

RSM8431: Introduction to Computer Science with Python – Session 6/Opt-Out Assignment

Introduction to Computer Science with Python is a course intended as an introduction for students that do not have a great deal of experience writing code in Python. We expect that students in the Master of Management Analytics program are comfortable writing scripts that are logical and well-structured. Having these skills will set you up for success in the other courses in the program. This course will teach fundamental Python skills from the ground up, and as a result, the material may be too much of a review for some. **If you feel as though you are comfortable writing code and have a programming background you can opt-out of the sessions by just doing the [opt-out assignment](#)** (aka the Session 6 assignment).

An essential skill for any developer is to be able to understand and utilize libraries written by other developers. The purpose of this assignment is to determine whether you can explore a relatively unfamiliar API developed by another developer and can apply your programming skills to it by constructing some useful helper functions and modules built around the API.

Any seasoned developer, when given well-documented libraries and help, should be able to utilize code written by others. In this assignment, we are going to delve into using Jupyter notebooks to perform GIS automation and spatial tasks. The first part of this exercise will be an introduction (or re-acquaintance) of the Jupyter notebook environment, followed by an in-depth exploration of the ArcGIS API for Python. After exploring the API, you're going to construct a series of helper functions that could potentially be useful and make it easier to undertake the analysis of crime patterns around the City of Toronto and particularly around the University of Toronto campus. Whenever working with another vendor or developer's API, often these classes, modules and functions are packaged together in a manner that makes sense for the developer. More often than not, you will need to repackage them or leverage them in a way that works for you, in this instance, to make using the API more applicable to analyzing data. To make the API more applicable to this problem, you will write a series of helper functions that will assist the University with their spatial analysis of crime instances across the city and around the campus.

Helper functions are used to leverage capabilities of the API in multiple areas of your programming code. The ArcGIS for Python API is pretty low-level, in that, there are few convenient ways to connect to ArcGIS servers, perform analyses, summarize data, create maps, etc. The functions you create will make it easier to use other functions within the API, for example, maybe we want a function that connects to the ArcGIS Online GIS server with your UTORID credentials, enumerates the data available on that server, finds a specific layer by name, and lists/summarizes the fields within that layer. All of these require utilization of other lower-level functions, but because of helper functions that you write can be done in one line of code.

Exercise: Introduction to the ArcGIS API for Python

The following exercise will give some background on setting up your environment to run the ArcGIS API for Python along with give some basic examples of how to use some of the basic functions. For those of you that have Anaconda already setup on your system, you can skip this section of the assignment except make sure that you open up your Anaconda Prompt and execute the bolded line below to install the ArcGIS API for Python. For those that do not have Anaconda/Jupyter notebooks yet the easiest way to setup Jupyter and the ArcGIS for Python API and all its dependencies is to use the open-source Anaconda Distribution. This is the industry standard for data scientists and is a complete package for doing machine learning on Windows (Linux, Mac, etc.).

The reason that it is important to have a distribution such as this is that sometimes dependencies can get out of sync – this helps to ensure that is less likely. For example, packages like NumPy and pandas have versions that can be compatible or incompatible with other libraries/packages on your system. By putting everything in one distribution this helps to minimize the changes that one library will break another and so on.

Here are some of the highlights of what this package provides/does:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop/train machine/deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

The package includes:



Of particular importance to what we're attempting to do is the Jupyter notebook element while the rest are just a bonus and used extensively in the data science space. Jupyter notebooks are used in many large corporations like Google, Microsoft and IBM for their design and how well-suited they are for demonstrations of programmatic and/or data science concepts and training.

To install Anaconda, visit the following website and click on your desired platform
<https://www.anaconda.com/products/individual> - in my case Windows. The website will attempt to choose your appropriate architecture automatically:

After downloading run the installer and just take all of the defaults and proceed with the full installation. The installation could take up to 15 minutes to complete because it will involve downloading a whole series of libraries onto your computer.

Once Anaconda is successfully setup on your system, now we'll install the ArcGIS for Python API. To do this, you can use the newly installed *Anaconda Prompt* (should now be in your Program menu, right-click and run it as administrator) – open this up and type the following into the command line window:

conda install -c esri arcgis

You can also visit the following website: <https://developers.arcgis.com/python/> to get the API as well but the conda method is far easier and will be integrated with the Jupyter notebook environment that we're going to use to demonstrate and explore the capabilities of the API. Once you type the above you should see the following window appear (or something similar):

```
The following NEW packages will be INSTALLED:

arcgis             esri/win-64::arcgis-1.9.0-py38_2172
blinker            pkgs/main/win-64::blinker-1.4-py38haa95532_0
ntlm-auth          esri/noarch::ntlm-auth-1.4.0-py_0
oauthlib           pkgs/main/noarch::oauthlib-3.1.1-pyhd3eb1b0_0
pyjwt              pkgs/main/win-64::pyjwt-2.1.0-py38haa95532_0
pyspho             pkgs/main/noarch::pyspho-2.1.3-pyhd3eb1b0_0
python-certifi-win esri/win-64::python-certifi-win32-1.2-py38_0
pywin32-security  esri/win-64::pywin32-security-228-py38_3
requests-kerberos  esri/win-64::requests-kerberos-0.12.0-0
requests-negotiat~ esri/win-64::requests-negotiate-sspi-0.5.2-py38_1
requests-oauthlib  pkgs/main/noarch::requests-oauthlib-1.3.0-py_0
requests-toolbelt  pkgs/main/noarch::requests-toolbelt-0.9.1-py_0
requests_ntlm      esri/noarch::requests_ntlm-1.1.0-py_0
winkerberos        esri/win-64::winkerberos-0.7.0-py38_0

The following packages will be UPDATED:

conda              4.10.1-py38haa95532_1 --> 4.10.3-py38haa95532_1

Proceed ([y]/n)? y
```

You will be prompted to confirm the installation, type Y and enter to continue. Conda will then go off onto the Internet to grab the latest version of the API and any dependencies that it requires and install them.

