Hello World!

Here's an example notebook with some documentation on how to access CMIP data.

```
/glade/work/mclong/miniconda3/envs/CMIP6-201910/lib/python3.7/site-packages/intake/source/discover
y.py:136: FutureWarning: The drivers ['stac-catalog', 'stac-collection', 'stac-item'] do not speci
fy entry_points and were only discovered via a package scan. This may break in a future release of
intake. The packages should be updated.
  FutureWarning)
```

```
hello world!
```

Demonstrate how to use `intake-esm`

Intake-esm is a data cataloging utility that facilitates access to CMIP data. It's pretty awesome.

An `intake-esm` collection object establishes a link to a database that contains file locations and associated metadata (i.e. which experiement, model, etc. thet come from).

Opening a collection

First step is to open a collection by pointing to the collection definition file, which is a JSON file that conforms to the ESM Collection Specification.

The collection JSON files are stored locally in this repository for purposes of reproducibility---and because Cheyenne compute nodes don't have Internet access.

The primary source for these files is the intake-esm-datastore repository. Any changes made to these files should be pulled from that repo. For instance, the Pangeo cloud collection is available here.

```
glade-cmip6-ESM Collection with 590735 entries:
	> 10 activity_id(s)

	> 21 institution_id(s)

	> 38 source_id(s)

	> 60 experiment_id(s)

	> 161 member_id(s)

	> 34 table_id(s)

	> 1022 variable_id(s)

	> 11 grid_label(s)

	> 59 dcpp_init_year(s)

	> 222 version(s)

	> 4275 time_range(s)

	> 590735 path(s)
```

`intake-esm` is build on top of [pandas]. It is possible to view the `pandas.DataFrame` as follows.

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	dcpp_init_year	versio
0	AerChemMIP	BCC	BCC-ESM1	ssp370	r2i1p1f1	Amon	hfls	gn	NaN	v2019062
1	AerChemMIP	BCC	BCC-ESM1	ssp370	r2i1p1f1	Amon	va	gn	NaN	v2019062
2	AerChemMIP	BCC	BCC-ESM1	ssp370	r2i1p1f1	Amon	tas	gn	NaN	v2019062
3	AerChemMIP	BCC	BCC-ESM1	ssp370	r2i1p1f1	Amon	rsds	gn	NaN	v2019062
4	AerChemMIP	BCC	BCC-ESM1	ssp370	r2i1p1f1	Amon	pr	gn	NaN	v2019062

It is possible to interact with the `DataFrame`; for instance, we can see what the "attributes" of the datasets are by printing the columns.

```
Index(['activity_id', 'institution_id', 'source_id', 'experiment_id',
       'member_id', 'table_id', 'variable_id', 'grid_label', 'dcpp_init_year',
       'version', 'time_range', 'path'],
      dtype='object')
```

