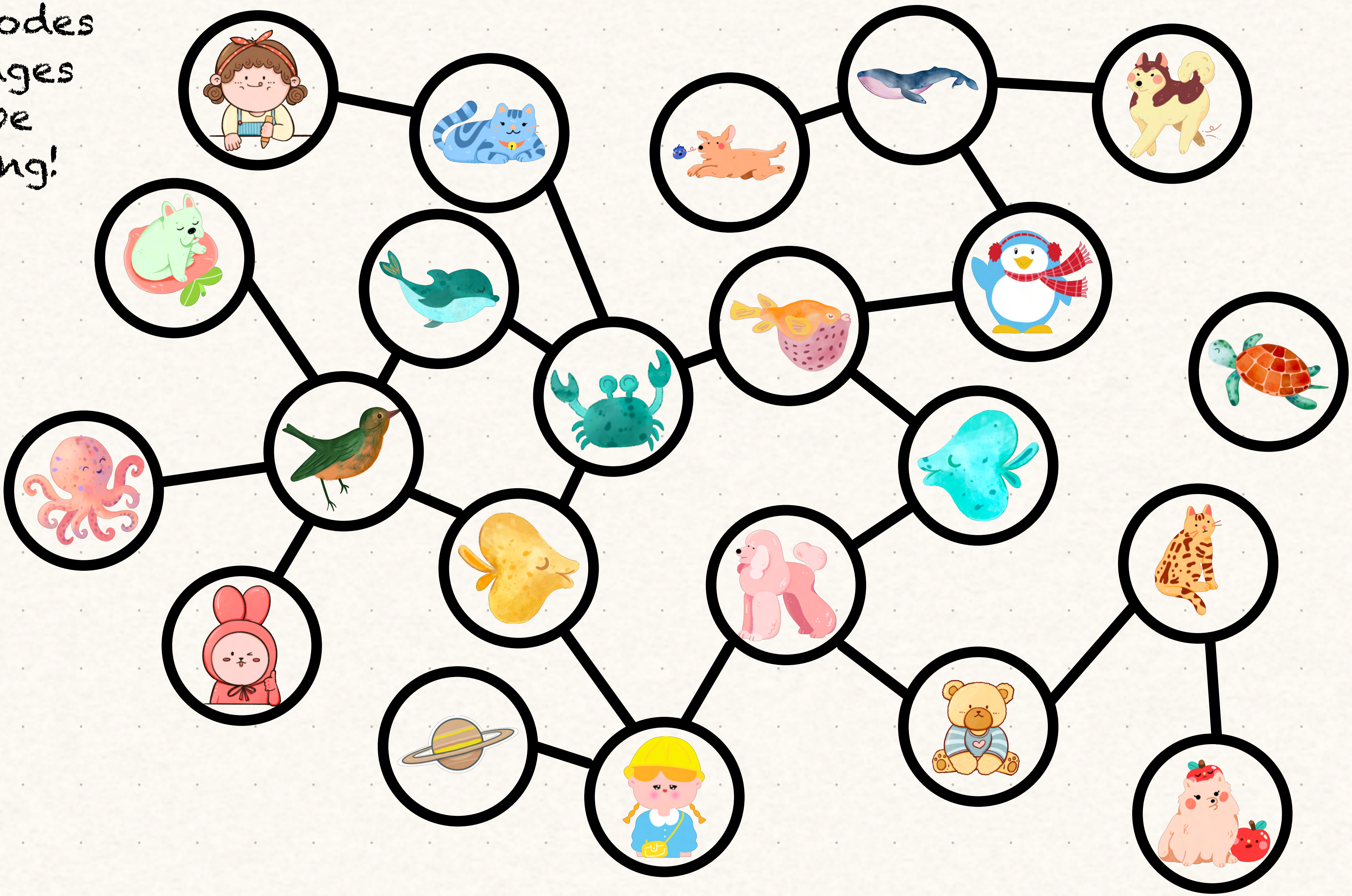
NOT these kind
of graphs

Graphs!



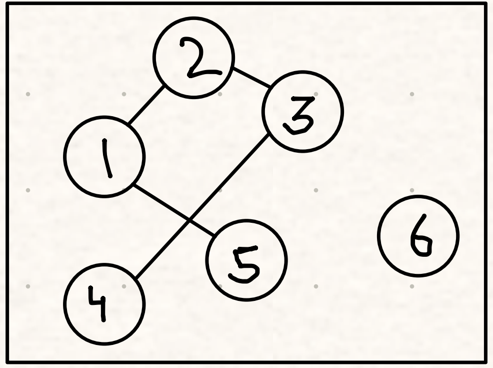

Node
Edges

These nodes
and edges
can be
anything!



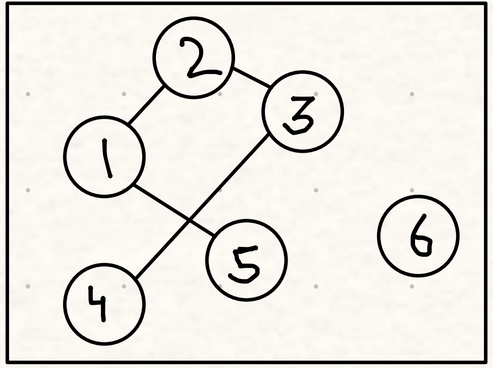
NetworkX API

```
>>> import networkx as nx

>>> G = nx.Graph()

>>> G.add_edges_from([(1,2), (3,4), (2,3),
(5,1)])

>>> G.add_node(6)

>>> nx.betweenness_centrality(G)
```



NetworkX API

```
>>> import networkx as nx

>>> G = nx.Graph()

>>> G.add_edges_from([(1,2), (3,4), (2,3),
(5,1)])

>>> G.add_node(6)

>>> nx.betweenness_centrality(G)

{1: 0.30000000000000004, 2: 0.4, 3:
0.30000000000000004, 4: 0.0, 5: 0.0, 6: 0.0}
```

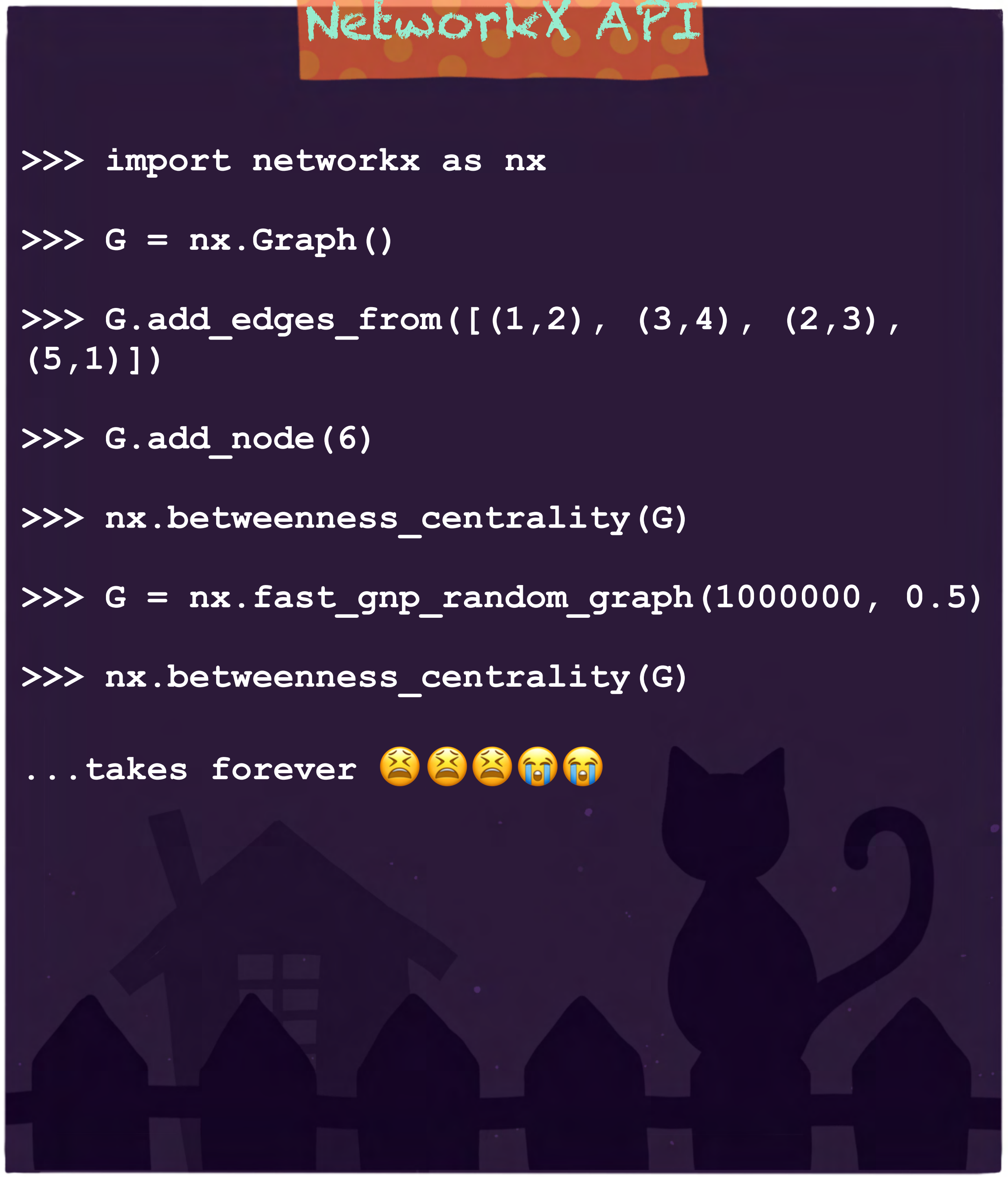


NetworkX API

Now, lets try this...

```
>>> import networkx as nx

>>> G = nx.Graph()

>>> G.add_edges_from([(1,2), (3,4), (2,3),
(5,1)])

>>> G.add_node(6)

>>> nx.betweenness_centrality(G)

>>> G = nx.fast_gnp_random_graph(1000000, 0.5)

>>> nx.betweenness_centrality(G)

...takes forever 😫😫😫😭😭
```

NetworkX API



```
>>> import networkx as nx

>>> G = nx.Graph()

>>> G.add_edges_from([(1,2), (3,4), (2,3),
(5,1)])

>>> G.add_node(6)

>>> nx.betweenness_centrality(G)
```

Now, lets try this...

```
>>> G = nx.fast_gnp_random_graph(1000000, 0.5)

>>> nx.betweenness_centrality(G)

...takes forever 😫😫😫😭😭
```

What to do now?

