**Collections**

Instances of primitive Python types represent individual values. Python also has built-in types that are compound, meaning that they group together multiple objects. These are called collections or containers.

There are three categories of built-in collection types in Python:

1. sets

Unordered, no duplication

1. sequences

Ordered, duplication

* + - range
    - list
    - tuple

1. mappings

Representation for tabular data

* dictionary

**Control statements**

* conditional
* loop
* iteration
* exception handlers
* Conditional

if, if-else, if-elif-else

* Loop control statements

continue

brake

pass

* Iteration
* Exception handlers

try-except

**Modes of opening file**

'r' Open a file for reading. (default)

'w' Open a file for writing.

'a' Open for appending at the end of the file without truncating it.

't' Open in text mode.

'b' Open in binary mode.

'+' Open a file for updating (reading and writing)

**Parsing**

Parsing, syntax analysis or syntactic analysis is the process of analysing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar.

**Algorithm**

A process or set of rules to be followed in calculations or other problem-solving operations

**Primitive data type**

* + bool true/false values
  + int/ long various whole digits
  + float not whole digits - with one or more signs after a dot
  + str any text
  + None

**Biopython**

The Biopython Project is an international association of developers of freely available Python http://www.python.org] tools for computational molecular biology. Provides an online resource for modules, scripts, and web links for developers of Python-based software for bioinformatics use and research. Biopython features include parsers for various Bioinformatics file formats IBLAST, Clustalw, FASTA, Genbank,...), access to online services (NCBI, Expasy,...), interfaces to common and not-so-common programs [Clustalw, DSSP, MSMS...), a standard sequence class, various clustering modules, a KD tree data structure etc. and even documentation.

**Biopython features:**

1. The ability to **parse bioinformatics files** into Python utilizable data structures, including support for the following formats: Blast output - both from standalone and \MWW Blast, Clustalw, FASTA, GenBank, PubMed and Medline, ExPASy files, like Enzyme and Prosite, SCOP, including'dom'and'lin'files, UniGene, SwissProt

2. Files in the **supported formats can be iterated over** record by record or indexed and accessed via a Dictionary interface.

3. Code to deal with **popular on-line bioinformatics destinations** such as:

o NCBI - Blast, Entrez and PubMed services

o ExPASy - Swiss-Prot and Prosite entries, as well as Prosite searches

4. **Interfaces to common bioinformatics programs** such as:

o Standalone Blast from NCBI

o Clustalw alignment program

o EMBOSS command line tools

5. A **standard sequence class that deals with sequences**, ids on sequences, and sequence features.

6**. Tools for performing common operations on sequences**, such as translation, transcription and weight calculations.

7. **Code to perform classification of data** using k Nearest Neighbors, Naive Bayes or Support Vector Machines.

8. **Code for dealing with alignments**, including a standard way to create and deal with substitution matrices.

9. **GUI-based programs** to do basic sequence manipulations, translations, BLASTing, etc.

10**. Integration with BioSQL**, a sequence database schema also supported by the BioPerl and Biojava projects.

**ORF**

An open reading frame (ORF) is the part of a reading frame that has the potential to be translated.

An ORF is a continuous stretch of codons that contain a start codon (usually AUG) and a stop codon (usually UAA, UAG or UGA)

**Complexity**

Computational complexity theory is a branch of the theory of computation in theoretical computer science that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. A computational problem is understood to be a task that is in principle amenable to being solved by a computer, which is equivalent to stating that the problem may be solved by mechanical application of mathematical steps, such as an algorithm.

The goal of computational complexity is to classify algorithms according to their performances.

**Time complexity**

In computer science, the time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input.

The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. When expressed this way, the time complexity is said to be described asymptotically, i.e., as the input size goes to infinity.

For example, if the time required by an algorithm on all inputs of size n is at most 5n3 + 3n for any n (bigger than some n0), the asymptotic time complexity is O(n3).

**Memory complexity**

Memory complexity is the size of work memory used by an algorithm.

**best-case complexity**

This is the complexity of solving the problem for the best input of size n.

**worst case complexity**

This is the complexity of solving the problem for the worst input of size n

**average case complexity**

This is the complexity of solving the problem on an average.

This complexity is only defined with respect to a probability distribution over the inputs.

For instance, if all inputs of the same size are assumed to be equally likely to appear, the average case complexity can be defined with respect to the uniform distribution over all inputs of size n.

Quicksort: Best O(n log n), worst O(n2)

Bubble sort: best O(n), worst O(n2)