Search and discovery

Finding unique entries

Let's query the data to see what models ("source_id"), experiments ("experiment_id") and temporal frequencies ("table_id") are available.

```
{'experiment_id': {'count': 60,
                   'values': ['ssp370', 'histSST-piNTCF', 'histSST',
                              'histSST-1950HC', 'hist-1950HC', 'hist-piNTCF',
                              'piClim-NTCF', 'ssp370SST-lowNTCF',
                              'ssp370-lowNTCF', 'ssp370SST', 'amip-future4K',
                              'amip-m4K', 'a4SST', 'aqua-p4K', 'piSST',
                              'amip-4xCO2', 'a4SSTice', 'amip-p4K',
                              'aqua-control', 'aqua-4xCO2', 'abrupt-4xCO2',
                              'historical', 'piControl', 'amip', '1pctCO2',
                              'esm-hist', 'esm-piControl', 'ssp245', 'ssp585',
                              'ssp126', 'dcppA-hindcast',
                              'dcppC-hindcast-noPinatubo',
                              'dcppC-hindcast-noElChichon', 'dcppA-assim',
                              'dcppC-hindcast-noAgung', 'highresSST-present',
                              'land-hist-princeton', 'land-hist-cruNcep',
                              'land-hist', 'deforest-globe',
                              'esm-ssp585-ssp126Lu', 'land-cCO2', 'hist-noLu',
                              'land-crop-noFert', 'ssp370-ssp126Lu',
                              'ssp126-ssp370Lu', 'land-noLu', 'land-noFire',
                              'land-hist-altStartYear', 'land-cClim', 'omip1',
                              'pdSST-piAntSIC', 'pdSST-futAntSIC',
                              'pdSST-pdSIC', 'pdSST-piArcSIC',
                              'pdSST-futArcSIC', 'ssp119', 'ssp434', 'ssp460',
                              'ssp534-over']},
 'source_id': {'count': 38,
               'values': ['BCC-ESM1', 'CESM2-WACCM', 'CESM2', 'CNRM-CM6-1',
                          'CNRM-ESM2-1', 'BCC-CSM2-MR', 'FGOALS-f3-L',
                          'FGOALS-g3', 'SAM0-UNICON', 'UKESM1-0-LL',
```

```
                              'HadGEM3-GC31-LL', 'HadGEM3-GC31-MM', 'AWI-CM-1-1-MR',
                              'GFDL-AM4', 'GFDL-ESM4', 'GFDL-CM4', 'GISS-E2-1-H',
                              'GISS-E2-1-G', 'CanESM5', 'E3SM-1-0', 'CAMS-CSM1-0',
                              'MCM-UA-1-0', 'EC-Earth3-LR', 'EC-Earth3',
                              'EC-Earth3-Veg', 'MRI-ESM2-0', 'NESM3', 'MIROC-ES2L',
                              'MIROC6', 'IPSL-CM6A-LR', 'NorCPM1', 'NorESM2-LM',
                              'FIO-ESM-2-0', 'NICAM16-7S', 'NICAM16-8S',
                              'NICAM16-9S', 'IPSL-CM6A-ATM-HR', 'MPI-ESM1-2-HR']},
 'table_id': {'count': 34,
              'values': ['Amon', 'AERmonZ', 'CFmon', 'day', 'EdayZ', 'Eday',
                         'CFday', 'EmonZ', 'AERday', 'Emon', 'fx', 'Lmon',
                         'AERmon', 'Ofx', 'Omon', 'SImon', 'Oyr', 'AERhr',
                         'Eyr', 'LImon', 'SIday', '6hrPlev', 'CFsubhr',
                         '6hrLev', 'Oday', 'ImonGre', 'ImonAnt', 'Efx',
                         'IfxGre', '3hr', '6hrPlevPt', 'E1hr', 'E3hr',
                         'CF3hr']}}
```


Searching for specific datasets

Let's find all the dissolved oxygen data at annual frequency from the ocean for the `historical` and `ssp585` experiments.

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	dcpp_init_year	v
42454	CMIP	NCAR	CESM2-WACCM	historical	r2i1p1f1	Oyr	o2	gn	NaN	v201
44704	CMIP	NCAR	CESM2-WACCM	historical	r1i1p1f1	Oyr	o2	gn	NaN	v201
46954	CMIP	NCAR	CESM2-WACCM	historical	r3i1p1f1	Oyr	o2	gn	NaN	v201
263234	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Oyr	o2	gn	NaN	v201
263717	CMIP	CCCma	CanESM5	historical	r5i1p1f1	Oyr	o2	gn	NaN	v201
...	...	...	...	...	...	...	...	...	...	
560932	ScenarioMIP	DKRZ	MPI-ESM1-2-HR	ssp585	r1i1p1f1	Oyr	o2	gn	NaN	v201
560933	ScenarioMIP	DKRZ	MPI-ESM1-2-HR	ssp585	r1i1p1f1	Oyr	o2	gn	NaN	v201
560934	ScenarioMIP	DKRZ	MPI-ESM1-2-HR	ssp585	r1i1p1f1	Oyr	o2	gn	NaN	v201
589961	ScenarioMIP	MIROC	MIROC-ES2L	ssp585	r1i1p1f2	Oyr	o2	gn	NaN	v201
590561	ScenarioMIP	IPSL	IPSL-CM6A-LR	ssp585	r1i1p1f1	Oyr	o2	gn	NaN	v201

93 rows × 12 columns

It might be desirable to get more specific. For instance, we may want to select only the models that have *both* `historical` and `ssp585` data. We coud do this as follows.

```
['CanESM5', 'IPSL-CM6A-LR', 'MIROC-ES2L']
```