Using Jupyter Notebooks

If you're familiar with Jupyter notebooks then please feel free to skip to the next section of the assignment. If you're still in an Anaconda prompt you can simply change to a directory either where you have Jupyter notebooks stored (chances are you probably don't have any yet) or where you want to have them created. I made a folder on my workstation called C:\TEMP\notebooks so I'm going to change to that directory by typing (in my Anaconda prompt). Feel free to direct the notebook to wherever you store your files:

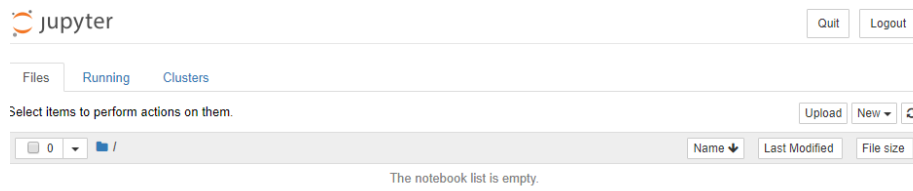
cd \temp\notebooks

Enter the following at the prompt to start jupyter:

jupyter notebook

By default, the jupyter notebook runs from the current working directory (the location from which you executed the jupyter notebook executable). So, in our example above, when you launch it, it will be reading from C:\TEMP. Depending on the speed of your computer this can take a few seconds to load up and you may be asked which browser you want to use – I always suggest running in Chrome as they seem

to work well in that browser, but technically any browser should suffice realistically. You should then get a browser window popped up that looks like the following – let’s take a minute to explore the Jupyter notebook structure and familiarize ourselves with what we’re looking at here



If we had notebooks located in our current path they would be displayed here, but since we don’t we just see an empty list. This is the window that we can use to open up existing notebooks. Let’s do a quick tour of the Jupyter dashboard as soon as you load the notebook. The Jupyter dashboard opens each file and directory has a checkbox next to it by clicking in and taking an item you could manipulate the selected object allowing you to do things like duplicate or shut down a running script. In addition, you can rename and delete folders the selection menu allows you to select all the files in the console the same type. By expanding this button for example, you could mark all the folders or all the running files as long as a file is loaded. Jupyter will call it a running file. You can have as many running files as you like and once loaded a file is considered a running file. Until you do not specifically shut it down.

The structure about directory/file management is the same as the one of an operating system. Files can be grouped into folders and folders can contain other folders from the upload button in the top right corner. You can upload a notebook into the directory you are in the standard Explorer box opens and when you select a file you can click on open and it will immediately appear in your directory.

Finally, you can expand the New button from the list that falls. You will most likely need to create a new text file a new folder or a new notebook file a notebook file can contain code in any of the languages in your notebooks section when you create a new Python notebook file. It will be recorded in the Python notebook format.

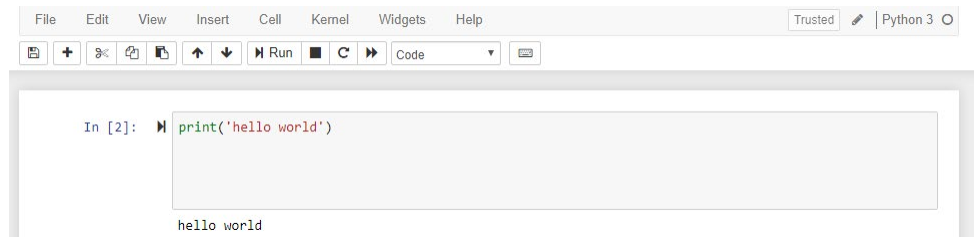
Let’s create a new notebook file by expanding the New button and selecting the default (a Python 3 notebook). Other Python formats may show up in your list.

Let's use the default Python format the browser immediately opens a new tab for me this is the interactive shell we mentioned earlier.

When you’re in the shell here – you will write code and see its output in the same interface. Here everything is integrated and there is no separate window for the results – you can store the results in the same notebook and even save the results for me to see if you sent me the notebook and what you were doing at the time. This makes for an exceptional learning experience and truly highlights one of the fundamental strengths of Python and that is its interactive-ness!

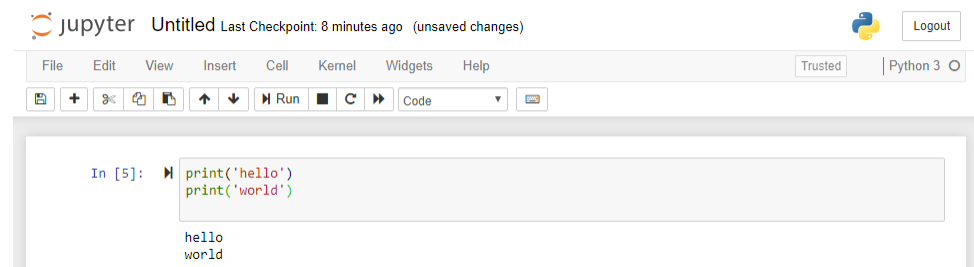
Let’s start with the programmers’ traditional input/output command to see “is this thing on?”. You may quickly notice that hitting enter just proceeds to insert a bunch of carriage returns into the cell within which you’re typing. This is normal behaviour in a Jupyter notebook because some of the cells will contain multiple lines of code and by default when you hit enter you most likely want to add a new statement rather than actually execute that particular statement. When we want to run our print line we simply hit

Shift+Enter. This will execute the current cell and everything that is in it. Notice that the output is basically in the same region as the code area where you were typing!



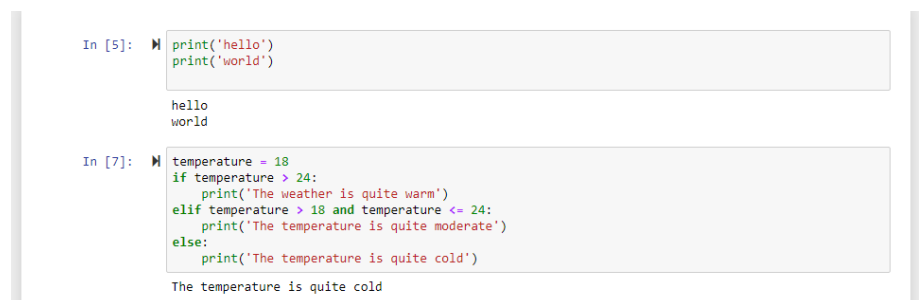
```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [2]: print('hello world')
hello world
```

