# MIT - CLIx Session on Research Analytics – Environment Setup

## What is the MIT-CLIx-Session folder on the USB drive

|  |  |  |
| --- | --- | --- |
| **Software** | **Scripts** | **Datasets** |
| Python – version 2.7  MongoDB – version 3.4  Wget – version 1.11 | Student activity / timestamp CSV with interactives  Assessment / click-log CSV  OST / RKR click-log CSV | rj16  mz10  mz22  mz25 |

* Python version 2.7.15

Filename: python-2.7.15.amd64.msi

Downloaded from: <https://www.python.org/downloads/>

* MongoDB version 3.4

Filename: mongodb-win32-x86\_64-2008plus-ssl-3.4.16-signed.msi

Downloaded from: <https://www.mongodb.com/download-center#atlas>

Select: Community Server and Version

* Wget for Windows

Filename: wget-1.11.4-1-setup

Downloaded from: <http://gnuwin32.sourceforge.net/packages/wget.htm>

* MIT CLIx Data Scripts – The data scripts that Cole and Glenda created are made up of multiple python scripts that work together.
* School Data example – The data sets were downloaded from the TISS server.

# Understanding the script environment

The scripts that Cole and Glenda developed are written in Python and are run from the command line. This means that the scripts do not have a graphical interface and you will need to type the commands into a Windows Terminal.

In order to analyze the data being gathered in the field, you will need to install and use several pieces of software.

* Wget – this is used to copy the data from the server to your local hard drive.
* MongoDB – this is the database where the CLIx data is stored.
* Python – this is the programming language used to develop the scripts.
* Scripts – these are Python scripts that go through the data in the MongoDB and collect selected pieces of information.
* Windows Terminal – sometime called the “command prompt”, the Windows Terminal is where you will type in the script commands.

## Setting up and testing your environment

Following these steps will install the necessary software and create a python sandbox with the required software versions needed for running the data scripts. You will only need to follow these steps once. Once you have created and setup your python sandbox, you will be able to use it whenever you need to run these scripts.

\*\* Please be aware: These scripts will only work on 64 bit windows operating systems because the data is stored in MongoDB using wired-tiger which is in a 64 bit format.

1. Copy the software, scripts and dataset to your local hard drive from the USB drive

On the USB drive is a folder called “MIT-CLIx-Session”, copy this folder to your local computer. This folder contains the software, scripts, and dataset that you will need for this session.

1. Install Python

In the MIT-CLIx-Session folder, double click on the python installer called python-2.7.15.amd64.msi

1. Install MongoDB

In the MIT-CLIx-Session folder, double click on the MongoDB installer called mongodb-win32-x86\_64-2008plus-ssl-3.4.16-signed.msi