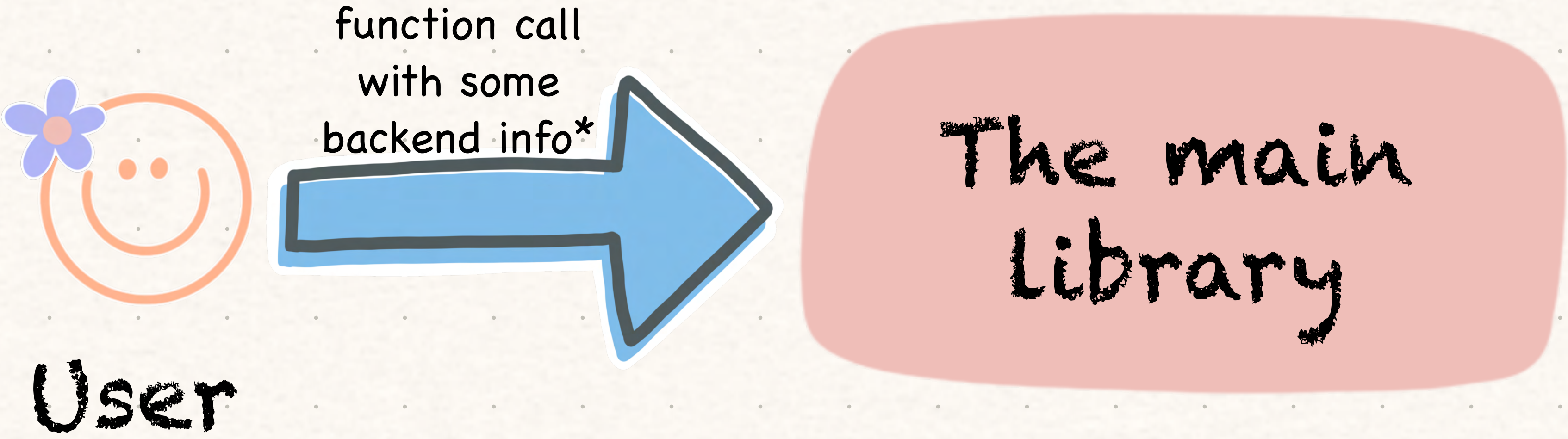
Understanding NetworkX's API Dispatching with a parallel backend

Understanding NetworkX's API Dispatching with a parallel backend

Understanding NetworkX's API Dispatching with a parallel backend

“What is Dispatching?”