Notice that you can stack multiple statements in one Jupyter “cell” and those multiple statements are executed by hitting Shift+Enter again.



```
jupyter Untitled Last Checkpoint: 8 minutes ago (unsaved changes) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [5]: print('hello')
        print('world')
hello
world
```

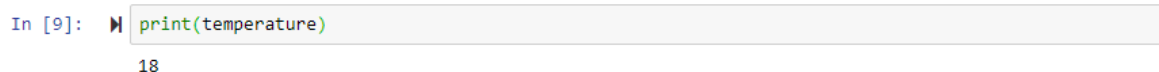
Let’s try some multiline statements where we can see that you can use decision control structures inside of Jupyter notebooks to test your logic and conditional statements and also couple that with changing the variable values themselves to evaluate the results in real-time. Let’s try the following in the new cell below our print statements.



```
In [5]: print('hello')
        print('world')
hello
world

In [7]: temperature = 18
        if temperature > 24:
            print('The weather is quite warm')
        elif temperature > 18 and temperature <= 24:
            print('The temperature is quite moderate')
        else:
            print('The temperature is quite cold')
The temperature is quite cold
```

Try altering the temperature and re-executing the code itself by again pressing Shift+Enter. You should see the new result! This is particularly useful for teaching you how to interact with a series of unfamiliar APIs – such as the ArcGIS for Python API because things like the map can be rendered right here in the notebook without leaving the environment or toggling between Python code windows and output windows. Moreover, I can easily see what you’ve done and help point you in the right direction if you’ve got incorrect syntax or issues with your structure or logic. You can see that variables persist outside of the current cell – in the example below I’ve printed the value of the temperature variable from a calculation done in the previous cell (or further above).



```
In [9]: print(temperature)
18
```

Rather than walk you through the subtleties of Jupyter notebooks themselves it is probably best to provide you with the best cheat sheet on the subject and have you go have a look at it for yourself to understand the environment. The cheat sheet below gives you the information you'll need to understand the GUI of the notebooks.

This is provided freely by a data science company called DataCamp – a higher resolution version of the PDF is located at :

[https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Jupyter Notebook Cheat Sheet.pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Jupyter_Notebook_Cheat_Sheet.pdf) -

have a look at the entire sheet in totality so that you have a good understanding of the interface and the possibilities with the environment.

There are also a series of essential keyboard shortcuts that will make interacting with Jupyter notebooks much easier! Sometimes you'll mistakenly hit a key while in command mode and wonder what has happened – the shortcuts below should answer some of those questions.

Command Mode (press <code>Esc</code> to enable)		Edit Mode (press <code>Enter</code> to enable)	
<code>F</code> : find and replace	<code>Shift-Down</code> : extend selected cells below	<code>Tab</code> : code completion or indent	<code>Ctrl-Right</code> : go one word right
<code>Ctrl-Shift-F</code> : open the command palette	<code>Shift-J</code> : extend selected cells below	<code>Shift-Tab</code> : tooltip	<code>Ctrl-Backspace</code> : delete word before
<code>Ctrl-Shift-P</code> : open the command palette	<code>A</code> : insert cell above	<code>Ctrl-I</code> : indent	<code>Ctrl-Delete</code> : delete word after
<code>Enter</code> : enter edit mode	<code>B</code> : insert cell below	<code>Ctrl-[</code> : dedent	<code>Ctrl-Y</code> : redo
<code>P</code> : open the command palette	<code>X</code> : cut selected cells	<code>Ctrl-A</code> : select all	<code>Alt-U</code> : redo selection
<code>Shift-Enter</code> : run cell, select below	<code>C</code> : copy selected cells	<code>Ctrl-Z</code> : undo	<code>Ctrl-M</code> : enter command mode
<code>Ctrl-Enter</code> : run selected cells	<code>Shift-V</code> : paste cells above	<code>Ctrl-/</code> : comment	<code>Ctrl-Shift-F</code> : open the command palette
<code>Alt-Enter</code> : run cell and insert below	<code>V</code> : paste cells below	<code>Ctrl-D</code> : delete whole line	<code>Ctrl-Shift-P</code> : open the command palette
<code>V</code> : change cell to code	<code>Z</code> : undo cell deletion	<code>Ctrl-U</code> : undo selection	<code>Esc</code> : enter command mode
<code>M</code> : change cell to markdown	<code>Q</code> , <code>Q</code> : delete selected cells	<code>Insert</code> : toggle overwrite flag	<code>Shift-Enter</code> : run cell, select below
<code>R</code> : change cell to raw	<code>Shift-M</code> : merge selected cells, or current cell with cell below if only one cell is selected	<code>Ctrl-Home</code> : go to cell start	<code>Ctrl-Enter</code> : run selected cells
<code>1</code> : change cell to heading 1	<code>Ctrl-S</code> : Save and Checkpoint	<code>Ctrl-Up</code> : go to cell start	<code>Alt-Enter</code> : run cell and insert below
<code>2</code> : change cell to heading 2	<code>S</code> : Save and Checkpoint	<code>Ctrl-End</code> : go to cell end	<code>Ctrl-Shift-Minus</code> : split cell at cursor
<code>3</code> : change cell to heading 3	<code>L</code> : toggle line numbers	<code>Ctrl-Down</code> : go to cell end	<code>Ctrl-S</code> : Save and Checkpoint
<code>4</code> : change cell to heading 4	<code>O</code> : toggle output of selected cells	<code>Ctrl-Left</code> : go one word left	<code>Down</code> : move cursor down
<code>5</code> : change cell to heading 5	<code>Shift-O</code> : toggle output scrolling of selected cells		<code>Up</code> : move cursor up
<code>6</code> : change cell to heading 6	<code>Q</code> : show keyboard shortcuts		
<code>7</code> : change cell to heading 7	<code>I</code> , <code>I</code> : interrupt the kernel		
<code>8</code> : change cell to heading 8	<code>R</code> , <code>R</code> : restart the kernel (with dialog)		
<code>K</code> : select cell above	<code>Esc</code> : close the pager		
<code>Up</code> : select cell above	<code>Q</code> : close the pager		
<code>Down</code> : select cell below	<code>Shift-L</code> : toggles line numbers in all cells, and persist the setting		
<code>Shift-K</code> : extend selected cells above	<code>Shift-Space</code> : scroll notebook up		
<code>Shift-Up</code> : extend selected cells above	<code>Space</code> : scroll notebook down		