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	dcpp_init_year	v
263234	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Oyr	o2	gn	NaN	v201
263717	CMIP	CCCma	CanESM5	historical	r5i1p1f1	Oyr	o2	gn	NaN	v201
264208	CMIP	CCCma	CanESM5	historical	r12i1p1f1	Oyr	o2	gn	NaN	v201
264702	CMIP	CCCma	CanESM5	historical	r1i1p2f1	Oyr	o2	gn	NaN	v201
265737	CMIP	CCCma	CanESM5	historical	r14i1p1f1	Oyr	o2	gn	NaN	v201
...	...	...	...	...	...	...	...	...	...	
557211	ScenarioMIP	CCCma	CanESM5	ssp585	r10i1p1f1	Oyr	o2	gn	NaN	v201
557346	ScenarioMIP	CCCma	CanESM5	ssp585	r7i1p1f1	Oyr	o2	gn	NaN	v201
557477	ScenarioMIP	CCCma	CanESM5	ssp585	r6i1p1f1	Oyr	o2	gn	NaN	v201
589961	ScenarioMIP	MIROC	MIROC-ES2L	ssp585	r1i1p1f2	Oyr	o2	gn	NaN	v201
590561	ScenarioMIP	IPSL	IPSL-CM6A-LR	ssp585	r1i1p1f1	Oyr	o2	gn	NaN	v201

62 rows × 12 columns

Loading data

`intake-esm` enables loading data directly into an xarray.Dataset.

Note that data on the cloud are in zarr format and data on glade are stored as netCDF files. This is opaque to the user.

`intake-esm` has rules for aggegating datasets; these rules are defined in the collection-specification file.

```
xarray will load netCDF datasets with dask using a single chunk for all arrays.
                     For effective chunking, please provide chunks in cdf_kwargs.
                     For example: cdf_kwargs={'chunks': {'time': 36}}

--> The keys in the returned dictionary of datasets are constructed as follows:
        'activity_id.institution_id.source_id.experiment_id.table_id.grid_label'

--> There will be 6 group(s)
```

`dset_dict` is a dictionary of `xarray.Dataset`'s; its keys are constructed to refer to compatible groups.

```
dict_keys(['CMIP.CCCma.CanESM5.historical.Oyr.gn', 'CMIP.IPSL.IPSL-CM6A-LR.historical.Oyr.gn', 'CM
IP.MIROC.MIROC-ES2L.historical.Oyr.gn', 'ScenarioMIP.CCCma.CanESM5.ssp585.Oyr.gn', 'ScenarioMIP.IP
SL.IPSL-CM6A-LR.ssp585.Oyr.gn', 'ScenarioMIP.MIROC.MIROC-ES2L.ssp585.Oyr.gn'])
```

We can access a particular dataset as follows.

```
<xarray.Dataset>
Dimensions:              (bnds: 2, i: 360, j: 291, lev: 45, member_id: 20, time: 165, vertices: 4)
Coordinates:
  * i                    (i) int32 0 1 2 3 4 5 6 ... 353 354 355 356 357 358 359
  * lev                  (lev) float64 3.047 9.454 16.36 ... 5.375e+03 5.625e+03
  * time                 (time) float64 182.5 547.5 912.5 ... 5.968e+04 6.004e+04
  * j                    (j) int32 0 1 2 3 4 5 6 ... 284 285 286 287 288 289 290
  * member_id            (member_id) <U9 'r12i1p1f1' 'r14i1p1f1' ... 'r9i1p1f1'
Dimensions without coordinates: bnds, vertices
Data variables:
    vertices_longitude   (j, i, vertices) float64 74.0 74.0 73.0 ... 72.95 73.0
```

```
    vertices_latitude    (j, i, vertices) float64 -78.29 -78.49 ... 50.11 50.11
    time_bnds            (time, bnds) float64 dask.array<chunksize=(165, 2), meta=np.ndarray>
    longitude            (j, i) float64 73.5 74.5 75.5 76.5 ... 72.95 72.96 72.99
    latitude             (j, i) float64 -78.39 -78.39 -78.39 ... 50.23 50.01
    lev_bnds             (lev, bnds) float64 0.0 6.194 6.194 ... 5.5e+03 5.75e+03
    o2                   (member_id, time, lev, j, i) float32 dask.array<chunksize=(1, 165, 45, 29
1, 360), meta=np.ndarray>
Attributes:
    data_specs_version:          01.00.29
    forcing_index:               1
    grid_label:                  gn
    cmor_version:                3.4.0
    parent_experiment_id:        piControl
    table_info:                  Creation Date:(20 February 2019) MD5:374fbe5a...
    institution_id:              CCCma
    source_id:                   CanESM5
    Conventions:                 CF-1.7 CMIP-6.2
    activity_id:                 CMIP
    experiment_id:               historical
    institution:                 Canadian Centre for Climate Modelling and Ana...
    parent_time_units:           days since 1850-01-01 0:0:0.0
    frequency:                   yr
    title:                       CanESM5 output prepared for CMIP6
    CCCma_pycmor_hash:           33c30511acc319a98240633965a04ca99c26427e
    references:                  Geophysical Model Development Special issue o...
    tracking_id:                 hdl:21.14100/b74e3b07-ed7b-43b3-976f-3b4a55c5...
    version:                     v20190429
    variable_id:                 o2
    branch_method:               Spin-up documentation
    history:                     2019-05-02T13:55:48Z ;rewrote data to be cons...
    parent_source_id:            CanESM5
    parent_activity_id:          CMIP
    parent_mip_era:              CMIP6
    CCCma_model_hash:            55f484f90aff0e32c5a8e92a42c6b9ae7ffe6224
    nominal_resolution:          100 km
    grid:                        ORCA1 tripolar grid, 1 deg with refinement to...
    initialization_index:        1
    mip_era:                     CMIP6
    sub_experiment:              none
    YMDH_branch_time_in_child:   1850:01:01:00
    experiment:                  all-forcing simulation of the recent past
    sub_experiment_id:           none
    source_type:                 AOGCM
    table_id:                    Oyr
    parent_variant_label:        r1i1p1f1
    contact:                     ec.cccma.info-info.ccmac.ec@canada.ca
    license:                     CMIP6 model data produced by The Government o...
    product:                     model-output
    realm:                       ocnBgchem
    CCCma_parent_runid:          rc3.1-pictrl
    external_variables:          areacello volcello
    branch_time_in_child:        0.0
    source:                      CanESM5 (2019): \naerosol: interactive\natmos...
    physics_index:               1
```

