# Extract TAIR gene IDs

**Marta Kozlowska, Biosystems Analytics Project**

# TAIR

TAIR (The Arabidopsis Information Resource) is a very useful website that contains information on the Arabidopsis genes and genome. Gene IDs used by the website follow a naming convention that is based on the chromosome locations of the genes. For example:

- `AT1G74310` (HEAT SHOCK PROTEIN 101, HOT1)
  - `AT` 1G74310 means that the gene is in Arabidopsis thaliana
  - AT `1` G74310 means that the gene is on chromosome 1
  - AT1 `G74310` is the unique gene identifier

A number of program outputs will contain only these TAIR identifiers, for example RNAseq data. While it would be interesting to look at each gene in a data set and have a description of it immediately, it may not be the most useful for some data sets.For example, I have an RNAseq data set with a list of TAIR gene IDs and their expression levels. I want to identify the expression pattern of a subset of these genes. In particular, I want to look at the expression of genes that typically are upregulated in response to a heat stress. I can go to the TAIR website and find a list of genes that are associated with the `response to heat` identifier. However, after I download that list, it gives me all this extra info:

```
TAIR Accession  Locus   Gene Model  Gene Type   Description Other Name(Type)
Keywords    Is full length cDNA
Gene:1005715766 AT5G67030   AT5G67030.2 protein_coding  Encodes a single copy
zeaxanthin epoxidase gene that functions in first step of the biosynthesis of the
abiotic stress hormone abscisic acid (ABA). Mutants in this gene are unable to express
female sterility in response to beta-aminobutyric acid, as wild type plants do. ABA
DEFICIENT 1; ABA1; ARABIDOPSIS THALIANA ABA DEFICIENT 1; ARABIDOPSIS THALIANA
ZEAXANTHIN EPOXIDASE; ATABA1; ATZEP; IBS3; IMPAIRED IN BABA-INDUCED STERILITY 3; LOS6;
LOW EXPRESSION OF OSMOTIC STRESS-RESPONSIVE GENES 6; NON-PHOTOCHEMICAL QUENCHING 2;
NPQ2; ZEAXANTHIN EPOXIDASE; ZEP chloroplast, chloroplast envelope, FAD binding, LP.02
two leaves visible stage, LP.04 four leaves visible stage, LP.06 six leaves visible
stage, LP.08 eight leaves visible stage, LP.10 ten leaves visible stage, LP.12 twelve
leaves visible stage, abscisic acid biosynthetic process, carpel, cauline leaf,
chloroplast, chloroplast envelope, collective leaf structure, cotyledon, flower,
flower pedicel, flowering stage, guard cell, hypocotyl, inflorescence meristem, leaf
apex, leaf lamina base, mature plant embryo stage, membrane, petal, petal
differentiation and expansion stage, petiole, plant embryo, plant embryo bilateral
stage, plant embryo cotyledonary stage, plant embryo globular stage, plastid, pollen,
response to heat, response to osmotic stress, response to red light, response to water
deprivation, root, rosette leaf, seed, sepal, shoot apex, shoot system, stamen, stem,
sugar mediated signaling pathway, vascular leaf, vascular leaf senescent stage,
xanthophyll biosynthetic process, zeaxanthin epoxidase [overall] activity true
```

Super interesting… But all I wanted was the list of TAIR gene IDs!

Alright, maybe I can obtain the data another way. There is a general resource website for such identifiers as the `response to heat`. These identifiers are also sometimes called GO terms, and you can look up these terms on http://amigo.geneontology.org. But even there, when I download the file, I get all this extra information I don't want or need:

```
AT5G12020|17.6 kDa class II heat shock protein|F14F18.190|F14F18_190
AT5G12020|17.6 kDa class II heat shock protein|F14F18.190|F14F18_190
AT5G41340|ATUBC4|ubiquitin conjugating enzyme 4|MYC6.5|MYC6_5
```

Should I copy and paste only the TAIR gene IDs out of these lists by hand? It should be simple enough when I have two or three, but when my list contains over 300 IDs? I think not!

Solution: write a program to do it for you!

# The Program

Write a python program called `gene_ids.py` that will extract TAIR gene IDs from a file and write those IDs to an outfile. The program will take in the following arguments:

- `-f|--file`: One or more files from which to extract the gene IDs (default: None)
- `-o|--outfile`: The output file to write the gene IDs to (default: out.txt)

When run with -h|--help the program should give the following:

```
$ ./gene_ids.py -h
usage: gene_ids.py [-h] -f FILE [FILE ...] [-o outfile]

Find TAIR locus ID

optional arguments:
  -h, --help            show this help message and exit
  -f FILE [FILE ...], --file FILE [FILE ...]
                        File to find TAIR gene IDs (default: None)
  -o outfile, --outfile outfile
                        Output file name (default: out.txt)
```

I can run the program with my uber convoluted file of genes related to `response to heat` that I got from the TAIR website:

```
$ ./gene_ids.py -f tair_heat.txt
  1: tair_heat.txt
Wrote 5 gene IDs from 1 file to file "out.txt"
```

I get back a list of gene IDs without all the clutter in my out.txt file:

```
$ cat out.txt
AT1G13930
AT5G67030
AT2G22360
AT1G16540
AT3G09440
```

I can also run the program on both of my convoluted files and get back a beautiful list of values:

```
$ ./gene_ids.py -f tair_heat.txt amigo_heat.txt
  1: tair_heat.txt
  2: amigo_heat.txt
Wrote 21 gene IDs from 2 files to file "out.txt"

$ cat out.txt
AT3G24500
AT5G67030
AT1G16540
AT1G13930
AT2G22360
AT3G06400
AT5G41340
AT1G64280
AT3G24520
AT5G03720
AT5G12020
AT2G33590
AT1G54050
AT3G10800
AT3G04120
AT5G12140
AT1G16030
AT3G09440
AT4G14690
AT4G19630
```

And if my file has repeat sequences, the program will only give me one iteration of my gene ID:

```
$ ./gene_ids.py -f amigo_repeat.txt
  1: amigo_repeat.txt
Wrote 5 gene IDs from 1 file to file "out.txt"

$ cat out.txt
AT4G14690
AT5G03720
AT5G12020
AT2G22360
AT5G41340
```

I can change the name of the file the gene IDs are written to as well:

```
$ ./gene_ids.py -f tair_heat.txt -o hothothot.txt
  1: tair_heat.txt
Wrote 5 gene IDs from 1 file to file "hothothot.txt"
```

The output file should:

- have gene IDs listed
- have gene IDs each on a new line
- not have any repeat gene IDs

A passing test suite should look like this:

```
$ make test
pytest -xv test.py
============================== test session starts ==============================
...
collected 9 items

test.py::test_exists PASSED
[ 11%]
test.py::test_usage PASSED
[ 22%]
test.py::test_missing_file PASSED
[ 33%]
test.py::test_bad_file PASSED
[ 44%]
test.py::test_amigo PASSED
[ 55%]
test.py::test_tair PASSED
[ 66%]
test.py::test_two_files PASSED
[ 77%]
test.py::test_repeat_seq PASSED
[ 88%]
test.py::test_outfile PASSED
[100%]

============================== 9 passed in 0.61s ===============================
```