Bioinformatical problem solving with Python

Wednesdays 17:30-19:00, M801

alexander.nater@uni-konstanz.de

Python objects

- Everything in Python is an object.
- Objects are entities combining data (attributes) and functions working on this data (methods).
- Variables are just names for objects.
- The type(), id(), dir(), and help() functions provide important information about objects.

Lists

- l = [] or l = list() to create a new empty list
- l[index] to access elements (0-based indexing)
- The list “function” also takes another list as argument
  -> allows proper copying of lists:
  la = [1,2,3]; lb = la; id(la); id(lb); la is lb -> true
  lc = list(la); id(la); id(lc) ; la is lc -> false
- Explore the list methods for useful tools to search and manipulate lists.

Lists

- l[index] to obtain values, l[index] = x to assign values
- Negative indices count from the end of the list.
- l[i:j] to obtain slice from index i up to j (exclusive!)
- l[i:] – slice from index i to the end of the list
- l[:j] – slice from first element to index j
- l[-3:] – slice from the third last element to the end

Lists

- The len() function gives the number of elements in a list.

- Certain arithmetic operators can be used with lists:
  List concatenation: lc = la + lb
  Repetition: lc = la * 3

For loops

- Indentation is important!
- For loops can take any iterable object (lists, tuples, dictionaries, strings, etc.).
- The range “function” can be used to create iterable integer sequences: range(start=0,stop[,step])

Exercise

- Create a list with integer values from 1 to 1000.
- Calculate the sum of all values in the list.
- Create a second list with the squared values of the first list.
- Create a third list with the values from the first list in reversed order.
- Create a fourth list containing the result of adding the values in the first and third list together for each element.

Solutions

-
  ```
  my_list = list(range(1,1001))
  ```

-
  ```
  total = 0
  for value in my_list:
      total += value
  ```
  Better use built-in function ‘sum’:
  ```
  total = sum(my_list)
  ```

-
  ```
  squared_list = []
  for value in my_list:
      squared_list.append(value ** 2)
  ```

Solutions

-
  ```
  rev_list = reversed(my_list)
  ```

-
  ```
  sum_list = []
  for index in range(len(my_list)):
      sum_list.append(my_list[index] + rev_list[index])
  ```

Conditions

- A piece of code that produces a true/false answer
  x == 5, y > 10, len(list) >= 10

- Comparison operators: ==, !=, >, <, >=, <=

- Multiple conditions can be linked with ‘and’ or ‘or’:
  x > 10 and x < 100
  x == 5 or y != 10

- The membership operator ‘in’ checks if element occurs in sequence:
  x in la

Conditional statements

-
  ```
  if condition:
    do something …
  elif condition:
    do something …
  elif condition:
    do something …
  else:
    do something …
  ```

While loop

- Execute loop body as long as condition is true:

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Exercise

- Create a list of the Fibonacci numbers from 1 to 1e6:

1, 1, 2, 3, 5, 8, 13, …

Solutions

- Create a list of the Fibonacci numbers from 1 to 1e6:

1, 1, 2, 3, 5, 8, 13, …

```
fib = [1]
next_num = 1
while next_num <= 1e6:
    fib.append(next_num)
    next_num = fib[-1] + fib[-2]
```