More advanced queries

As motivation for diving into more advanced manipulations with `intake-esm`, let's consider the task of getting access to grid information in the `Ofx` table_id.

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	dcpp_init_year	v
262740	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Ofx	areacello	gn	NaN	v201

262741	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Ofx	thkcello	gn	NaN	v201
263236	CMIP	CCCma	CanESM5	historical	r5i1p1f1	Ofx	thkcello	gn	NaN	v201
263718	CMIP	CCCma	CanESM5	historical	r12i1p1f1	Ofx	thkcello	gn	NaN	v201
264210	CMIP	CCCma	CanESM5	historical	r1i1p2f1	Ofx	areacello	gn	NaN	v201
...	...	...	...	...	...	...	...	...	...	
557356	ScenarioMIP	CCCma	CanESM5	ssp585	r18i1p1f1	Ofx	areacello	gn	NaN	v201
557374	ScenarioMIP	CCCma	CanESM5	ssp585	r11i1p1f1	Ofx	sftof	gn	NaN	v201
557384	ScenarioMIP	CCCma	CanESM5	ssp585	r6i1p1f1	Ofx	areacello	gn	NaN	v201
557385	ScenarioMIP	CCCma	CanESM5	ssp585	r6i1p1f1	Ofx	thkcello	gn	NaN	v201
590533	ScenarioMIP	IPSL	IPSL-CM6A-LR	ssp585	r1i1p1f1	Ofx	areacello	gn	NaN	v201

186 rows × 12 columns

This, however, comes with lots of redundant information.

Additionally, it may be necessary to do more targeted manipulations of the search. For instance, we've found a handful of corrupted files on `glade` and might need to work around loading these.

As an illustration of this, in the code below, we specify a list of to queries (in this case one) to eliminate.

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	dcpp_init_year	v
262740	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Ofx	areacello	gn	NaN	v201
262741	CMIP	CCCma	CanESM5	historical	r2i1p1f1	Ofx	thkcello	gn	NaN	v201
263236	CMIP	CCCma	CanESM5	historical	r5i1p1f1	Ofx	thkcello	gn	NaN	v201
263718	CMIP	CCCma	CanESM5	historical	r12i1p1f1	Ofx	thkcello	gn	NaN	v201
264210	CMIP	CCCma	CanESM5	historical	r1i1p2f1	Ofx	areacello	gn	NaN	v201
...	...	...	...	...	...	...	...	...	...	
557356	ScenarioMIP	CCCma	CanESM5	ssp585	r18i1p1f1	Ofx	areacello	gn	NaN	v201
557374	ScenarioMIP	CCCma	CanESM5	ssp585	r11i1p1f1	Ofx	sftof	gn	NaN	v201
557384	ScenarioMIP	CCCma	CanESM5	ssp585	r6i1p1f1	Ofx	areacello	gn	NaN	v201
557385	ScenarioMIP	CCCma	CanESM5	ssp585	r6i1p1f1	Ofx	thkcello	gn	NaN	v201
590533	ScenarioMIP	IPSL	IPSL-CM6A-LR	ssp585	r1i1p1f1	Ofx	areacello	gn	NaN	v201

185 rows × 12 columns

We then drop duplicates.