1. Install Wget

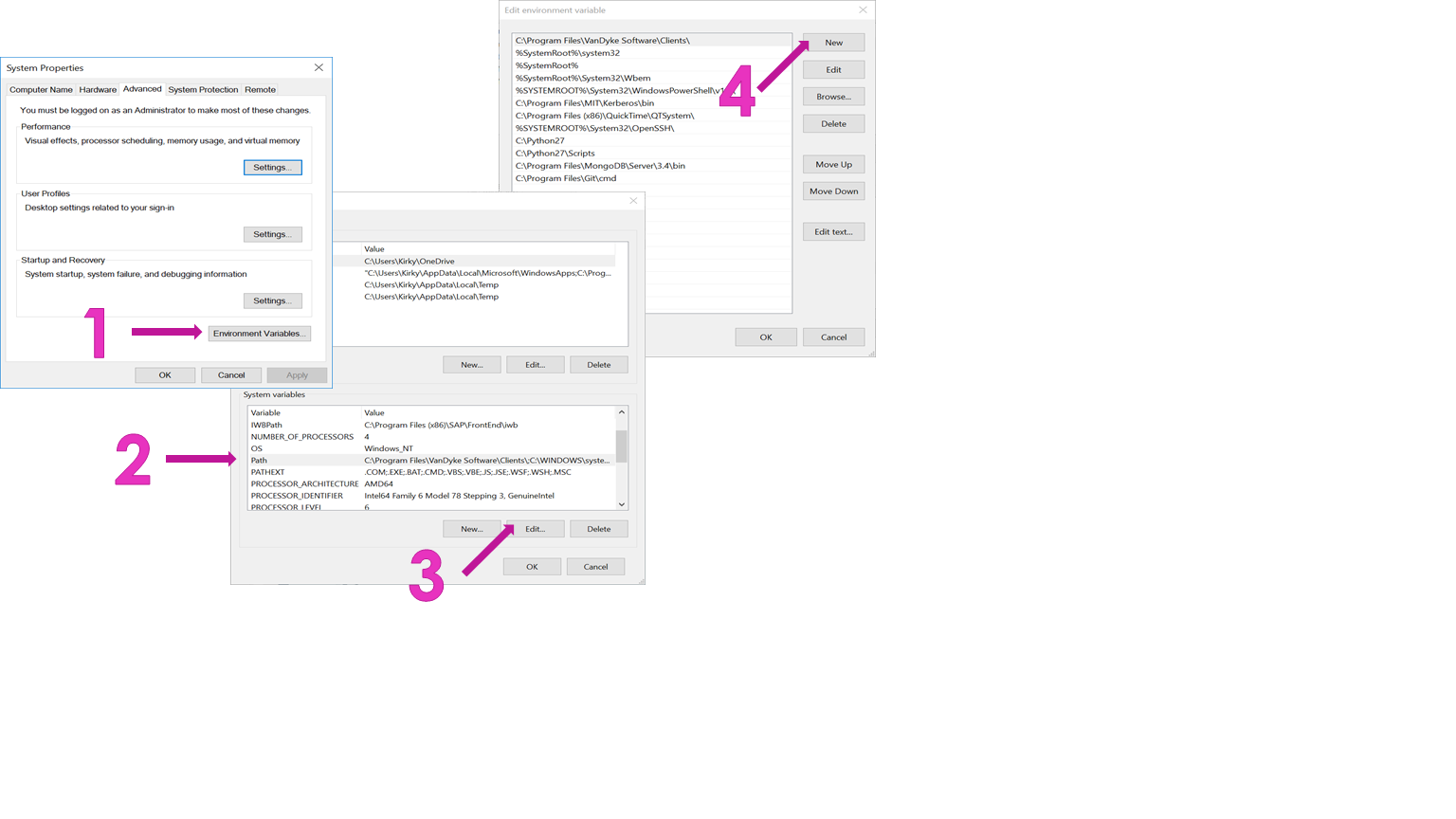
In the MIT-CLIx-Session folder, double click on the wget-1.11.4-1-setup.exe installer. For today’s session, the data we will be using is already in the USB drive folder. For future data analysis, you will want to download the raw data from CLIx server using a tool like wget.

1. For Windows machines, you will have to add Python and MongoDB software to your "Path" variable. The following instructions are for **Windows 10** operating systems.

* Search for "advanced settings" in the search bar
* Click "Environment variables"
* Under "System variable" look for "Path"
* Highlight the "Path" variable and select Edit

Add "New" entries that point to where you installed the above software packages. The default program locations would be:

* + C:\Python27
  + C:\Python27\Scripts
  + C:\Program Files\MongoDB\Server\3.4\bin



