Contents

Python

Motivation (run by the instructor)

This exampale should show you that you can write very powerful tool with only a few lines of code. In this example we plot the gene length distribution of *Salmonella* Typhimurium using the gff file we downloaded for the Unix Shell part. If you do not have the file please download it again.

```
wget ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Bacteria/\
Salmonella_enterica_serovar_Typhimurium_SL1344_uid86645/\
NC_016810.gff
```

Run the final version of the Python script `plot_gff_gene_length_distribution.py` and run it on `NC_016810.gff`. Then open the resulting file NC_016810.gff_gene_length_distribution.pdf.

Python basics in a Jupyter notebook

The following topics are taught in a Jupyter notebook

- Print, literal constants
- Variables
- String format operators
- Data structures: str, int, float, list, dict
- File handling
- New lines, tab etc., regular expression
- `if else` startement
- `for` loop

Writing Python scripts

Here we will create the file `plot_gff_gene_length_distribution.py` from scratch and add successively features.

We read through the lines of the input GFF file an print each line:

```
for line in open("NC_016810.gff"):
    print(line)
```

Call the script like this:

```
python plot_gff_gene_length_distribution.py
```

We have hard coded the file name. Now we want to make it more flexible. The file given as argument should be used. For this we can make use of the module `sys`.

```
import sys

print(sys.argv)

for line in open("NC_016810.gff"):
    # print(line)
    pass
```

This time we call the script by giving the name of the input GFF file as argument.

```
python plot_gff_gene_length_distribution.py NC_016810.gff
```

As output we get the used arguments (including the name of the script!) as a list.

```
['temp.py', 'NC_016810.gff']
```

We can now simply take the second element of that list as argument for `open`:

```
import sys

for line in open(sys.argv[1]):
    print(line)
```

We know that the GFF file is tabular separated file. We can split the column of each line by using the `split` method of the string variavle.

```
import sys

for line in open(sys.argv[1]):
    split_line = line.split()
    print(split_line)
```

We now want to calculate the length of gene which is the gene's end position (column 5) minus the gene's start position (column 4):

```
import sys

for line in open(sys.argv[1]):
    split_line = line.split()
    print(split_line)
    print(split_line[4] - split_line[3])
```

When we call the script in this way we will get an error:

```
['##gff-version', '3']
Traceback (most recent call last):
  File "plot_gff_gene_length_distribution.py", line 5, in <module>
      print(split_line[4] - split_line[3])
      IndexError: list index out of range
```

The header lines are not formated as the entry lines. They start with an "`#`". We use this information to skipt the header lines:

```
import sys
for line in open(sys.argv[1]):
    if line.startswith("#"):
        continue
    split_line = line.split()
    print(split_line)
    print(split_line[4] - split_line[3])
```

Still, we get an error:

```
[...]
Traceback (most recent call last):
  File "plot_gff_gene_length_distribution.py", line 7, in <module>
      print(split_line[4] - split_line[3])
      TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

When splitting the lines Python generates a list of new strings. But our minus operation requires interger (of floats). We have to make this conversion explicitly by using `int`.

```
import sys
for line in open(sys.argv[1]):
    if line.startswith("#"):
        continue
    split_line = line.split()
    print(split_line)
    print(int(split_line[4]) - int(split_line[3]))
```

Now the calculation works. Instead of printing the gene length values we generate a list and store them in there:

```
import sys

gene_lengths = []

for line in open(sys.argv[1]):
    if line.startswith("#"):
        continue
    split_line = line.split()
    curr_gene_length = int(split_line[4]) - int(split_line[3])
    gene_lengths.append(curr_gene_length)

print(gene_lengths)
```

Finally, we plot the values as histogram using `matplotlib`:

```
import sys
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

gene_lengths = []
for line in open(sys.argv[1]):
    if line.startswith("#"):
        continue
    split_line = line.split()
    if split_line[2] == "gene":
        cur_gene_length = int(split_line[4]) - int(split_line[3])
        gene_lengths.append(cur_gene_length)

plt.hist(gene_lengths, bins=100, color="gray")
plt.savefig("{}_gene_length_distribution.pdf".format(sys.argv[1]))
```

A file called `NC_016810.gff_gene_length_distribution.pdf` that contains the plot was generated.