Now, since we've only retained one ensemble member, we need to eliminate that column. If we omit this step, `intake-esm` will throw an error, complaining that different variables are present for each ensemble member. Setting the `member_id` column to NaN precludes attempts to join along the ensemble dimension.

After this final manipulation, we copy the `DataFrame` back to the collection object and proceed with loading the data.

```
xarray will load netCDF datasets with dask using a single chunk for all arrays.
                     For effective chunking, please provide chunks in cdf_kwargs.
                     For example: cdf_kwargs={'chunks': {'time': 36}}

--> The keys in the returned dictionary of datasets are constructed as follows:
        'activity_id.institution_id.source_id.experiment_id.table_id.grid_label'

--> There will be 3 group(s)

dict_keys(['CMIP.CCCma.CanESM5.historical.Ofx.gn', 'CMIP.IPSL.IPSL-CM6A-LR.historical.Ofx.gn', 'CM
IP.MIROC.MIROC-ES2L.historical.Ofx.gn'])

Data variables:
    latitude             (j, i) float64 dask.array<chunksize=(291, 360), meta=np.ndarray>
    longitude            (j, i) float64 dask.array<chunksize=(291, 360), meta=np.ndarray>
    vertices_latitude    (j, i, vertices) float64 dask.array<chunksize=(291, 360, 4), meta=np.ndarr
ay>
    vertices_longitude   (j, i, vertices) float64 dask.array<chunksize=(291, 360, 4), meta=np.ndarr
ay>
    areacello            (j, i) float32 dask.array<chunksize=(291, 360), meta=np.ndarray>
    lev_bnds             (lev, bnds) float64 dask.array<chunksize=(45, 2), meta=np.ndarray>
    thkcello             (lev, j, i) float32 dask.array<chunksize=(45, 291, 360), meta=np.ndarray>
    type                 |S3 ...
    sftof                (j, i) float32 dask.array<chunksize=(291, 360), meta=np.ndarray>
Data variables:
    nav_lat          (y, x) float32 dask.array<chunksize=(332, 362), meta=np.ndarray>
    nav_lon          (y, x) float32 dask.array<chunksize=(332, 362), meta=np.ndarray>
    bounds_nav_lon   (y, x, nvertex) float32 dask.array<chunksize=(332, 362, 4), meta=np.ndarray>
    bounds_nav_lat   (y, x, nvertex) float32 dask.array<chunksize=(332, 362, 4), meta=np.ndarray>
    area             (y, x) float32 dask.array<chunksize=(332, 362), meta=np.ndarray>
    areacello        (y, x) float32 dask.array<chunksize=(332, 362), meta=np.ndarray>
    basin            (y, x) float32 dask.array<chunksize=(332, 362), meta=np.ndarray>
Data variables:
    y_bnds               (y, bnds) float64 dask.array<chunksize=(256, 2), meta=np.ndarray>
    x_bnds               (x, bnds) float64 dask.array<chunksize=(360, 2), meta=np.ndarray>
    latitude             (y, x) float32 dask.array<chunksize=(256, 360), meta=np.ndarray>
    longitude            (y, x) float32 dask.array<chunksize=(256, 360), meta=np.ndarray>
    vertices_latitude    (y, x, vertices) float32 dask.array<chunksize=(256, 360, 4), meta=np.ndarr
ay>
    vertices_longitude   (y, x, vertices) float32 dask.array<chunksize=(256, 360, 4), meta=np.ndarr
ay>
    areacello            (y, x) float32 dask.array<chunksize=(256, 360), meta=np.ndarray>
    type                 |S3 ...
    sftof                (y, x) float32 dask.array<chunksize=(256, 360), meta=np.ndarray>
```

Demonstrate how spin-up a dask cluster

If you expect to require Big Data capabilities, here's how you spin up a dask cluster using dask-jobqueue.

The syntax is different if on an NCAR machine versus the cloud.

```
/glade/work/mclong/miniconda3/envs/CMIP6-201910/lib/python3.7/site-packages/distributed/dashboard/
core.py:72: UserWarning:
Port 8787 is already in use.
Perhaps you already have a cluster running?
Hosting the diagnostics dashboard on a random port instead.
  warnings.warn("\n" + msg)
VBox(children=(HTML(value='<h2>NCARCluster</h2>'), HBox(children=(HTML(value='\n<div>\n  <style sc
oped>\n    ...
```

Client

Scheduler: tcp://128.117.181.208:32844
Dashboard: https://jupyterhub.ucar.edu/dav/user/mclong/proxy/34037/status

Cluster

Workers: 0
Cores: 0
Memory: 0 B