ArcGIS for Python API (Preamble - for those unfamiliar)

Our world is very connected, one in which systems communicate with one another all the time and in real-time. Have you wondered how this communication takes place? How does data get from the device in your pocket to a system that may be halfway around the world? Inevitably, the answer comes down to APIs or Application Programming Interfaces. APIs broker and handle the connections between the distinct/different systems that typically run on divergent and differing technology stacks (languages, databases, and platforms). Similar to what you've learned in the introductory colloquia class, these APIs are objects, methods and properties used by developers to enable integration. In this particular instance, the API we're going to work with is the ArcGIS for Python API developed by ESRI. The ArcGIS for Python

API is a series of classes, modules and functions for interfacing with Geographic Information Systems (GIS). ESRI is the world-leading provider of GIS software and is a complete extendable platform for analyzing any data in geographic space. We will leverage the spatial analytical tools provided by this API to perform tasks and operations on our own data or on the data owned by others without having to know the nuts and bolts of how these operations take place. This is one of the more powerful elements of being a proficient developer – not only do you know how to write your own software, but by knowing how to code you also know how to leverage software written by others. This is very powerful, for example, we may not know the exact inner workings of how ESRI implements something advanced like a spatial join between two disparate data layers, but we know how to use the “black box” and give it the required parameters in order to undertake the analysis.

The realm of spatial/geographic data may be new to most of you as many have not touched on these types of data before. Here are some of the properties and characteristics of spatial data:

- Geographic features
 - natural environment – rivers, lakes
 - built environment – roads, cities, parks
- Attributes
 - size of towns and cities
 - types and lengths of roads
- Location
 - absolute – precise location with grid references
 - relative – places identified in relation to the location of other places
- Spatial relationships
 - direction, connectivity, containment, adjacency
 - proximity – nearness or distance between places
- Time
 - geographic features are time sensitive, road networks change, communities grow, new features are added

The ArcGIS API for Python communicates directly with a GIS server (either your own via ArcGIS Enterprise or those supplied by ESRI - ArcGIS Online) using [REST](#). For example, when creating a map, the API sends a request to a service to create a map based on the objects and properties set in your programming code. The API was simplified so that any novice developer can implement GIS functions in to their Python scripts by writing just a few lines of code. I’ve implemented a GIS server for you so we can do all of the analysis using this machine without it costing us credits. The first thing we need to do to begin, is to ensure that your environment is setup correctly to start working with the API. For this we will use the Anaconda distribution.

Exploration of the ArcGIS for Python API

Now that we’re familiar with the concept and practical application of Jupyter notebooks let’s look at them from the perspective of ArcGIS’s Python API. As with anything in Python, to include capabilities outside of Python’s standard libraries we have to reference them using an import statement. This is no different when we’re using our Jupyter notebook.

Rather than reiterate publicly available information, have a read through the following article that explains how to harness the power of the ArcGIS for Python API https://1drv.ms/b/s!AhqafjB_FqAekPBQfYKT8pPg5ZmBjw. This explains the structure of the API, the purpose of it and describes some of the capabilities of the API. The latter statement is constantly being added to as new versions of the API get released so the list of capabilities will surely change through time as more enhancements get made.

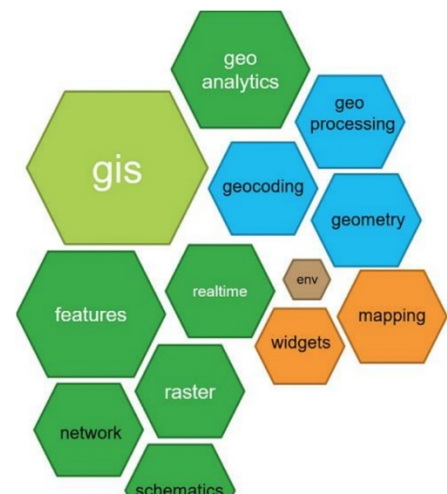
Here is a list of the modules that exist in the ArcGIS for Python API:

```
arcgis.gis
arcgis.env
arcgis.features
arcgis.raster
arcgis.network
arcgis.schematics
arcgis.geoanalytics
arcgis.geocoding
arcgis.geometry
arcgis.geoprocessing
arcgis.realtime
arcgis.mapping
```

The current version is version 1.9 (at the time of writing this) was released in late July, 2021 (there may be updates since then as the API is always evolving!).

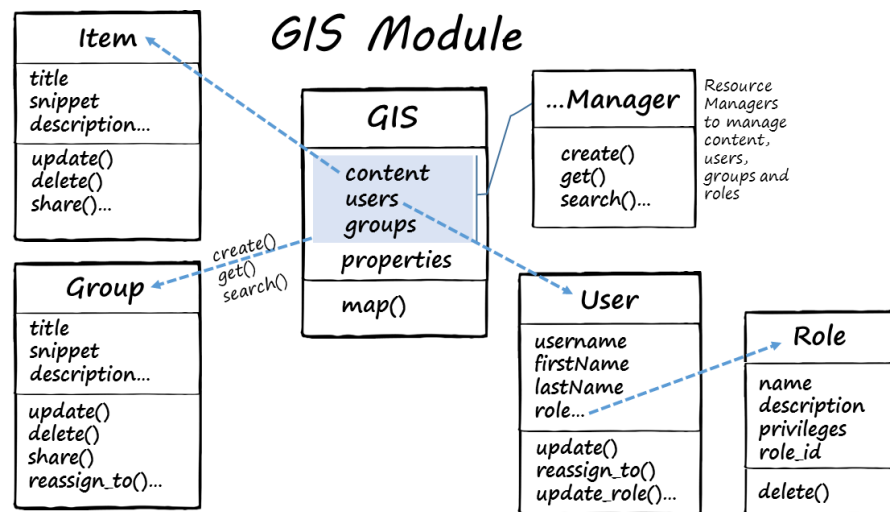
The API consists of several modules for:

- managing local ArcGIS enterprise / ArcGIS Online (module `arcgis.gis`)
- vector data analysis, features and feature layers (module `arcgis.features`)
- raster data analysis (module `arcgis.raster`)
- undertaking network analyses (module `arcgis.network`)
- distributed analysis of large datasets (module `arcgis.geoanalytics`)
- geocoding and georeferencing tasks (module `arcgis.geocoding`)
- answering questions about locations (module `arcgis.geoenrichment`)
- manipulating spatial geometries (module `arcgis.geometry`)
- geoprocessing tools (creating / sharing)
- visualizing data and creating maps (module `arcgis.mapping`)
- real-time data feeds (module `arcgis.realtime`)
- including maps and visualizations as widgets (module `arcgis.widgets`)




The `gis` module allows us to connect to a Geographic Information System either on-prem or in the cloud. It features the following structure in general as seen in the image below. You can find out more about

how to use this specific module by visiting this url: <https://developers.arcgis.com/python/guide/using-the-gis/>



Source: ESRI, 2021

To use the GIS module, we type the following (see below), notice the lack of autocomplete after referencing and running the first cell. To enable autocomplete, you type in the identifier/variable name (in this case **gis** and append a period after the name and hit the TAB key and it will show you the options available). If you don't know what something is or how to use it you can use the `dir` command or even better check the help menu on the developers.arcgis.com site as they will have a detailed explanation of the methods and properties associated with that particular object. Whenever you see the arrow  in my instructions that means that you're typing the statement into a jupyter notebook cell.

We're going to use an ArcGIS Online organizational account through the University of Toronto using your UTORID Click the following link and then use your Enterprise logon <http://bit.ly/2O1jKWh>

Everyone starts with some credits; however, there are many things you can do with ArcGIS Online that does not require any credits at all. If a student wants to do something that could potentially be a heavy usage of credits, such as geocoding a large number of addresses (i.e., putting pins on a map that correspond to postal addresses) then they should contact MDL for assistance. For our example, we're going to try and minimize the use of credits as we're just in the exploratory stages!

Traditionally you could authenticate directly through the API if you were a developer with an ArcGIS Online account, however, since we're using the University's ArcGIS Online organization and their single-sign-on, we'll need to use OAuth to authenticate (rather than providing ESRI with our UTORID credentials – which would be insecure!). Instead we'll use OAuth 2 as per below.

ArcGIS for Python API - User authentication with OAuth 2.0

The ArcGIS Python API supports [OAuth 2.0](#) as an authentication method, and acts as a [serverless native application](#) when using OAuth 2.0 authorization with ArcGIS.

To use this mode of authorization, you need a `client id` . If you already have a client id, you can skip the following section.

Obtaining a client id

The steps below show how a client id can be obtained by registering a new application with your GIS. Only one client id is required, so if your GIS has one already, you may use it instead of creating a new application.

- Log into your web GIS
- If you do not have an account please create one by using OPTION #1 here: <https://mdl.library.utoronto.ca/technology/tutorials/logging-arcgis-online>
- the UToronto ArcGIS Organization: <http://bit.ly/2O1jKWh>
- Go to **Content** tab
- Click '+ New Item > Application' menu
- Add an application – feel free to give it a title & tags:
 - Type: **Application**
- On the Item details page of this newly created application, navigate to **Settings** tab and scroll down to the **Application** section.
- Click the **Registered Info** button. It's towards the bottom of the page.
- This will display an App ID. This is your client id and needs to be passed in as the `client_id` parameter when constructing a GIS object. You need this in your Python code to log in.

You can then log on to your org using the Python API using the code shown below:

```
from arcgis.gis import GIS
gis = GIS("https://utoronto.maps.arcgis.com",client_id="<your-client-id aka APP ID string>")
print("Successfully logged in as: " + gis.properties.user.username)
```

This uses interactive sign-in experience: you would be redirected to our organization's sign in page using the configured identity provider. Upon signing in, you would get a code, that you can paste back to complete the sign-in process (either in VSCode or in your Jupyter notebook). You have to paste it back quite quickly or it becomes invalidated! **You may receive a warning about an unverified SSL connection to utoronto.maps.arcgis.com**, you can safely ignore this or dismiss it completely by adding the following 2 lines to the top of your code

```
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
```

```
C:\Users\username\anaconda3\lib\site-packages\urllib3\connectionpool.py:10
13: InsecureRequestWarning: Unverified HTTPS request is being made to host
'utoronto.maps.arcgis.com'
```

Once you have a valid “gis” object, you can find out more of the details of the objects by typing their name and putting a question mark ?

```
In [5]: gis?
```

```
Type: GIS
String form: GIS @ https://portal.arts.ryerson.ca/arcgis version:7.1
File: c:\programdata\anaconda3\lib\site-packages\arcgis\gis\__init__.py
Docstring:
.. _gis:

A GIS is representative of a single ArcGIS Online organization or an ArcGIS Enterprise dep
provides helper objects to manage (search, create, retrieve) GIS resources such as content

Additionally, the GIS object has properties to query its state, which is accessible using
```

```
In [ ]: gis.
```

```
content
groups
map
properties
update_properties
url
users
version
```

You can see the five helper objects that help manage GIS resources. Read about these five helper objects here: <https://developers.arcgis.com/python/guide/using-the-gis/#Helper-objects>

```
In [ ]: gis.
```

```
gis.content
gis.groups
gis.map
gis.properties
gis.users
```

The gis object has a very basic method called map that takes in the argument of the location you want to map. This creates a map widget and allows you to quickly map any location identifiable by name. Try a few locations out! Points of interest are also possible – for example, if you type in University of Toronto it will bring you to the spatial extent of the campus.

We can use our helper functions to enumerate the data sources in our enterprise GIS system. I’ve taken the liberty of adding some data to our organization for you to work with – just a few layers, tagged with the tag **python2019** or owner **macdo925_utoronto** . To find out what sources are available, use the gis.content object’s search method and pass in an empty string – the second parameter denotes what type of information we want to return – in this case we want to show vector layers (aka feature layers).