1. Create a Mongo data directory

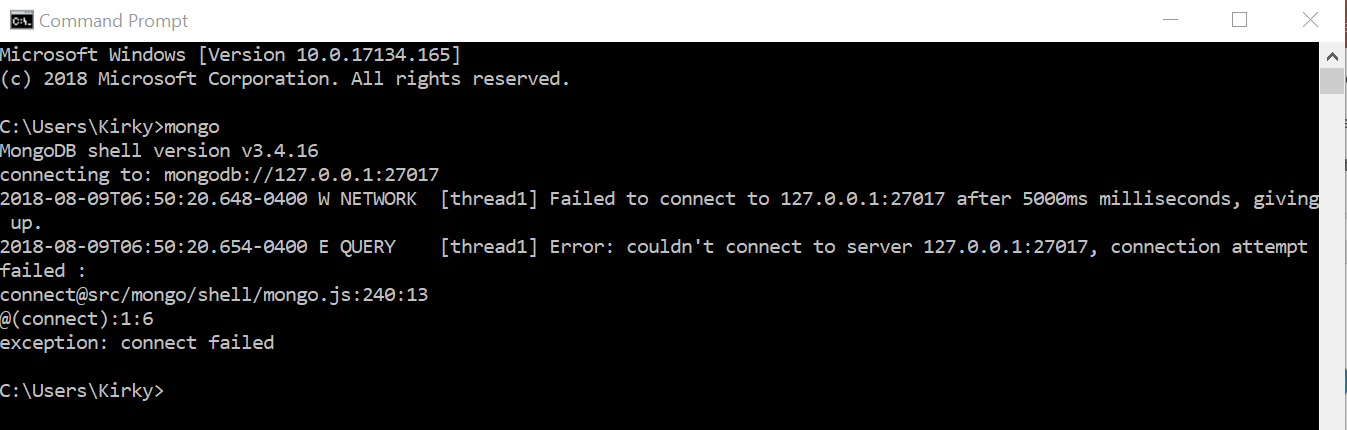
MongoDB expects the default data directory to be at C:\data\db\

* Using the Windows Explorer, create a folder called “data” with a subfolder called “db”

1. Test your MongoDB installation

* Open a terminal (also known as the Command Prompt) by typing CMD into the search bar
* Type: mongo

MongoDB will not start because you have not specified a database but you should see it try to start and return a “fail to connect” message.



1. Test your Python installation

* In the terminal
* Type: python

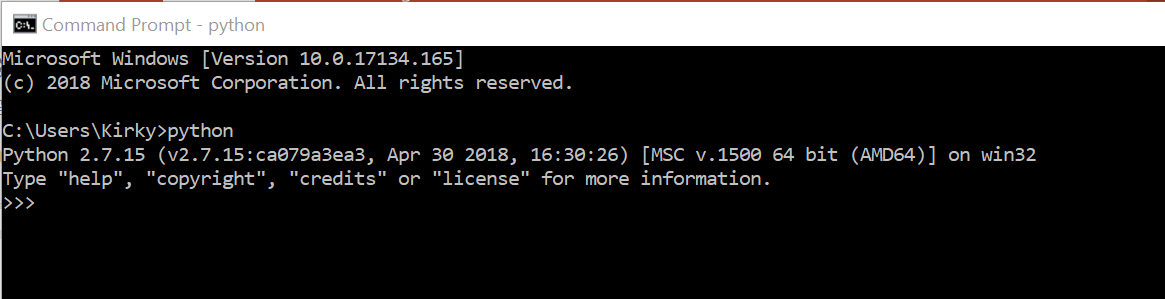
You should see something similar to:

Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>

* Type: exit()



1. Install sandbox environment software (this creates a safe and reusable environment for running scripts)

* In the terminal
* In the CMD window, type: pip install virtualenv virtualenvwrapper-win

1. Create your sandbox and name it.

Create a sandbox called clix-data (you can call your sandbox anything you want – this one is called clix-data). Please be sure to remember or write down the name of your sandbox environment.

* In the terminal
* Type: mkvirtualenv clix-data

Once you have created and setup your python sandbox (in this case called clix-data), you will be able to use it whenever you need to run these scripts without having to reinstall or setup software. If you already have a sandbox created and want to get back into it, type workon clix-data (or your sandbox name) in the terminal window.

1. Navigate to scripts directory

* In the CMD window, use the “cd <directory path>“ command to navigate to the directory where your copy of the scripts is located.

cd = change directory

dir = directory listing

* The scripts are located in the MIT-Data-Scripts sub-folder of the MIT-CLIx-Session folder you copied to your hard drive.
* If you copied the folder to your desktop, you would type: cd Desktop/ MIT-CLIx-Session/ MIT-Data-Scripts

1. Install libraries needed for scripts (libraries are a collection of software tools used to run the scripts)

* In the terminal
* Type: pip install -r requirements.txt

1. Close the terminal
2. Setup scripts environment (this is not done in the terminal)

* Open the MIT-Data-Scripts folder
* Create a copy of the settings.py.example file. Name this new file settings.py
* Open the IDLE (Python GUI) editor using the search bar
* Select File -> Open
* Navigate to the directory with the scripts
* Select the settings.py file and open
* Edit the settings.py file to look like the text below