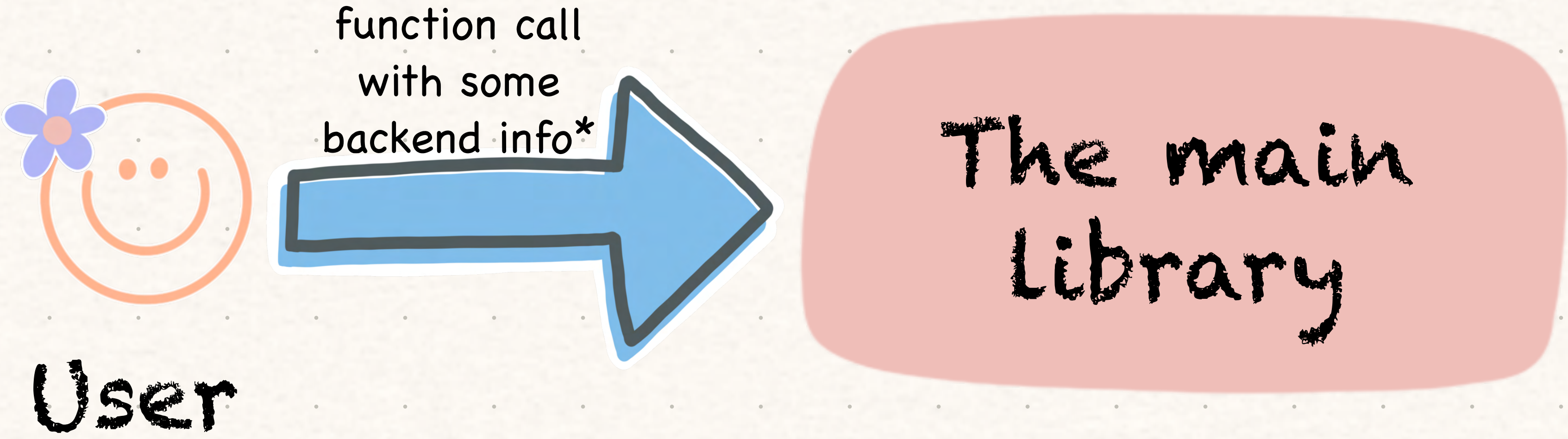
User



User

function call
with some
backend info*

The main library



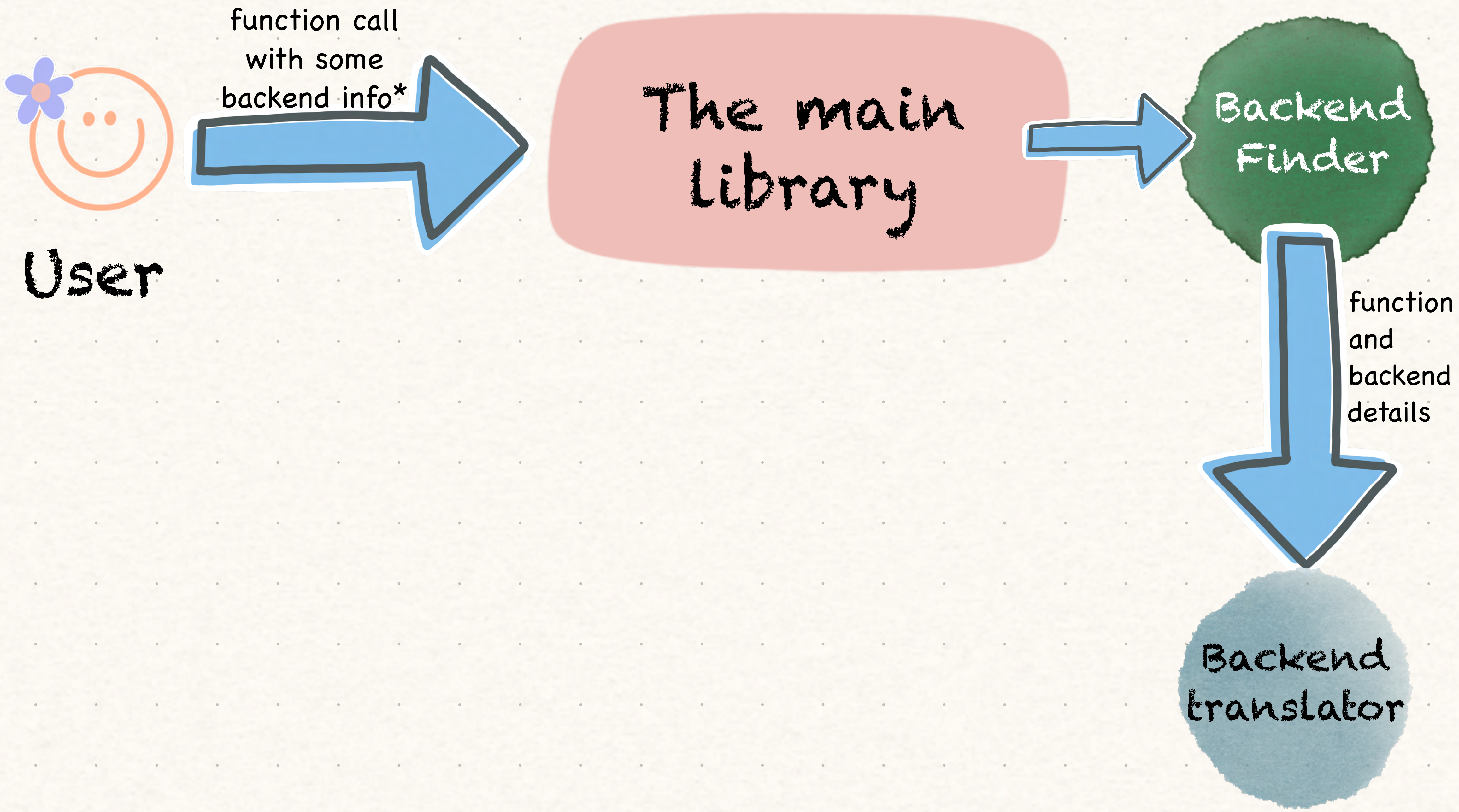
function call with some backend info*

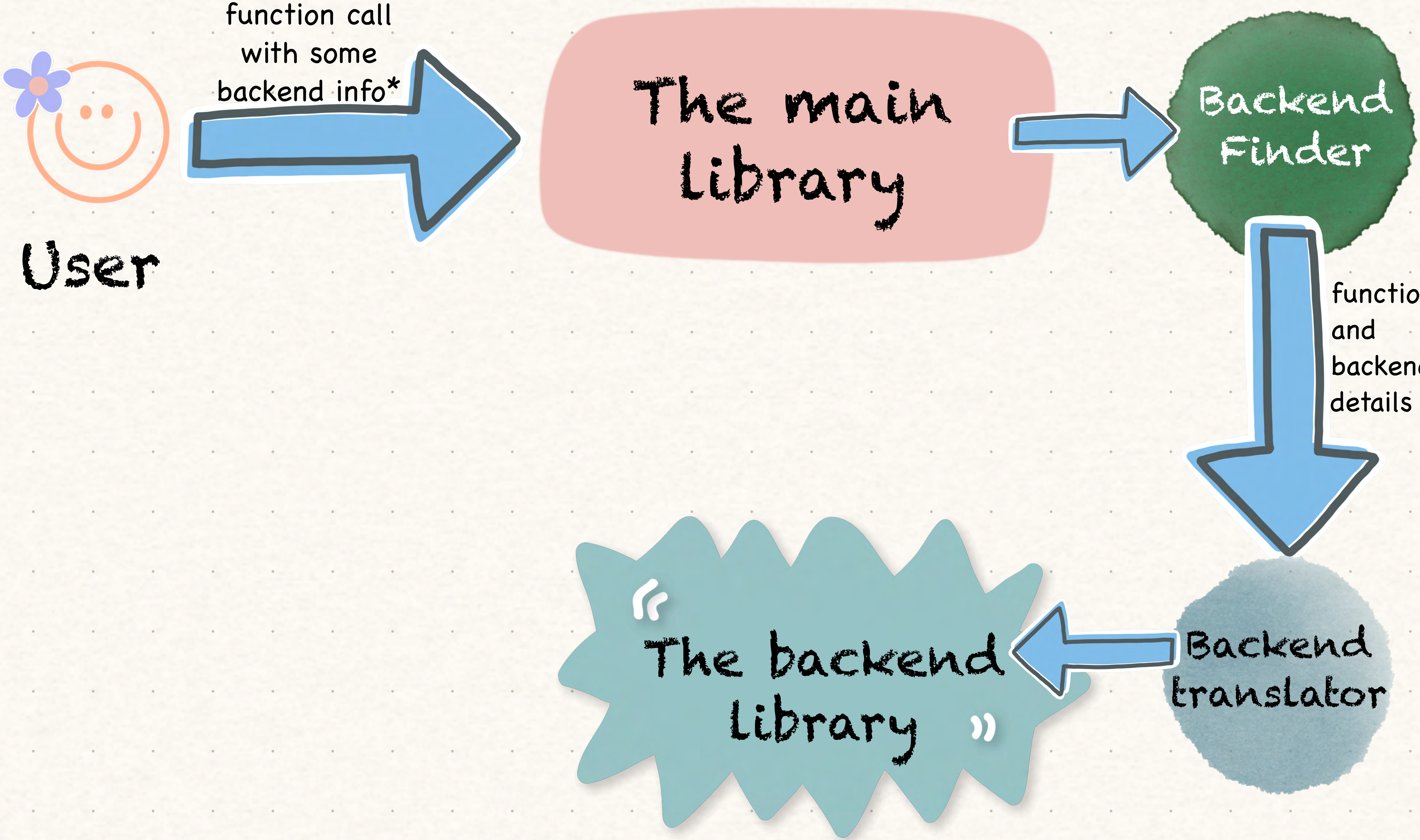
User

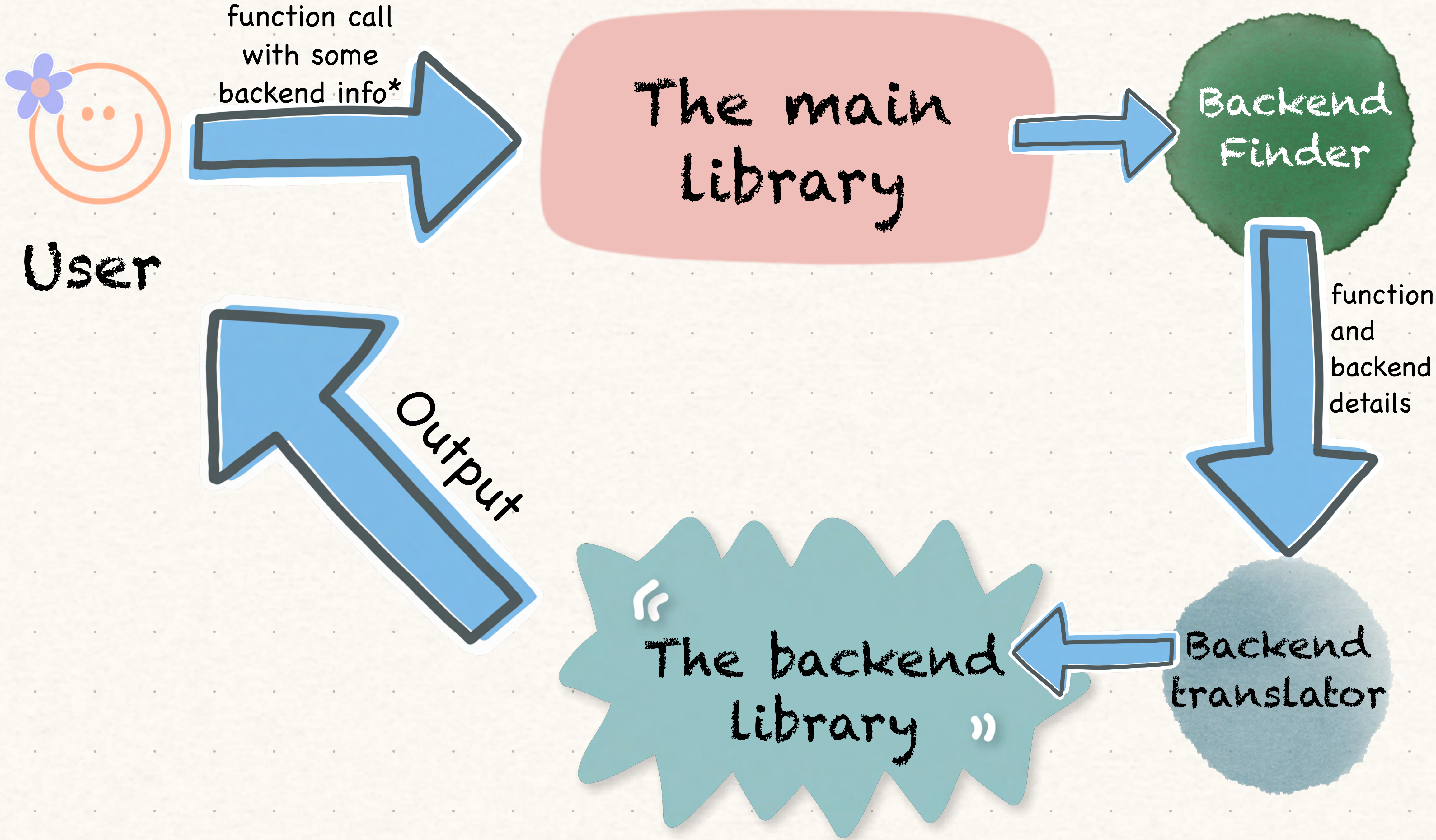
The main library

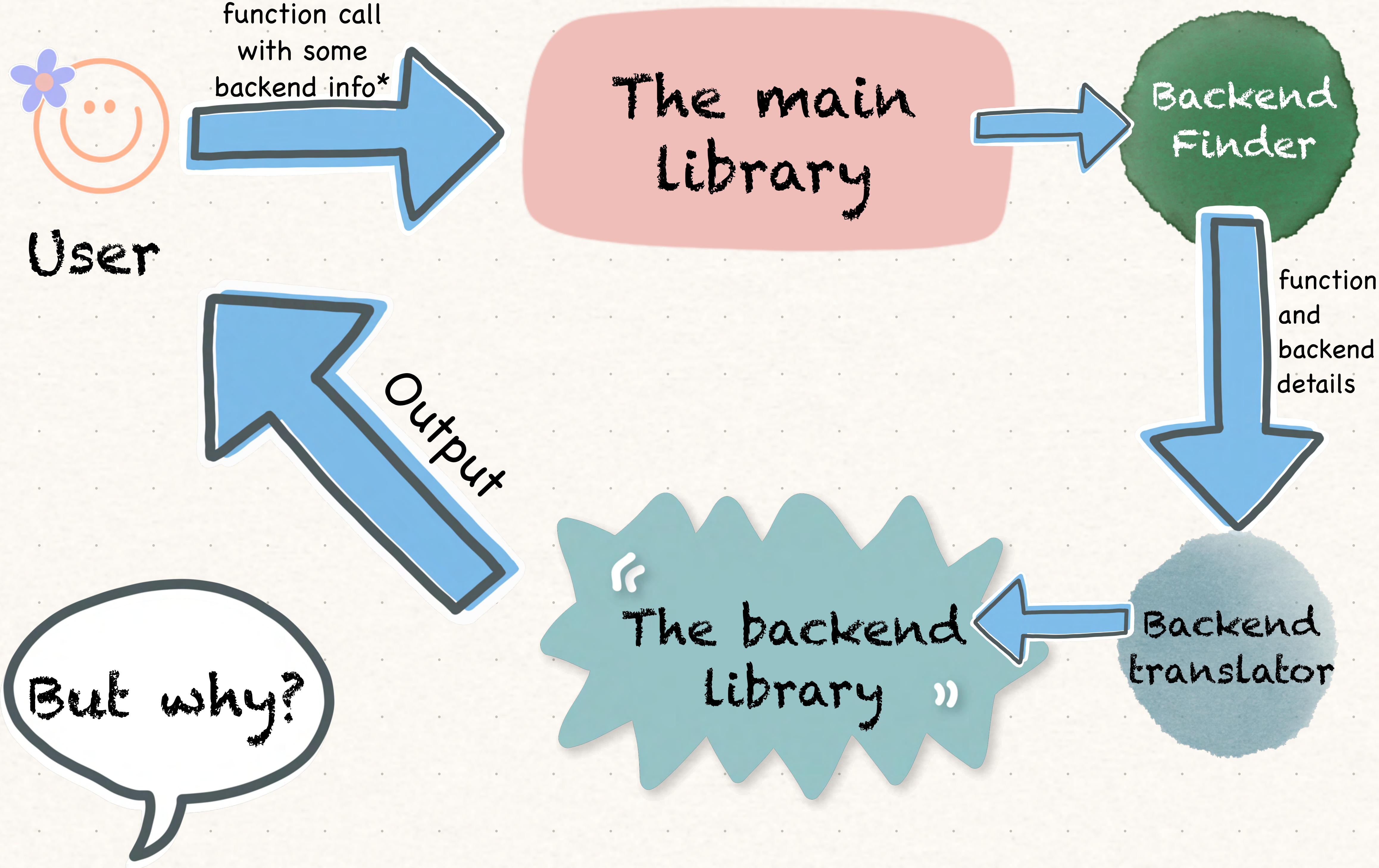
*This backend information might not be entered in the function call itself but can also be globally stored, like as an environment variable.