You can search for any content within your organization or filter by a bunch of different properties: <https://developers.arcgis.com/python/guide/accessing-and-creating-content/#searching-for-content>

Creating a Map widget:

Let’s try doing a basic map of the Toronto area – to do this just issue the following command:

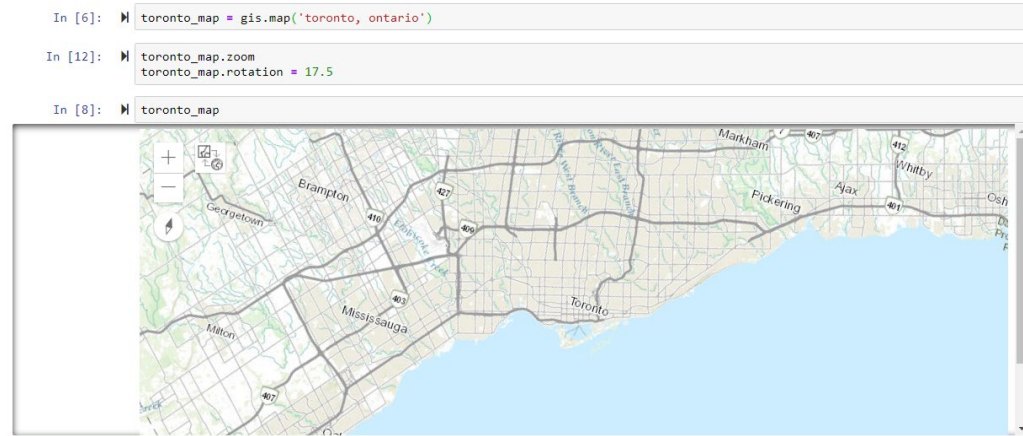
You should notice that the map immediately appears in the same cell



If we're looking to modify the properties of the map we can actually assign a variable that will be an instance of the map class and then we can manipulate its properties. To do so issue a command like the following:

```
In [6]: toronto_map = gis.map('toronto, ontario')
```

You should notice that nothing appears – however you can type `toronto_map` and append a period and you can see that it is, in fact, an object/variable that we can then manipulate. Let's alter some of the properties of the object and then we'll display it



Notice that the map now appears to be a slightly different rotation and zoom level. Play with some of the methods and properties of the object itself – you can find documentation on them here <https://esri.github.io/arcgis-python-api/apidoc/html/arcgis.widgets.html#mapview>. The `Toronto_map` variable is essentially a map widget

Changing the properties of the map

Here are some examples to try in addition to your own:

```

In [16]: ► toronto_map.center

Out[16]: {'spatialReference': {'latestWkid': 3857, 'wkid': 102100},
          'x': -8837146.757199818,
          'y': 5411286.851528861}

In [17]: ► toronto_map.basemap

Out[17]: 'default'

In [18]: ► toronto_map.extent

Out[18]: {'spatialReference': {'latestWkid': 3857, 'wkid': 102100},
          'xmin': -8908921.126759626,
          'ymin': 5380712.040214779,
          'xmax': -8765372.38764001,
          'ymax': 5441861.662842942}

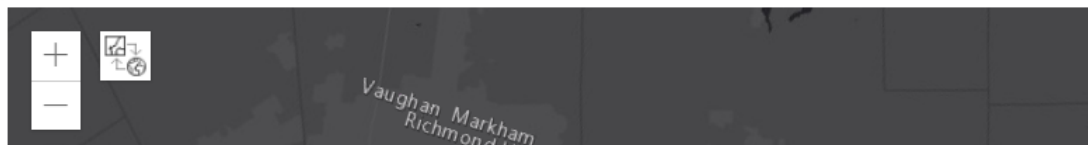
```

Try playing with the basemap property and applying a different basemap to your `toronto_map`. You can do this by issuing the following command – explore some of the other options on your own:

```

In [22]: ► toronto_map.basemap = ('dark-gray')
toronto_map

```



Valid basemap types are: *['dark-gray', 'dark-gray-vector', 'gray', 'gray-vector', 'hybrid', 'national-geographic', 'oceans', 'osm', 'satellite', 'streets', 'streets-navigation-vector', 'streets-night-vector', 'streets-relief-vector', 'streets-vector', 'terrain', 'topo', 'topo-vector']*

The mapping widget is even capable (in the most recent releases of the ArcGIS for Python API of having a 3d view!). Try that using the following commands:

```

In [28]: ► toronto_map.basemap = ('dark-gray')
toronto_map.mode="3D"
toronto_map.zoom=4
toronto_map

```

Play with the **tilt** and the **heading** properties (submit numeric values for both)

By using the statement **toronto_map** at the end it causes the object to render itself in the Jupyter notebook.

Analysis

I've published a subset of the Major Crime Indicators on a GIS server – you can retrieve it by using the **gis.content.get()** method as follows – the long string passed is the GUID of the layer that we want to render. Other layers are available by searching ArcGIS Online's free data sources at <https://www.arcgis.com/home/index.html> :

```

In [1]: items = gis.content.search('', 'feature layer')
for item in items:
    print(item)

<Item title:"Toronto Neighbourhoods" type:Feature Layer Collection owner:pythondev>
<Item title:"University of Toronto" type:Feature Layer Collection owner:pythondev>
<Item title:"Crime2018" type:Feature Layer Collection owner:pythondev>

```

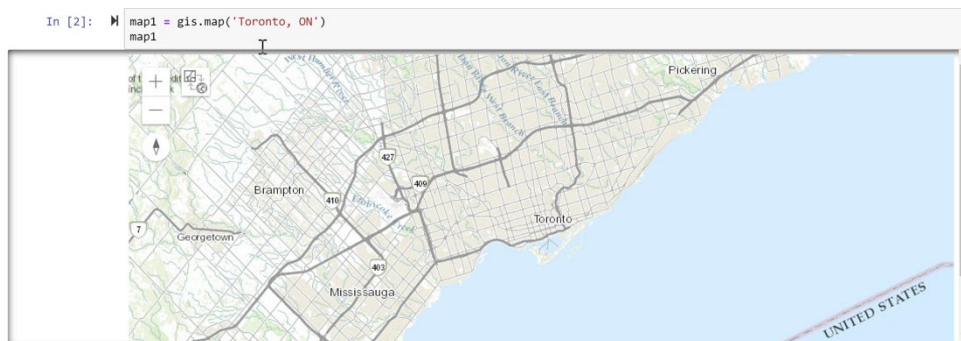
Each “item” here can actually consist of one or more layers of information. You can see this by iterating over the layers property of each respective item. To see this, we can iterate over each object and it’s respective layers by doing the following – you can also try searching by **tags:python2019** instead by replacing the owner parameter below – it should return roughly the same result unless others have added data with that particular tag:

```

In [2]: items = gis.content.search('owner:macdo925_utoronto', 'feature layer')
for item in items:
    print(item)
    for lyr in item.layers:
        print(lyr.properties.name)

<Item title:"Crime2018" type:Feature Layer Collection owner:macdo925_utoronto>
Crime2018
<Item title:"RSM8431 Toronto Neighbourhoods" type:Feature Layer Collection owner:macdo925_utoronto>
toronto_neighbourhoods
<Item title:"utoronto" type:Feature Layer Collection owner:macdo925_utoronto>
utoronto