# Use a non-standard MongoDB port

MONGO\_DB\_PORT = 27027

MONGO\_DB\_HOST = '127.0.0.1'

SYNC\_DATA\_PATH = 'data'

JENKINS\_JOB\_BUILD\_PATH = ('/home/.jenkins/jobs/your-sync-job/'

'builds/lastSuccessfulBuild/build.xml')

* Save the settings.py file with the changes you made

1. You are now ready to run the scripts.

You have now completed the software installation and created a python sandbox for running the data scripts. You will not need to follow the above steps again **as long as you put any new data sets into the “data” folder with the scripts** (specified in the settings.py).

## Getting the data

For today’s session, the data we will be using is already in the USB drive folder.

For future data analysis, you will want to download the raw data from CLIx server using a tool like wget. GNU Wget ("web get") is a computer program that retrieves content from web servers and copies it to your local computer. When run from the command-line, wget is non-interactive, meaning that it can work in the background, and has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved.

Please be aware that the data sets on the server are large and can take several hours to download.

Wget is run from the terminal using the format:

wget <<options>> <<data directory you want to copy>>

There are several options that you will want to include. These options will limit what is downloaded.

wget --recursive -nH --no-parent --cut-dirs=3 –no-check-certificate <<data directory>>

I used the following command to download the data sets we will be using today. The download of all the 2018 Mizoram data took about 4 hours on MIT’s high speed network. Because the wget utility is not in the windows path, the directory where the program is stored needs to be specified when it is called. In this case, wget is in the c:\Program Files (x86)\GnuWin32\bin directory.

"c:\Program Files (x86)\GnuWin32\bin\wget" --continue --recursive -nH --no-parent --cut-dirs=3 --no-check-certificate https://clixplatform.tiss.edu/softwares/Reseach\_data/2018/mz/

## Running the Scripts

The following scripts developed by Cole and Glenda, analyze the raw MongoDB database files to create CSV files that can be used for CLIx research analytics. There are three scripts.

1. Student activity / timestamp CSV with interactives
2. Assessment / click-log CSV
3. OST / RKR click-log CSV

Results for each of these files is e-mailed to the provided email address, if the script is run within the MIT network (on the internet within MIT - the email address can be a non-MIT address and still run within the MIT network). The results are then deleted afterwards. If the script is not run on the MIT network, an error message will appear in the terminal and the output files will be available in the folder where the scripts are located. Look for a file with the extension ‘.csv’

All three of these scripts are run from the command-line inside of the Sandbox environment. This is very important. These scripts will only run from Sandbox environment you created (clix-data or whatever you named your sandbox) because the Sandbox environment you created has the libraries installed that the scripts will need.

To run any of the scripts, open the Sandbox environment you created

* Open a terminal using the CMD command
* Navigate to the directory with the scripts
* Open the Sandbox environment

Type: workon clix-data (or whatever you named your sandbox)

## Student activity / timestamp CSV Script with interactives

Use this script to get student activity or timestamp data that includes interactive tools. This script returns a CSV file with all students listed in one class. Each student’s records are on one line.

This script is named track\_student\_activity.py and takes 3 arguments:

1. school: the school code as a string, like mz10
2. date: a starting date to get data from, like 2018-01-01
3. email: where to send the results (only works if script run from MIT network, but you must include this argument, even if not on the MIT network), like kirky@mit.edu

Examples:

(clix-data) $ python track\_student\_activity.py --school=<<school code>> --date=<<date>> [--email=<<email](mailto:--email=example@mit.edu) address>>

(clix-data) $ python track\_student\_activity.py --school=rj16 --date=2017-08-01 [--email=kirky@mit.edu](mailto:--email=kirky@mit.edu)

(clix-data) $ python track\_student\_activity.py --school=mz10 --date=2018-05-01 --email=kirky@mit.edu

## Assessment / click-log CSV Script

This script will get the open assessment data for one particular assessment. This script returns two CSV files: a CSV log file (log.csv) and a CSV file that provides assessment results (results.csv).

