Elias Lehner, Ricardo Orbegozo Medina 21.11.2024

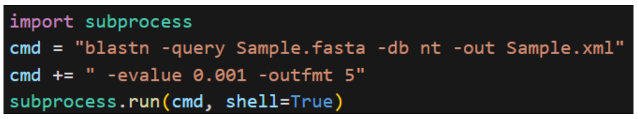
Biopython – Blast

BLAST

- **Sequence Identification**: Identifies unknown DNA/protein sequences
- **Phylogenetic Analysis**: Studies evolutionary relationships
- **Functional Annotation**: Predicts protein functions
- **Efficiency and Speed**: Analyses large datasets quickly

Biopython-BLAST locally

- **Advantages**: Speed, Custom Databases, Privacy
- **Disadvantages**: Requires Installation, Database Setup



```python
import subprocess
cmd = "blastn -query Sample.fasta -db nt -out Sample.xml"
cmd += " -evalue 0.001 -outfmt 5"
subprocess.run(cmd, shell=True)
```

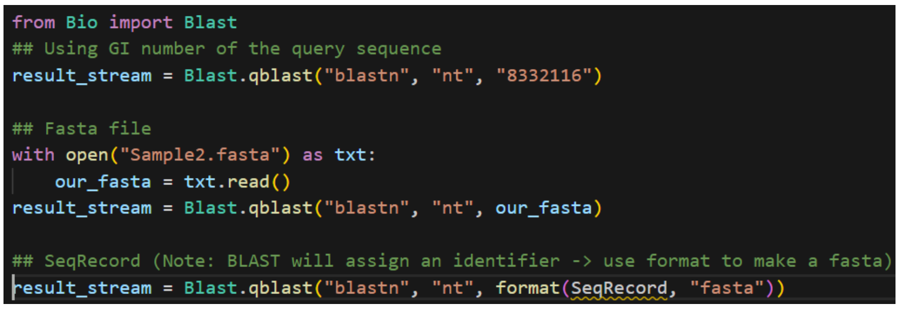
Biopython-BLAST over the Internet

NCBI-Guidelines for using Blast:

- **Server Contact Frequency**: Max once every 10 seconds
- **RID Polling**: Max once per minute
- **Identification**: Use email and tool URL parameters
- **High-Volume Searches**: Run scripts weekends or 9pm-5am ET

Important Arguments:

- **Program**: BLAST program to use (e.g. blastn, blastp)
- **Database**: BLAST database to search against (e.g. nr, nt)
- **Sequence**: Sequence or FASTA file
- **Opional**: format_type (XML, HTML, Text, XML2, JSON2, Tabular)



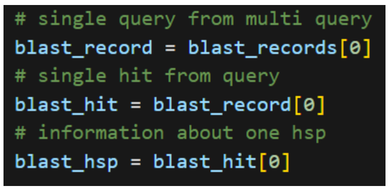
```python
from Bio import Blast
## Using GI number of the query sequence
result_stream = Blast.qblast("blastn", "nt", "8332116")

## Fasta file
with open("Sample2.fasta") as txt:
    our_fasta = txt.read()
result_stream = Blast.qblast("blastn", "nt", our_fasta)

## SeqRecord (Note: BLAST will assign an identifier -> use format to make a fasta)
result_stream = Blast.qblast("blastn", "nt", format(SeqRecord, "fasta"))
```

Biopython-BLAST parsing the output for usage

- **Format**: Only XML, XML2, tabular (no HTML or text)
- **Output**: Can be generated from Biopython, NCBI Website, or local
- **Attention**: Results can only be parsed once!

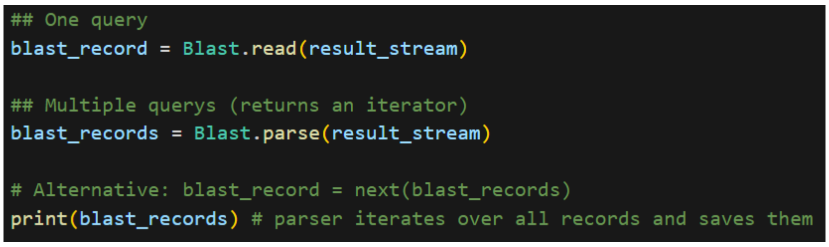
BLAST-object Hierarchy:

- **Bio.Blast.Records**: Contains multiple queries
- **Bio.Blast.Record**: Contains only one query
- **Bio.Blast.Hit**: Contains one hit from a query
- **Bio.Blast.HSP**: Contains all information about the high scoring pairs (HSP)



```python
# single query from multi query
blast_record = blast_records[0]
# single hit from query
blast_hit = blast_record[0]
# information about one hsp
blast_hsp = blast_hit[0]
```

The lower hierarchies can be accessed through indices.
Exception: Bio.Blast.Records is by default an iterator (one time use only, except if it is saved (print))



```python
## One query
blast_record = Blast.read(result_stream)

## Multiple querys (returns an iterator)
blast_records = Blast.parse(result_stream)

# Alternative: blast_record = next(blast_records)
print(blast_records) # parser iterates over all records and saves them
```