```

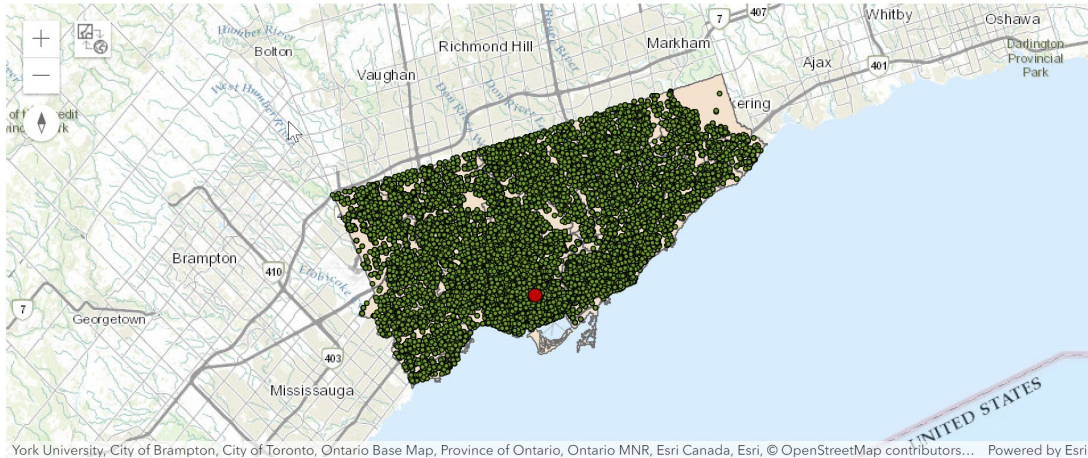


Now let’s add some data to the map

```

In [3]: map1 = gis.map('') # Reset the map widget
crime_locations = items[0] # get the first item - the crimes in Toronto in 2018
toronto_neighbourhoods = items[1] # get the second item - the toronto neighbourhoods
map1.add_layer(toronto_neighbourhoods) # add the neighbourhoods to the map
university_location = items[2] # get the third item - the university location
map1.add_layer(crime_locations) # add the crimes Layer to the map
map1.extent = toronto_neighbourhoods.extent # set the extent to be the spatial envelop of the neighbourhoods
map1.add_layer(university_location) # add the university to the map
map1

```

We can explore the data that we added onto the map widget by iterating through the first layer in the hosted feature layer. This can be done by creating the following cell in the jupyter notebook – this will list out all of the fields that exist:

```
# Let's evaluate what fields exist within our data Layers

for field in crime_locations.layers[0].properties.fields:
    print(field['name'])

<class 'arcgis.gis.Item'>
['advancedQueryCapabilities', 'allowGeometryUpdates', 'allowTrueCurvesUpdates', 'allowUpdateWithoutMValues', 'capabilities', 'cimVersion', 'copyrightText', 'currentVersion', 'dateFieldsTimeReference', 'defaultVisibility', 'description', 'displayField', 'drawingInfo', 'editFieldsInfo', 'enableZDefaults', 'extent', 'fields', 'geometryField', 'geometryType', 'globalIdField', 'hasAttachments', 'hasM', 'hasMetadata', 'hasZ', 'htmlPopupType', 'id', 'indexes', 'isCoGoEnabled', 'isDataArchived', 'isDataBranchVersioned', 'isDataVersioned', 'maxRecordCount', 'maxRecordCountFactor', 'maxScale', 'minScale', 'name', 'objectIdField', 'onlyAllowTrueCurveUpdatesByTrueCurveClients', 'parentLayer', 'relationships', 'serviceItemId', 'sourceSpatialReference', 'sqlParserVersion', 'standardMaxRecordCount', 'supportedQueryFormats', 'supportsAdvancedQueries', 'supportsApplyEditsWithGlobalIds', 'supportsAsyncDelete', 'supportsAttachmentsByUploadId', 'supportsCalculate', 'supportsCoordinatesQuantization', 'supportsRollbackOnFailureParameter', 'supportsStatistics', 'supportsValidatesSQL', 'syncCanReturnChanges', 'templates', 'tileMaxRecordCount', 'type', 'typeIdField', 'types', 'useStandardizedQueries', 'zDefault']
fid
index_
event_uniq
occurrence
reportedda
premisetyp
ucr_code
ucr_ext
```

Items that you can work with:

Feature – one particular object, one dot on the map aka one crime.

has attributes like: geometry, attributes, fields

FeatureSet – set of features, can be put on map widget or input into a tool for geoprocessing

has attributes like: df, features, fields and spatial_reference

FeatureCollection - items in GIS, input/output of geoprocessing

has attributes like: query and others

FeatureLayer - through individual layers of a feature services.

has attributes like: query, edit_features, calculate, query_related_records()

Spatial Enabled DataFrame – simple object to manipulate geometric and attribute data

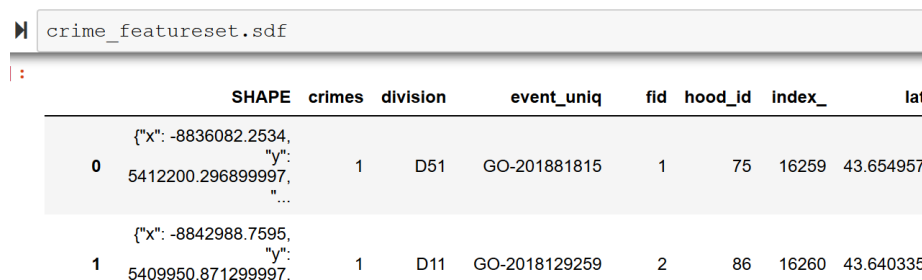
```
crime_featurelayer = crime_locations.layers[0]
crime_featureset = crime_featurelayer.query()
print(crime_featureset.geometry_type)

esriGeometryPoint
```


The command below can take a while – the notebook is ready once the [*] is finished and changes back to a numeric value in between the square brackets.