function call
with some
backend info*
User
The main
Library
Backend
Finder


User
function call
with some
backend info*
The main
library
Backend
Finder
function
and
backend
details
Backend
translator


User
function call with some backend info*
The main library
Backend Finder
function and backend details
Backend translator
“The backend library”


function call
with some
backend info*
The main
library
Backend
Finder
User
function
and
backend
details
Output
"The backend
library"
Backend
translator

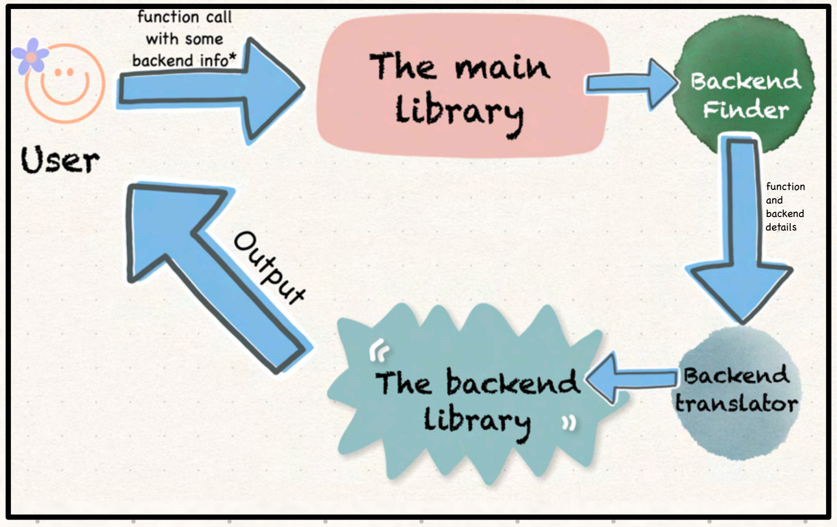

function call with some backend info*
User
The main library
Backend Finder
function and backend details
Backend translator
"The backend library"
Output
But why?

But why dispatching?

- The API remains similar for the end user, except they just have to pass some additional backend information.
- Is API dispatching the best thing for your package? - It depends...
- NetworkX is a big and old project.

Lets see API dispatching in NetworkX

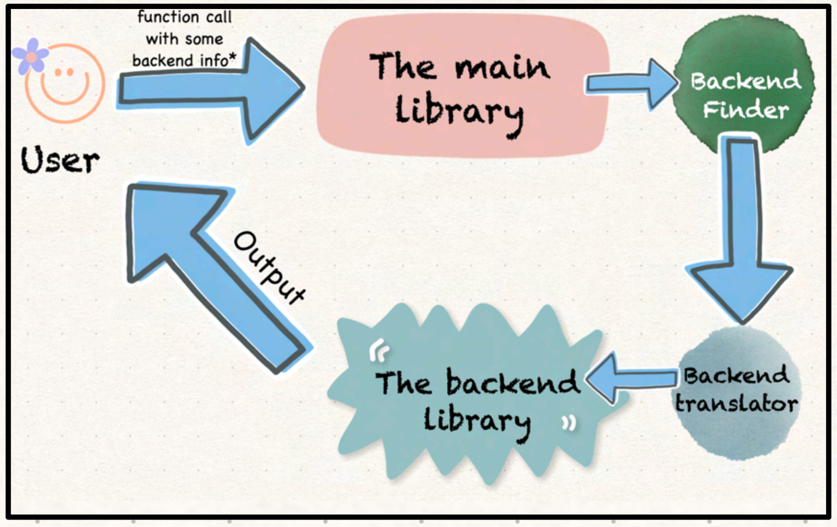
Function call with "some backend information" in NetworkX



1. `backend` kwarg in the function call:

```
nx.betweenness_centrality(G, backend="parallel")
```

Function call with "some backend information" in NetworkX



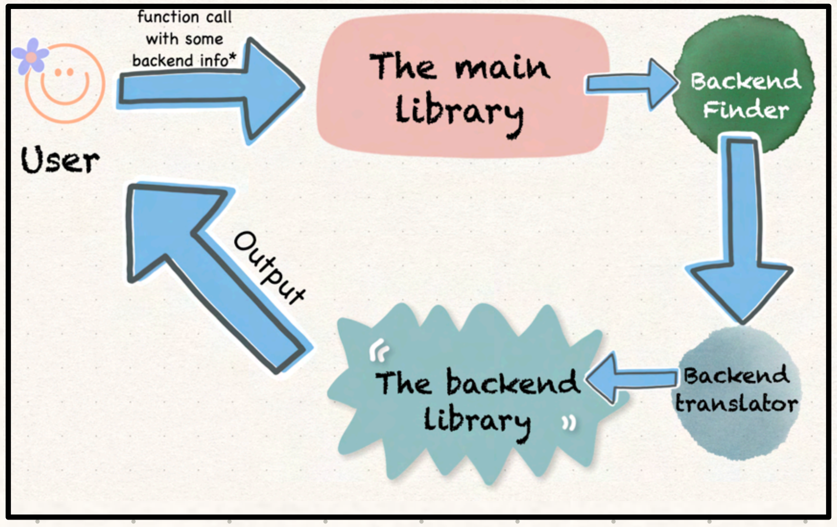
1. `backend` kwarg in the function call:

```
nx.betweenness_centrality(G, backend="parallel")
```

2. Type-based dispatching

```
import nx_parallel as nxp
H = nxp.ParallelGraph(nx.complete_graph(3))
nx.betweenness_centrality(H)
```

Function call with "some backend information" in NetworkX



1. `backend` kwarg in the function call:

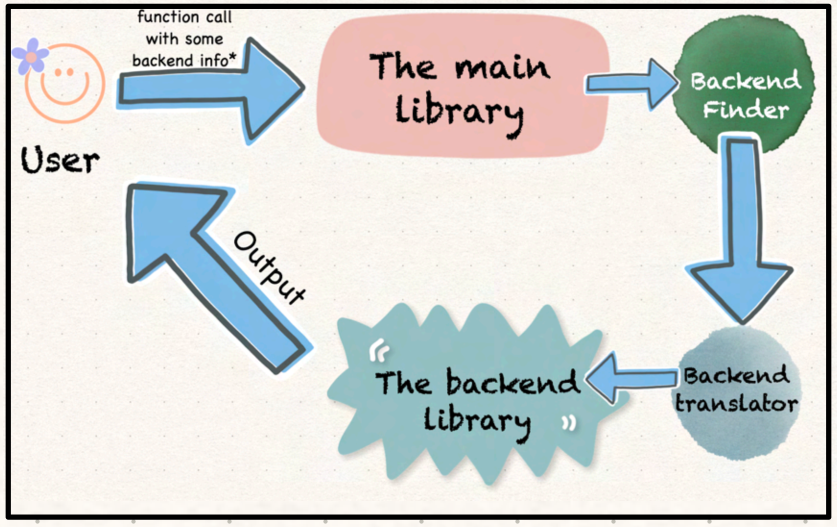
```
nx.betweenness_centrality(G, backend="parallel")
```

2. Type-based dispatching

```
import nx_parallel as nxp
H = nxp.ParallelGraph(nx.complete_graph(3))
nx.betweenness_centrality(H)
```

```
class ParallelGraph:
    __networkx_backend__ = "parallel"

    def __init__(self, ...):
        ....
```

Function call with "some backend information" in NetworkX



1. `backend` kwarg in the function call:

```
nx.betweenness_centrality(G, backend="parallel")
```

