Bioinformatical problem solving with Python

Wednesdays 17:30-19:00, M801

alexander.nater@uni-konstanz.de

Exercises

- Download the vcf file 'Apo_cl2_scaffold2_5Mb_stats.vcf' from Github.
- This vcf file contains a complete header (lines starting with '##').
- Each variant site has now an INFO field with different statistics.
- Expand your previous script to filter variant sites by the DP (>1000) and AC (>10) statistic. Calculate mean individual heterozygosities for the filtered sites only.

Object-oriented programming

- A method of designing software systems where data of a specific identifiable type, as well as the code that operates on that data, are packaged together in a modular, reusable unit known as an object.
- Object: Entity that stores a specific state (variables) and behavior (methods). E.g., cars have state (e.g. current speed) and behavior (e.g. changing gear).
- Class: A class is a template/blueprint that is used to create objects.
- Instance: A unique copy of a class representing an object.

Built-in classes

- my_list = list()
  type(my_list)
  → <class 'list’>

- my_list.extend([1, 2, 3, 4, 5])
  print(my_list)
  → [1, 2, 3, 4, 5]

- dir(my_list)
  → […, 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort’]

- But we can also design our own custom classes!

Exercises

- Write a class to represent a DNA sequence.
- Implement the get_gc_content function as a method working on the stored sequence string.

Custom classes

```
class Sequence:
    """Stores a DNA sequence as string."""
    def __init__(self):
        self.name = "Human gene 123"
        self.dna = "ACTTGCATCGT"

    def get_gc_content(self):
        g_count = self.dna.count('G')
        c_count = self.dna.count('C')
        return (g_count + c_count) / len(self.dna)
```

Using custom classes

- my_sequence = Sequence()
  → Creates a new instance of the ‘Sequence’ class

- dir(my_sequence)

- help(my_sequence)

- print(my_sequence.dna)
  → ACTTGCATCGT

- gc_content = my_sequence.get_gc_content()
  print(round(gc_content, 2))
  → 0.45

Improving custom classes

- The ‘Sequence’ class stores a predefined DNA string.
- It would be much more useful if the DNA sequence can be defined when creating the object instance.
- We can modify the ‘__init__’ method (constructor) to take a string as argument.

Improving custom classes

```
class Sequence:
    """Stores a DNA sequence as string."""
    def __init__(self, my_name, my_dna):
        self.name = my_name
        self.dna = my_dna

    def get_gc_content(self):
        g_count = self.dna.count('G')
        c_count = self.dna.count('C')
        return (g_count + c_count) / len(self.dna)

my_sequence = Sequence("Human gene 123",
"ATGCGGTCT")
```

Exercises

- Extend the “Sequence” class with a method to extract a substring of the DNA sequence.
- Extend the “Sequence” class with a method that returns a dictionary with the counts of each base in the DNA sequence.
- Write a method that returns an instance of the sequence class containing the reverse complement of the DNA sequence.

Inheritance

- Classes can inherit attributes and methods from other classes (base class → derived class).
- Derived classes add functionality or overwrite existing functionality.
- For example:
  Base class ‘Bird’: can eat, fly, walk, etc.
  Derived class ‘Duck’: can also swim, quack, etc.

Inheritance

```
class Bird:
    “””Base class representing a bird.”””
    def __init__(self, my_name):
        self.name = my_name
    def fly(self, dest):
        print(“I’m flying to destination “, dest, “.”, sep=“”)

class Duck(Bird):
    “””Derived class representing a duck.”””
    def quack(self, times):
         print(“I’m quacking “, times, “ times.”, sep=“”)
```

Inheritance

- my_bird = Bird(“Harald”)
  isinstance(my_bird, Bird) → True
  my_bird.fly(“Honolulu”)

- my_duck = Duck(“Donald”)
  isinstance(my_duck, Duck) → True
  isinstance(my_duck, Bird) → True
  isinstance(my_bird, Duck) → False
  my_duck.quack(5)
  my_bird.quack(5)

- issubclass(Duck, Bird) → True
  issubclass(Bird, Duck) → False