Once we have the featureset we can find out the geometry of the specific layer, its spatial reference or projection ([what is a spatial reference?](#)) by using the *geometry_type* and *.spatial_reference* properties respectively.

Let's display the data now using a new object introduced recently into the API called a spatial data frame (or sdf). To do this, we use our existing object *crime_locations* which is just an item data type. We're going to use this to access the layers property and return the first layer which we know to be the crimes themselves. At this point, other than the attributes we don't have information about what these data fields look like – a spatial data frame (basically a glorified table) will display this information in the notebook – this will inform our queries going forward.



	SHAPE	crimes	division	event_uniq	fid	hood_id	index_	lat
0	{ "x": -8836082.2534, "y": 5412200.296899997, "..."	1	D51	GO-201881815	1	75	16259	43.654957
1	{ "x": -8842988.7595, "y": 5409950.871299997, "..."	1	D11	GO-2018129259	2	86	16260	43.640335

Take a detailed dive into the data just by scrolling through as this is going to inform what we're able to do with the data. For example, what properties we're able to query, map, chart, etc. Have a look at this so you can determine what you want to be able to query for your own custom queries you'll develop.

Explore the readings, particularly the ones in the appendices here and feel free to get creative with how you build your functions! There is no "right answer" and as a developer you may choose to do things differently.

ASSIGNMENT: Build a COVID-19 Jupyter Spatial Notebook:

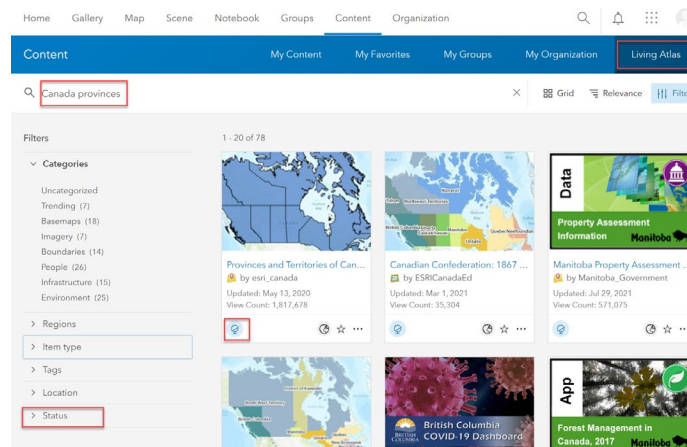
Often, in your careers as a data scientist or analyst, you will be tasked with using diverse sources of data and APIs created by others. For this assignment, we're going to use the ArcGIS API for Python and some data that is maintained by ESRI and/or other sources that you curate.

To begin, make sure your environment is setup to run the ArcGIS for Python API – you can do that by referring to the optional exercise above [Exercise: Introduction to the ArcGIS API for Python](#) – specifically [installing the API](#).

Review the following sample notebook <https://developers.arcgis.com/python/sample-notebooks/covid19-part1-mapping-the-pandemic/> to have a look at what is possible both with the API and the data source available. Feel free to use your own data sources for this exercise – the possibilities are only limited by your imagination and the available data! Based upon the evaluation of the API and available data sources – feel free to use anything here or other sites you find on your own: <https://mdl.library.utoronto.ca/covid-19/data> but I'm fine with you using the ESRI examples as long as you've chosen a different geography than in the examples.

Your submission should include / have the following elements:

- (1) A function/method that connects to a GIS server and returns an active connection – return the connection object. Since the API is essentially stateless, it may be a good idea to have your functions return objects that you wish to persist outside of the current scope.
- (2) Choose a geography that you're interested and have that sufficient cases/geographic area to make an interesting visualization.
 - a. Note: you'll have to likely get the geometry of the country/region you're choosing to visualize. You can generally do that by searching for content in the **Living Atlas**. You'll generally want something that has is curated by ESRI. In this example, the first one would have the provincial boundaries that we are looking for.



- (3) Create two maps of your chosen geography – perhaps a choropleth map using a map classification technique (classificationMethod). Consider making functions where it makes

sense. The example provided above shows how this can be done for China and the US (as well as the World) – feel free to use maps similar to these examples but for your unique area/region.

<https://developers.arcgis.com/python/sample-notebooks/covid19-part1-mapping-the-pandemic/>

- (4) Create some charts / visualizations using pandas of the data for your selected region and a temporal timeframe if your data permits. Consider making functions where it makes sense.
- (5) Make sure all functions handle errors and fail cleanly.
- (6) Make sure your code is well-commented and documented (with docstrings, etc.)
- (7) Include a README file explaining your submission and listing any required dependencies, issues encountered, etc.

Email deliverables to michaelw.macdonald@utoronto.ca in .ipynb, .py files, .zip if needed, etc. Include a README detailing anything I'll need to execute/test your module(s). **DUE: Sept 5th, 2021**

Appendix – Readings / Tutorials

ArcGIS API for Python: Getting to Know Pandas and the Spatial Enabled DataFrame

<https://www.youtube.com/watch?v=h1Lnz-rfWQo>

Spatially-Enabled DataFrames Advanced Topics <https://developers.arcgis.com/python/guide/spatially-enabled-dataframe-advanced-topics/>

The GIS module <https://developers.arcgis.com/python/api-reference/arcgis.gis.toc.html>

Finding places with geocoding <https://developers.arcgis.com/python/guide/using-the-geocode-function/>

Mapping and visualization <https://developers.arcgis.com/python/guide/using-the-map-widget/>

Smart mapping <https://developers.arcgis.com/python/guide/smart-mapping/>

Working with web maps and web scenes <https://developers.arcgis.com/python/guide/working-with-web-maps-and-web-scenes/>

Feature data and analysis <https://developers.arcgis.com/python/guide/working-with-feature-layers-and-features>

Using geoprocessing tools <https://developers.arcgis.com/python/guide/an-introduction-to-geoprocessing/>

Working with big data <https://developers.arcgis.com/python/guide/working-with-big-data/>

Mapping COVID-19 <https://developers.arcgis.com/python/sample-notebooks/covid19-part1-mapping-the-pandemic/>