2. Type-based dispatching

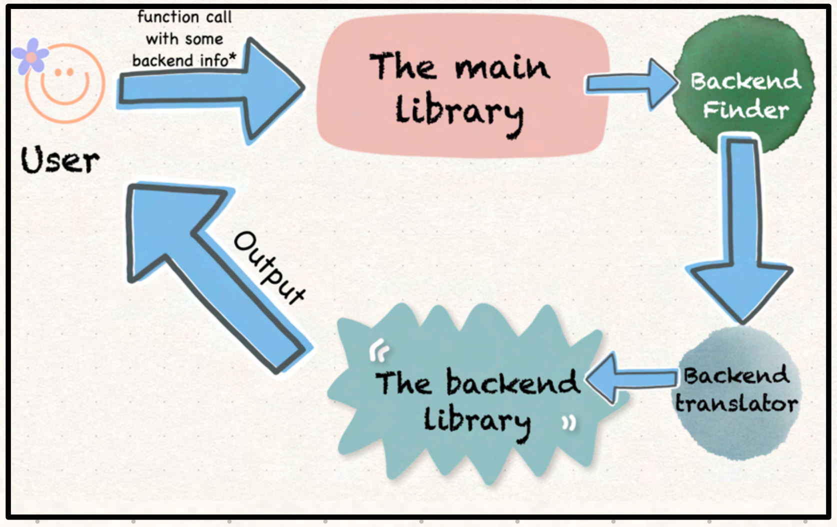
```
nx.betweenness_centrality(H)
```

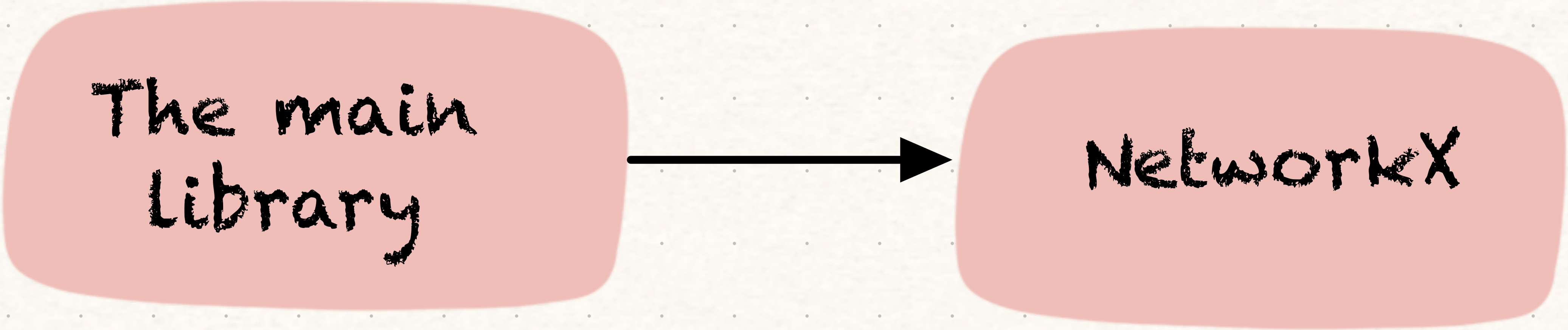
3. Using networkx's configurations

```
with nx.config(backend_priority=['parallel',]):
    nx.square_clustering(G)
```

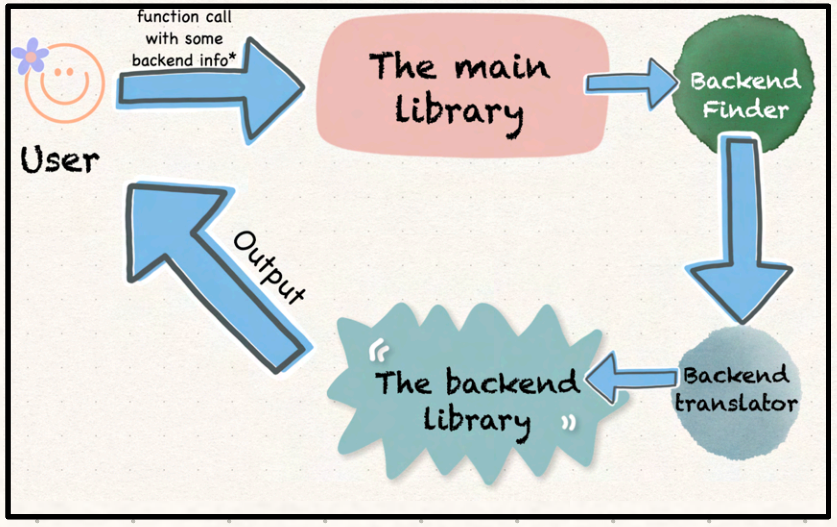
`@nx._dispatchable` decorator's job:

To identify "some backend info" and accordingly redirect the call to the specified backend's implementation.


User
function call
with some
backend info*
The main
library
Backend
Finder
Backend
translator
"The backend
library"
Output

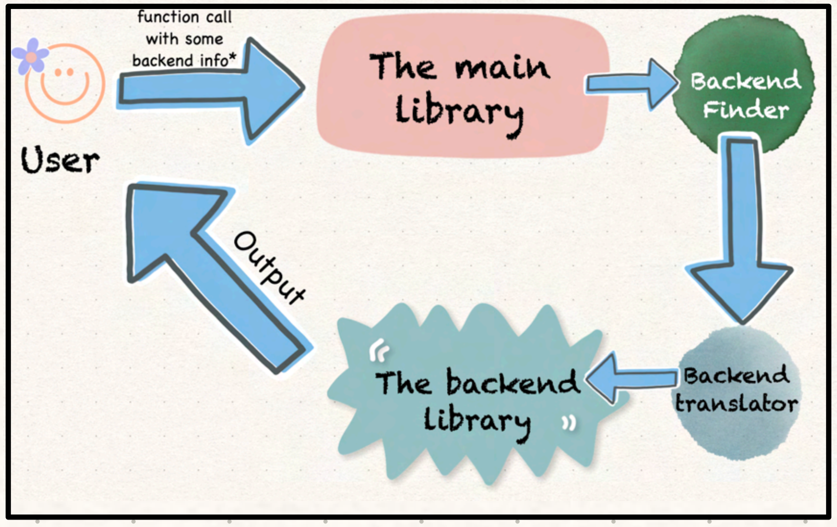

The main
Library
NetworkX

Backend Finder in NetworkX

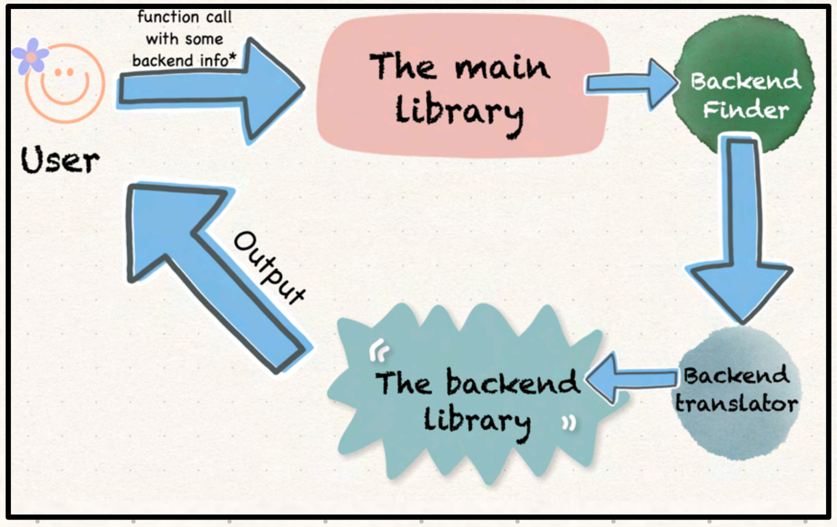


Backend Finder in NetworkX

NetworkX discovers backend packages by loading the `networkx.backends` Python entry_point



Backend Finder in NetworkX



NetworkX discovers backend packages by loading the `networkx.backends` *Python entry_point*

What is `networkx.backends`*?*

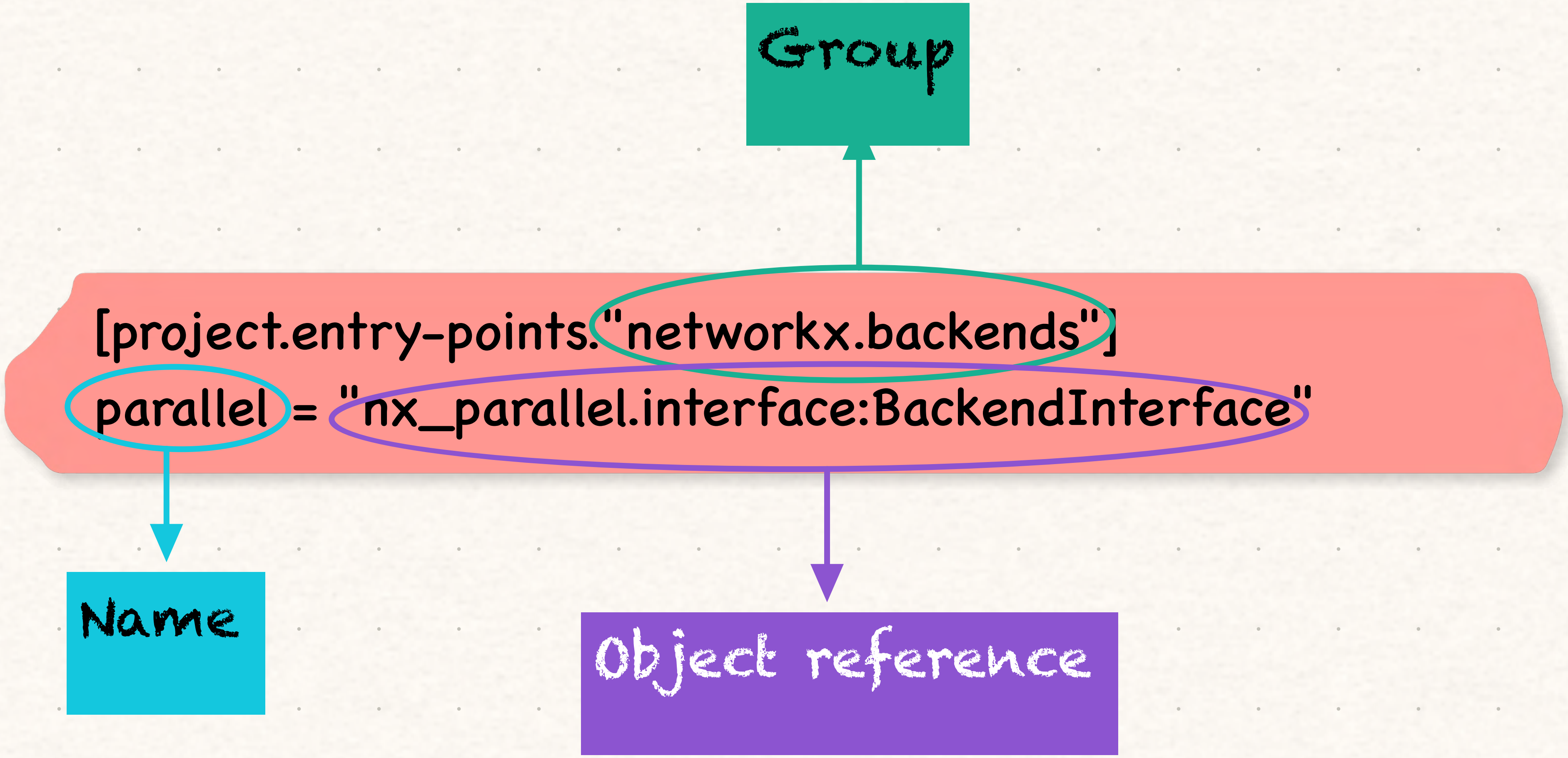
what is an entry_point?