This script is named aggregate\_assessment\_data.py and takes 3 arguments:

1. bank: the bankId, from the iframe code, like assessment.Bank%3A58c991b791d0d90467f4679d%40ODL.MIT.EDU
2. offered: the offeredId, from the iframe code, like assessment.AssessmentOffered%3A59101a5991d0d93c65c55d41%40ODL.MIT.EDU
3. email: where to send the results (only works if script run from MIT network, but you must include this argument, even if not on the MIT network), like kirky@mit.edu

Example:

(clix-data) $ python aggregate\_assessment\_data.py --bank=<<bankId>> --offered=<<offeredId>> --email=<<email address>>

(clix-data) $ python aggregate\_assessment\_data.py --bank=assessment.Bank%3A58c991b791d0d90467f4679d%40ODL.MIT.EDU --offered=assessment.AssessmentOffered%3A59101a5991d0d93c65c55d41%40ODL.MIT.EDU --email=kirky@mit.edu

## bankId and offeredId

What are the bankId and the offeredId and why are they important?

The Open Educational Assessments (OEA) tool is embedded into a webpage using an iframe. An iframe, or "inline frame", is used to load content from someplace else into the current webpage. In this case, the iframe is loading the OEA tool into the current webpage and displaying an assessment for the student to take.

The OEA tool, just like a script, needs to know certain information before it can work. In this case, the tool needs to know where the assessment collection is located (bankId) and which particular assessment it should display (offeredId). This information is located in the src variable in the iframe. The iframe code can be found in two places: the authoring server <embed code> and the GStudio or Unplatform page where the assessment is located.

For example, look at the iframe for English Beginner, Unit 1, Lesson 3. This is a word play activity.

<iframe src="https://unplatform.mit.edu/static/oea?unlock\_next=ON\_CORRECT&bank=assessment.Bank%3A57e526c2b3fcec5f10d00d48%40ODL.MIT.EDU&assessment\_offered\_id=assessment.AssessmentOffered%3A57e90d2ab3fcec1309c4e5f7%40ODL.MIT.EDU" title="eb\_u01l03a06 Assessment"/>

Look at just the src variable and break down its components.

src="https://unplatform.mit.edu/static/oea?unlock\_next=ON\_CORRECT&bank=assessment.Bank%3A57e526c2b3fcec5f10d00d48%40ODL.MIT.EDU&assessment\_offered\_id=assessment.AssessmentOffered%3A57e90d2ab3fcec1309c4e5f7%40ODL.MIT.EDU"

## From this iframe src variable you can see several things. (1) what URL was used to get the assessment, (2) what bankId or assessment collection it accesses and (3) which assessment was displayed (offeredId).

