Algorithms in Python

Sobia Amjad

October 1, 2020

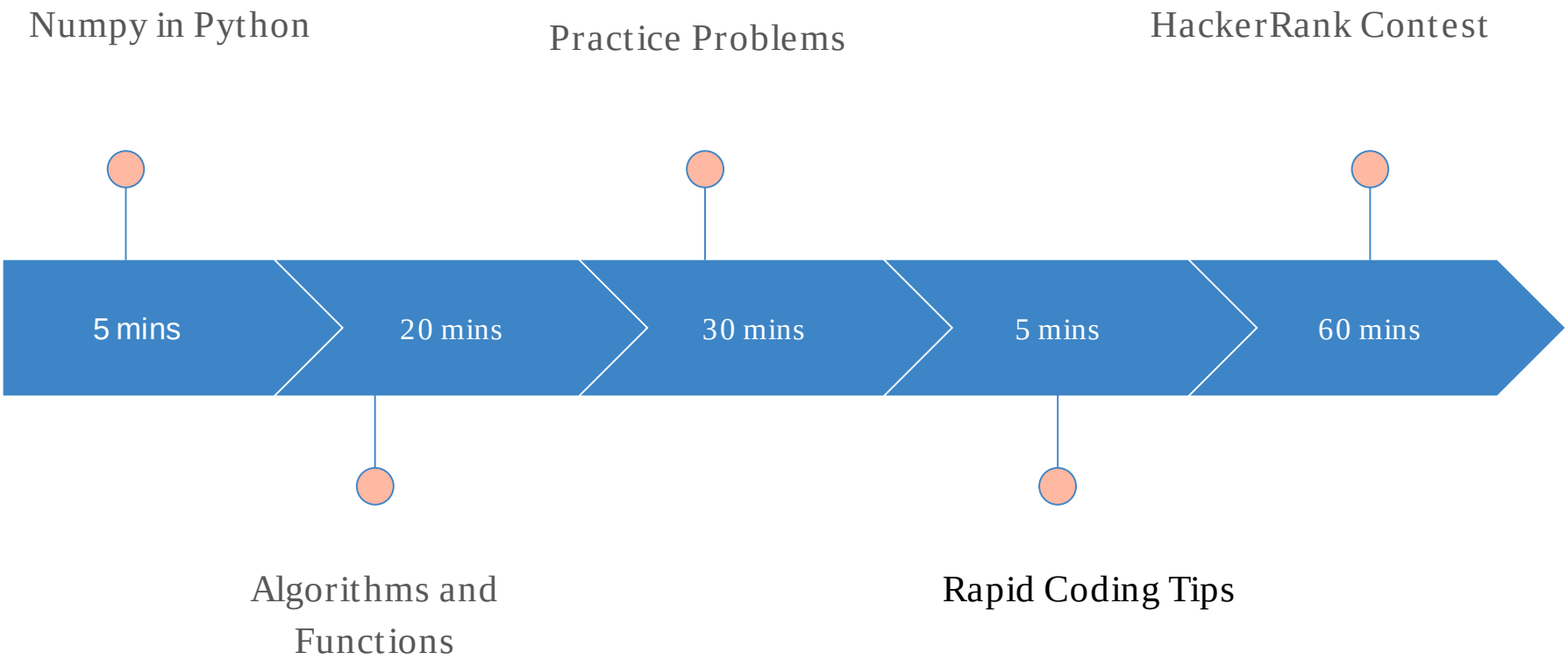
IEEE Student Branch
UNIVERSITY OF MELBOURNE

IEEE Women in Engineering
WIE

IEEE WOMEN IN ENGINEERING
The University of Melbourne

Lecture Contents

1. Numpy in Python
2. Algorithms and Functions
3. Practice Problems
4. Rapid Coding Tips
    a. Team Management
    b. Cheatsheet
5. HackerRank Contest


Numpy in Python
Practice Problems
HackerRank Contest
5 mins
20 mins
30 mins
5 mins
60 mins
Algorithms and Functions
Rapid Coding Tips

Contents

- **Numpy in Python**
- Algorithms & Functions
- Practice Problems
- Rapid Coding
- HackerRank Contest

NumPy in Python

NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster that traditional Python lists.