NetworkX's entry_points

```
[project.entry-points."networkx.backends"]
parallel = "nx_parallel.interface:BackendInterface"
```

```
[project.entry-points."networkx.backend_info"]
parallel = "_nx_parallel:get_info"
```

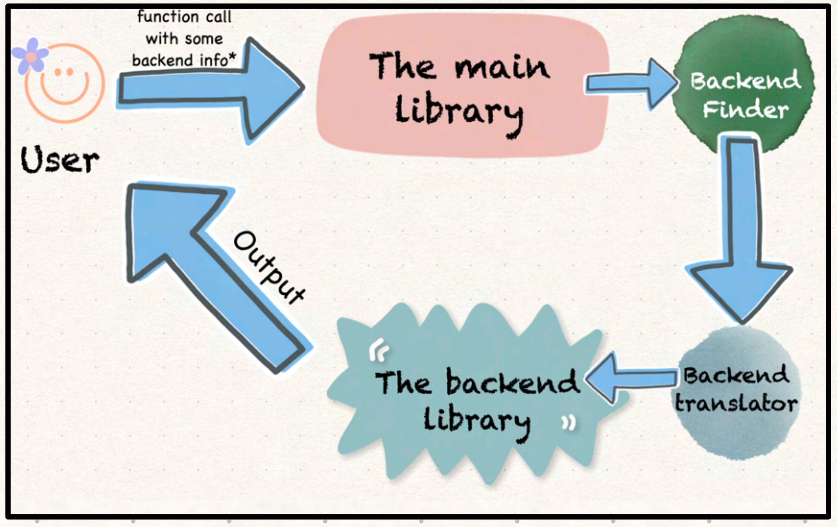
Three required properties to define a Python entry_point:



```
[project.entry-points."networkx.backends"]
parallel = "nx_parallel.interface:BackendInterface"
```

ref. https://packaging.python.org/en/latest/specifications/entry-points/

Backend Finder in NetworkX

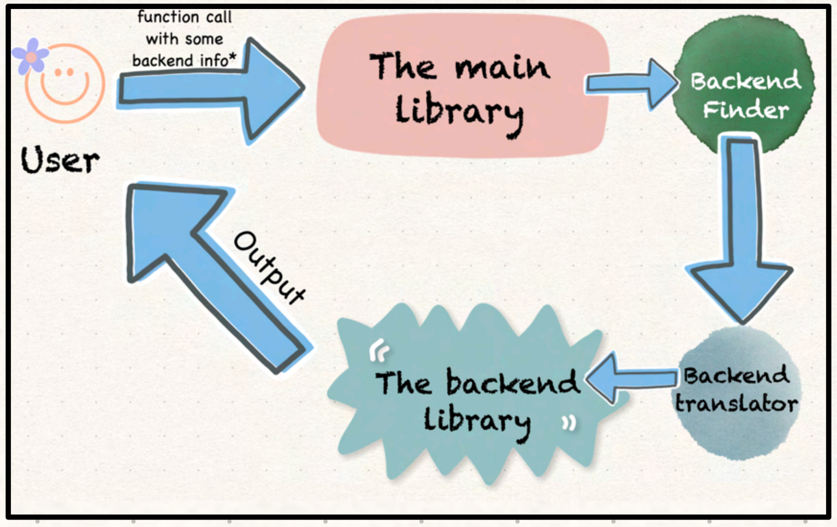


NetworkX discovers backend packages by loading the `networkx.backends` Python entry_point

Loading? -

```
>>> from importlib.metadata import entry_points
>>> entry_points(group="networkx.backends")
(EntryPoint(name='parallel',
value='nx_parallel.interface:BackendInterface',
group='networkx.backends'),
EntryPoint(name='nx_loopback',
value='networkx.classes.tests.dispatch_interface:bac
kend_interface', group='networkx.backends'))
```

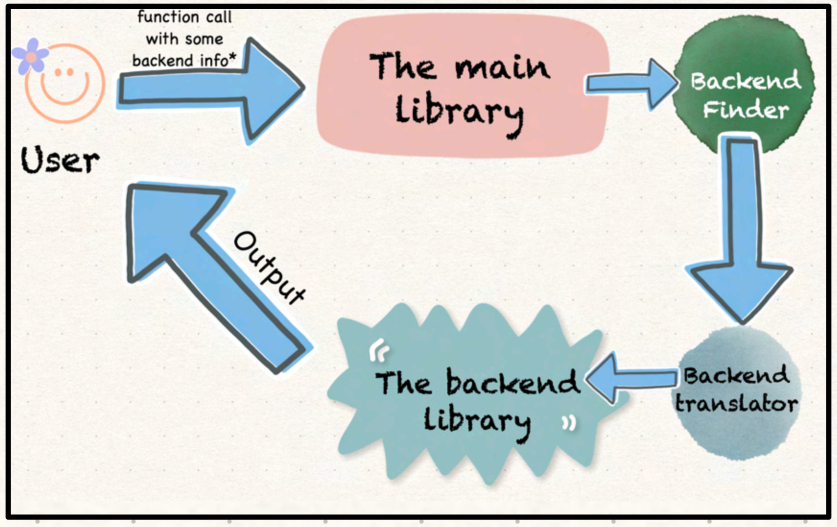
Backend translator in NetworkX



To convert args in the function call into something a backend can understand we use `convert_from_nx` *and* `convert_to_nx` *(which are the attributes of the object referenced in the "networkx.backends" entry_point by the backend)*

`convert_from_nx` *and* `convert_to_nx` are just for args that are Graph objects.

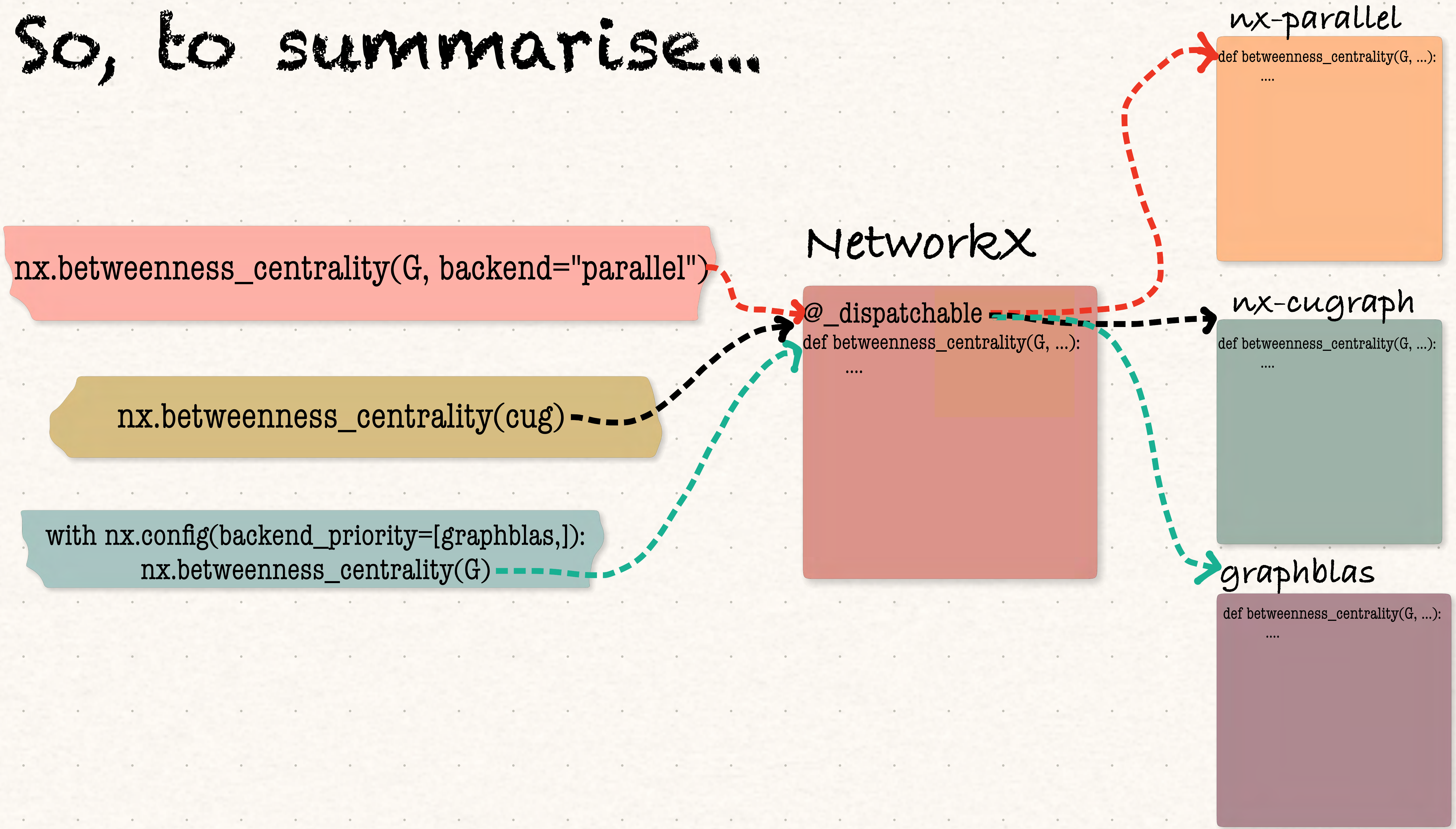
“The backend library” in NetworkX



What makes a networkx backend?

