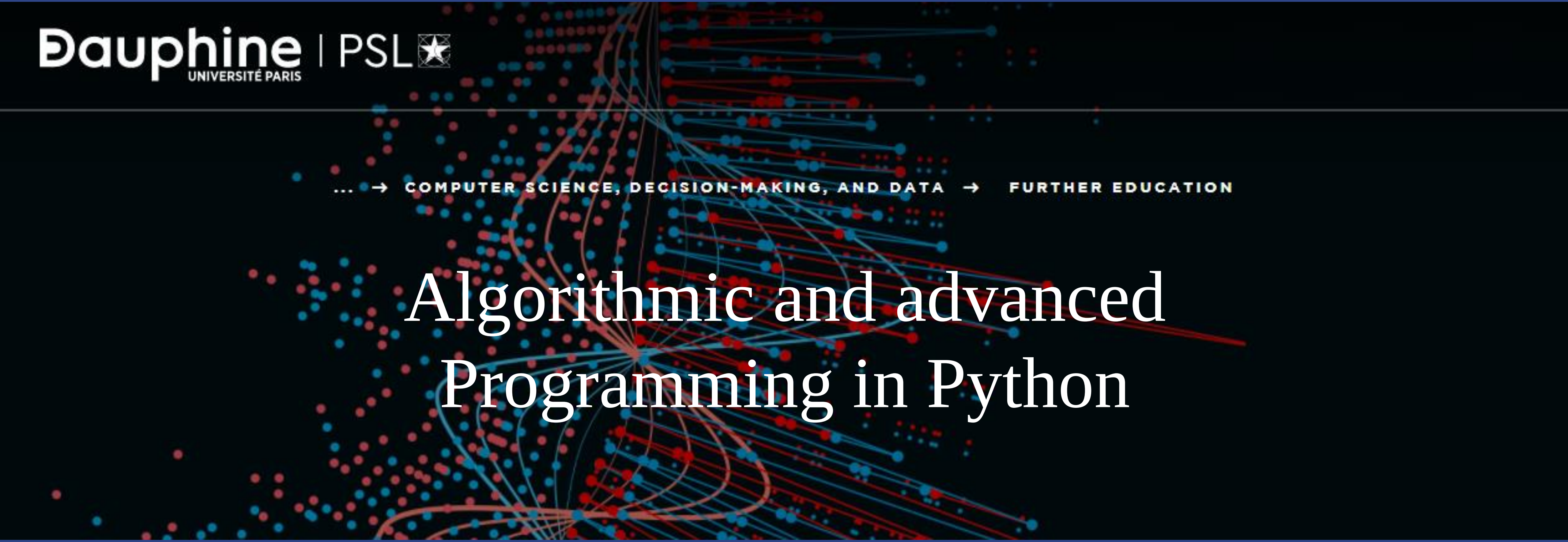
... → COMPUTER SCIENCE, DECISION-MAKING, AND DATA → FURTHER EDUCATION

Algorithmic and advanced Programming in Python

Remy Belmonte remy.belmonte@dauphine.eu

Lab 4

Problem 1: implement Hash table

- In the following you will implement a hash table!

Question 1: hash table implementation

- To create a hash table of given size, say n, we allocate an array of n/L (whose value is usually between 5 and 20) pointers to list, initialized to nil.

- Question 1.1: Finish the constructor by implementing self.slots and self.variable two arrays of size self.size

Question 3.2

- To perform $get/put/delete$ operations, we first compute the index of the table from the given key by using $hashfunction$ and then do the corresponding operation in the linear list maintained at that location. To get uniform distribution of keys over a hash table, maintain table size as the prime number.

- Question 3.2:
  - handle the case self.slots[hashvalue] == None by allocating
  - the key at self.slots[hashvalue]
  - the data at self.data[hashvalue]

Question 3.3

- Question 3.3:
  - In case of collision do a rehash with the following rehash function

    (oldhash + 1) % size

  Implement the rehash function

  ```
  def hashfunction(self, key, size):
  ```

Question 3.4

Implement the end of the code for the put function

Question 4

- Implement the get method: you should do a while loop until found and return the corresponding data

Question 5:

- Validate the code with the following example

```
H = HashTable()
H[54] = "Dauphine "; H[54] = "m1 "; H[26] = "advanced "; H[93] = "programming "; H[44] = "and"
H[55] = "data "; H[20] = "structure "; H[17] = "in "; H[77] = "python"
H[31] = "rocks"
```

- What

```
print('slots aree', H.slots ) should print
print('data are', H.data ) should print
print('show H[20]', H[20]) should print
```

Problem 2: Removing duplicates

- Given an array of integers, give an algorithm for removing the duplicates.
- Logic: Start with the first character and check whether it appears in the remaining part of the string using a simple linear search. If it does not exists in the remaining string, add that character to the $result$ list. Continue this process for character of the given array.

Problem 3

- Can we find any other idea to solve this problem in better time than O(n^2)? Observe that the order of characters in solutions do not matter.

- Solution: Use sorting to bring the repeated characters together. Finally scan through the array to remove duplicates in consecutive positions.

- What is the complexity?

Problem 4: A even better solution

- Can we solve this problem in a single pass over given array?

- Hint: We can use hash table to check whether a character is repeating in the given string or not. If the current character is not available in hash table, then insert it into hash table and keep that character in the given string also. If the current character exists in the hash table then skip that character.

In python use a set!

Problem 5: checking arrays

- Given two arrays of unordered numbers, check whether both arrays have the same set of numbers?

Problem 6: a better solution

- Can we improve the time complexity of Problem-5?

Problem 7: implement Bloom filters