## https://unplatform.mit.edu/static/oea?unlock\_next=ON\_CORRECT

## bank=assessment.Bank%3A57e526c2b3fcec5f10d00d48%40ODL.MIT.EDU

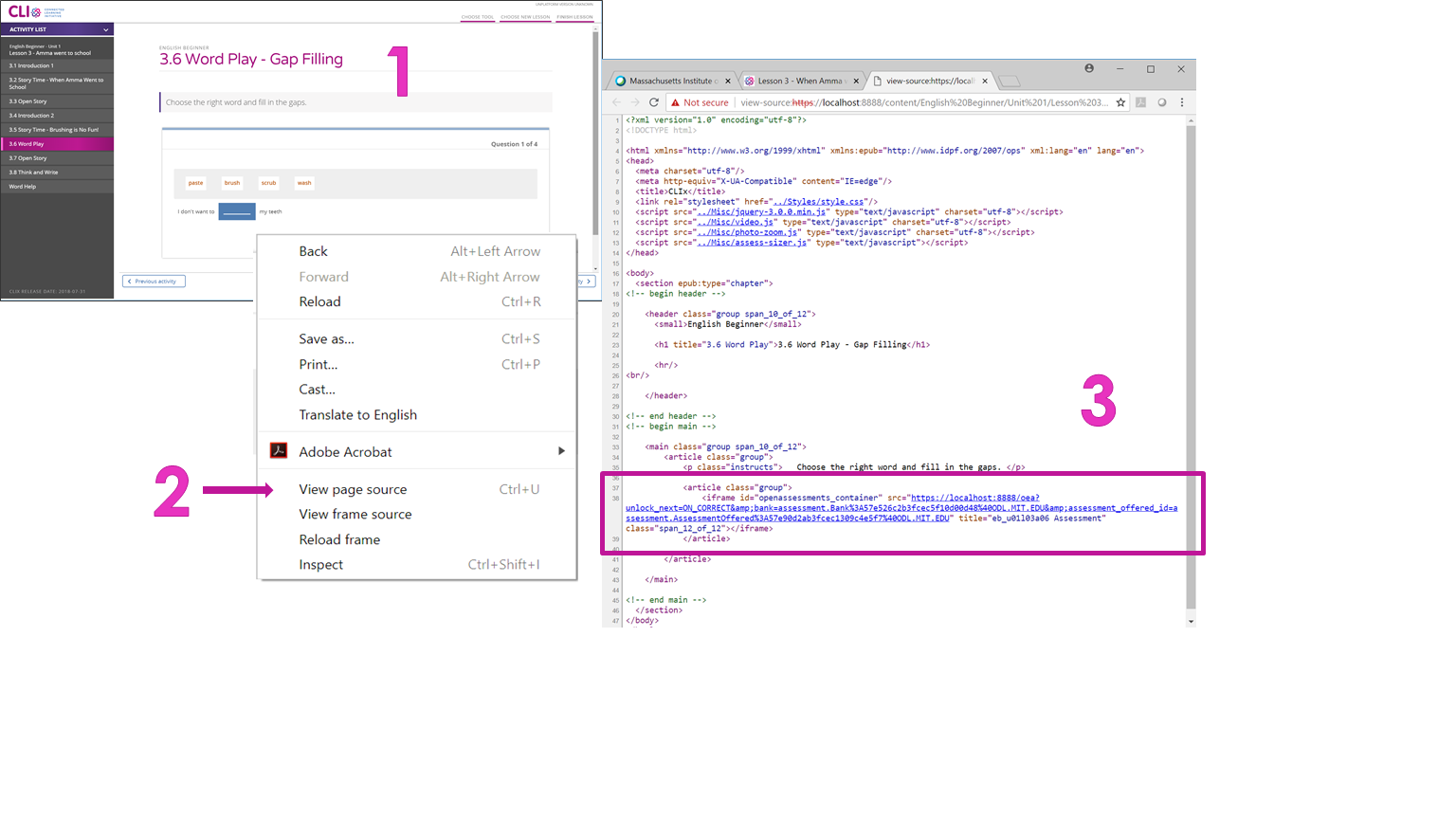
## assessment\_offered\_id=assessment.AssessmentOffered%3A57e90d2ab3fcec1309c4e5f7%40ODL.MIT.EDU

For the Assessment / click-log CSV Script, you will need items (2) the bankId and (3) the offeredId to run the script.

Remember the script needs the following information:

(clix-data) $ python aggregate\_assessment\_data.py --bank=<<bankId>> --offered=<<OfferedId>> --email=<<email address>>

## (clix-data) $ python aggregate\_assessment\_data.py --bank= assessment.Bank%3A57e526c2b3fcec5f10d00d48%40ODL.MIT.EDU --offered= assessment.AssessmentOffered%3A57e90d2ab3fcec1309c4e5f7%40ODL.MIT.EDU --email=kirky@mit.edu



## OST / RKR click-log CSV Script

This script is named get\_default\_log\_entries.py and takes 2 arguments:

1. tool: the name of the tool. Acceptable values are OpenStory or runkittyrun
2. email: where to send the results (only works if script run from MIT network, but you must include this argument, even if not on the MIT network), like kirky@mit.edu

Please be aware this script will not work with the 2017 platform data because of a misconfiguration issue, the tool data was not collected. This script will only work with the 2018 platform data (\*\* The Learning Outcome study in Mizoram used the 2017 platform).

Example:

(clix-data) $ python get\_default\_log\_entries.py --tool=<<tool name>> --email=<<email address>>

(clix-data) $ python get\_default\_log\_entries.py --tool=OpenStory --email=kirky@mit.edu