`networkx.backends` entry_point referring to a `BackendInterface` object with attributes - `convert_from_nx`, `convert_to_nx`, and all the algorithms implemented in the backend. And a backend graph object with `__networkx_backend__` attribute.

So, to summarise...



```
nx.betweenness_centrality(G, backend="parallel")
```

```
nx.betweenness_centrality(cug)
```

```
with nx.config(backend_priority=[graphblas,]):
    nx.betweenness_centrality(G)
```

NetworkX

```
@_dispatchable
def betweenness_centrality(G, ...):
    ....
```

nx-parallel

```
def betweenness_centrality(G, ...):
    ....
```

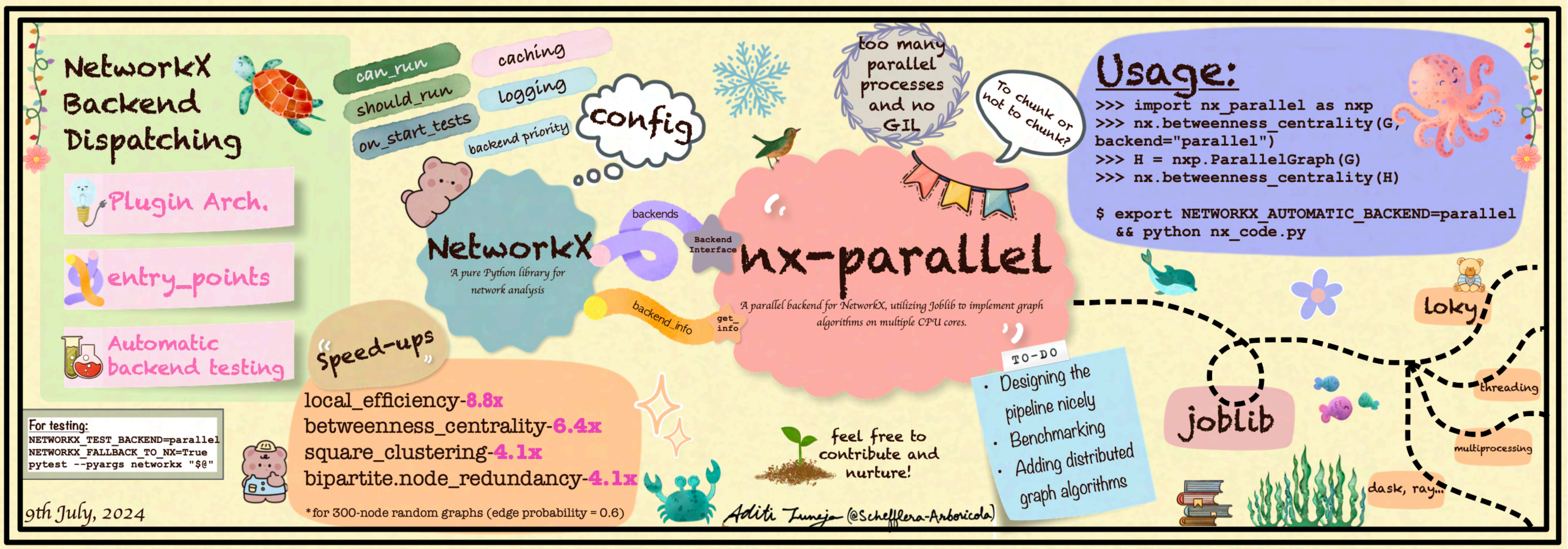
nx-cugraph

```
def betweenness_centrality(G, ...):
    ....
```

graphblas

```
def betweenness_centrality(G, ...):
    ....
```

Going back to the poster...



NetworkX Backend Dispatching

- Plugin Arch.
- entry_points
- Automatic backend testing

can_run
should_run
on_start_tests
caching
logging
backend priority
config

For testing:

```
NETWORKX_TEST_BACKEND=parallel
NETWORKX_FALLBACK_TO_NX=True
pytest --pyargs networkx "$@"
```

NetworkX

A pure Python library for network analysis

backends
Backend Interface
backend_info
get_info

too many parallel processes and no GIL

To chunk or not to chunk?

"nx-parallel"

A parallel backend for NetworkX, utilizing Joblib to implement graph algorithms on multiple CPU cores.

Usage:

```
>>> import nx_parallel as nxp
>>> nx.betweenness_centrality(G,
backend="parallel")
>>> H = nxp.ParallelGraph(G)
>>> nx.betweenness_centrality(H)

$ export NETWORKX_AUTOMATIC_BACKEND=parallel
  && python nx_code.py
```

"Speed-ups"

local_efficiency-**8.8x**
betweenness_centrality-**6.4x**
square_clustering-**4.1x**
bipartite.node_redundancy-**4.1x**

*for 300-node random graphs (edge probability = 0.6)

feel free to contribute and nurture!

TO-DO

- Designing the pipeline nicely
- Benchmarking
- Adding distributed graph algorithms

joblib
loky
threading
multiprocessing
dask, ray...

9th July, 2024

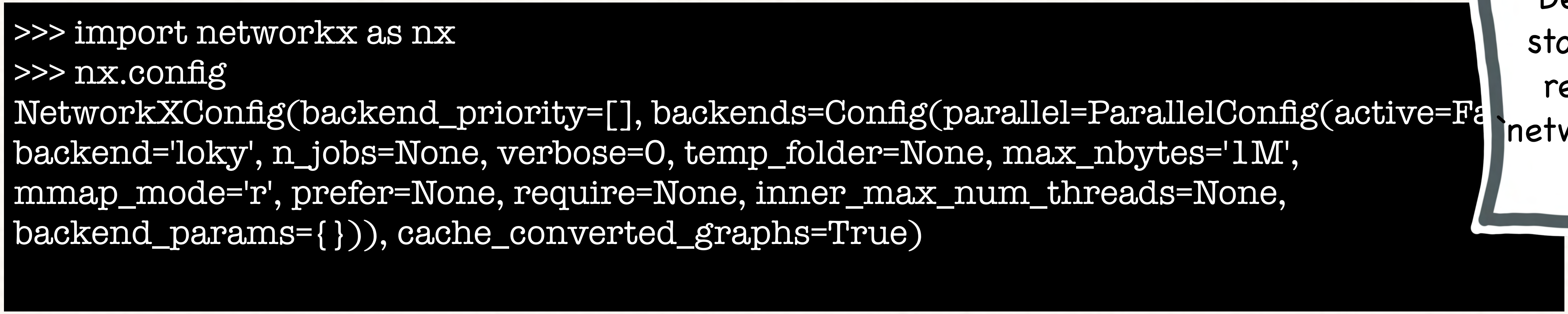
Aditi Juneja (@Schefflera-Arboricola)

Understanding NetworkX's API Dispatching with a parallel backend

Configs in NetworkX

```
>>> import networkx as nx
>>> nx.config
NetworkXConfig(backend_priority=[], backends=Config(parallel=ParallelConfig(active=False,
backend='loky', n_jobs=None, verbose=0, temp_folder=None, max_nbytes='1M',
mmap_mode='r', prefer=None, require=None, inner_max_num_threads=None,
backend_params={ })), cache_converted_graphs=True)
```

Configs in NetworkX



```
>>> import networkx as nx
>>> nx.config
NetworkXConfig(backend_priority=[], backends=Config(parallel=ParallelConfig(active=Fa
backend='loky', n_jobs=None, verbose=0, temp_folder=None, max_nbytes='1M',
mmap_mode='r', prefer=None, require=None, inner_max_num_threads=None,
backend_params={})), cache_converted_graphs=True)
```

Default configs are stored in the object referenced by the `networkx.backend_info` entry_point

A standard nx-parallel algorithm

(a few days ago)

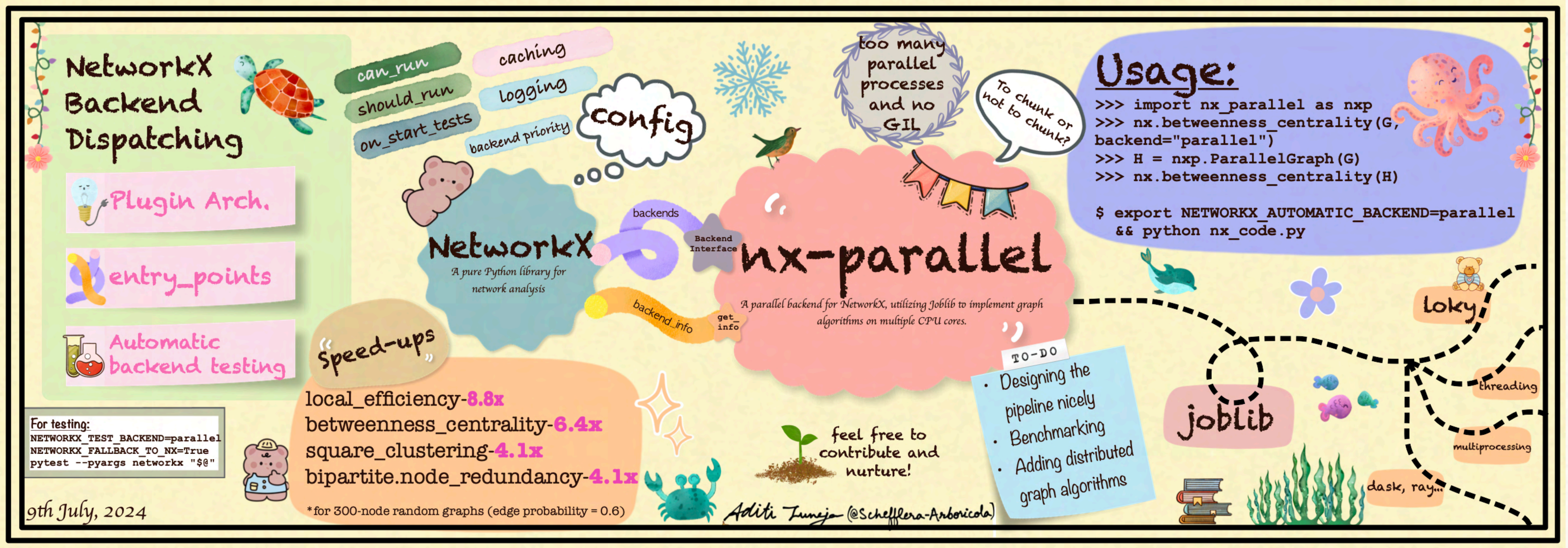
```
def xyz(G,..., get_chunks="chunks"):
     chunks = chunk(G)
    results = joblib.Parallel(n_jobs=-1)(
            joblib.delayed(chunks)
      )
    result = combine(results)
    return result
```

A standard nx-parallel algo

(a few days ago)

```
def xyz(G,..., get_chunks="chunks")
    chunks = chunk(G)
   results = joblib.Parallel(n_jobs=-1)(
            joblib.delayed(chunks)
     )
   result = combine(results)
   return result
```

allowing users
to modify
these
parameters



```
>>> import nx_parallel as nxp
>>> nx.betweenness_centrality(G,
backend="parallel")
>>> H = nxp.ParallelGraph(G)
>>> nx.betweenness_centrality(H)

$ export NETWORKX_AUTOMATIC_BACKEND=parallel
  && python nx_code.py
```

```
NETWORKX_TEST_BACKEND=parallel
NETWORKX_FALLBACK_TO_NX=True
pytest --pyargs networkx "$@"
```

Dashed line --> Solid line

Configuring nx-parallel

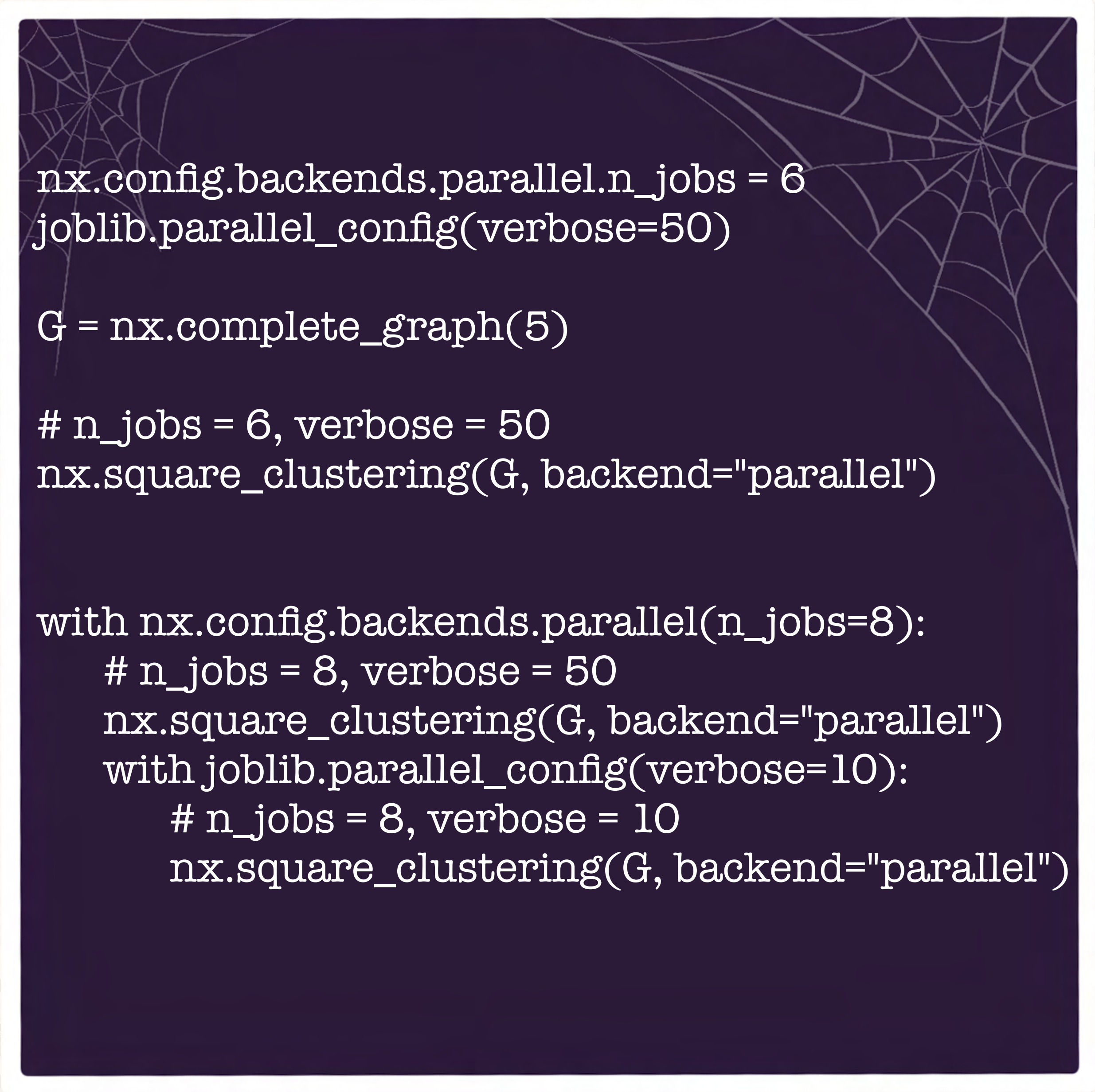
Joblib

```python
joblib.parallel_config(n_jobs = 6, verbose=50)

G = nx.complete_graph(5)

# n_jobs = 6, verbose = 50
nx.square_clustering(G, backend="parallel")


with joblib.parallel_config(n_jobs=8):
    # n_jobs = 8, verbose = 50
    nx.square_clustering(G, backend="parallel")
```

NetworkX

```python
nx.config.backends.parallel.active = True
nx.config.backends.parallel.n_jobs = 6
nx.config.backends.parallel.verbose=50

G = nx.complete_graph(5)

# n_jobs = 6, verbose = 50
nx.square_clustering(G, backend="parallel")


with nx.config.backends.parallel(n_jobs=8):
    # n_jobs = 8, verbose = 50
    nx.square_clustering(G, backend="parallel")
```

Syncing the two?



```
nx.config.backends.parallel.n_jobs = 6
joblib.parallel_config(verbose=50)

G = nx.complete_graph(5)

# n_jobs = 6, verbose = 50
nx.square_clustering(G, backend="parallel")


with nx.config.backends.parallel(n_jobs=8):
    # n_jobs = 8, verbose = 50
    nx.square_clustering(G, backend="parallel")
    with joblib.parallel_config(verbose=10):
        # n_jobs = 8, verbose = 10
        nx.square_clustering(G, backend="parallel")
```

ref. https://github.com/networkx/nx-parallel/issues/76 for more.

Dispatching **evolves** an API... to be exactly the same!

- "Double down" on the strength of a library
  - **API:** becomes *the* API, avoids fragmentation
  - **Community:** grow a community of backends
- Enable new capabilities
  - Scale performance and data
  - Allow specialization and customization
  - New possibilities, new use cases
- All without fundamentally changing a library
  - Keep what works and what makes it good!

NetworkX
Network Analysis in Python

`nx-parallel`

`nx-pandas`

`nx-cugraph`

`nx-arangodb`

`graphblas-algorithms`

`<private>`

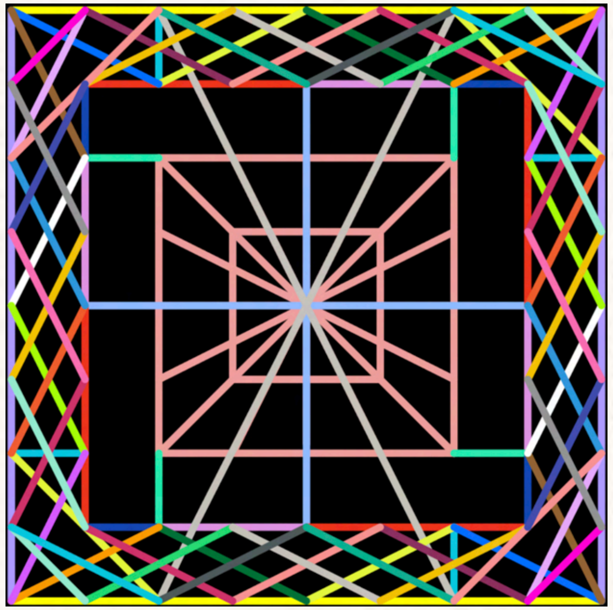
`nx-???`
`(what next?)`

Coming soon:
spatch to generalize and document dispatching
https://github.com/scientific-python/spatch

Join us:
NetworkX dispatching call
https://scientific-python.org/calendars

Erik Welch

“ Thank you :) ”



Aditi Juneja (Mostly Schefflera Arboricola)