NumPy, created in 2005 by Travis Oliphant, is an open source project for free usage.

NumPy in Python

Complex Operations

- Arrays
- Vectors
- Matrices

```
import numpy as np
A = np.array([3, 6., 9, 12])  # 1-dimensional array
B = np.array([(2,4),(6,8)]) # 2-dimensional array
C = np.ones((2,3))  #  2x3 array with all values 1
D = np.full((2,3),5) # 2x3 array with all values 5
E = np.random.rand(2,3)*10 # 2x3  random 0–10
F = np.empty([2, 2], dtype=float) # 2x2  empty
>>> A/3       → [1. 2. 3. 4.]#Array Division
>>> B/2    → ?
>>> type(A) → <class 'numpy.ndarray'>
>>> A[1:2]) → [6]                          #Array Slicing
>>> C.shape → (2,3)
>>> A.dtype → float64
>>> C.T →  3x2 array #Transpose
>>> A.reshape(2,2) # Reshape A to 2x2 array
```

NumPy in Python

Complex Operations

- Arrays
- Vectors
- Matrices

```
A = [[1, 0], [0,-2]]

B = np.array([[4, 1], [2, 2]])

>>> np.dot(A, B)                                    #Matrix Product

Ans: array([[4, 1], [-4, -4]])

>>> print(type(A)) → <class 'list'>

>>> A = np.array(A) # convert A to numpy array

>>> np.delete(C,2,axis=0) # Deletes row #2 of C

>>> np.delete(C,2,axis=1) # Deletes column #2

>>> np.concatenate((A,B),axis=0) # rows B to A

>>> np.concatenate((A,B),axis=1) # cols B to A

>>> A[1,1]=10 # update element in A

>>> A[A > 1] → array([2])

>>> np.abs(A) # absolute values

>>> np.floor(A) # Rounds down to the nearest int
```

Contents

- Numpy in Python
- **Algorithms & Functions**
- Practice Problems
- Rapid Coding
- HackerRank Contest

Algorithms

1. **Slicing an array:** Contents of an array object can be accessed and modified by indexing or slicing.

2. **Sorting an array:** Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

Algorithms

Slicing

- A Python slice object is constructed by giving **start**, **stop**, and **step** parameters to the built-in slice function.
- The step allows you to specify slicing (sampling) frequency e.g. slice only every other item.
- This algorithm returns a slice object that can be used used to slice strings, lists, tuple.
- This slice object is usually passed to the array to extract a part of array.

In-built Function

Slicing

slice(start, stop, step)

```
# Get a substring from the given string
py_string = 'Python'
>>> py_string[slice(3)] → ‘Pyt’
>>>  py_string[:3]        → ‘Pyt’
>>> py_string[slice(1,16,2)] → ‘yhn’

-------------------------------------
Case I
a = ("a", "b", "c", "d", "e", "f", "g", "h")
>>> a[ slice(3,5)] → ???
>>> a[3:5]          → ???
Case II
>>>a[slice(0,8,3)] → ???
```

Algorithms

1. **Slicing an array:** Contents of an array object can be accessed and modified by indexing or slicing.

2. **Sorting an array:** Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

Algorithms

Sort function in Python

```
# Lists:
 a=[2,3,5,1,7,2]
>>> sorted(a,reverse=True) → [7, 5, 3, 2, 2, 1]
-----------------------------
# array:
B = np.array([[4, 1], [2, 2]])
>>> B.sort(axis=0, kind='quicksort')
>>> B[np.argsort(B[:, 1])]
-----------------------------
# dictionary
>>> sorted(car.keys())
Dict = {  'e': 72,  'a': 48,   'c': 41}
>>> sorted(Dict .items(), key=lambda x: x[1],
reverse=True)
Ans: [('e', 72), ('a', 48), ('c', 41)]
```

Algorithms

Quick Sort

Divide and Conquer algorithm

- The key process in quickSort is partition
- Pick an element as pivot and partitions the given array around the picked pivot
- Given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (< x) before x, and put all greater elements (> x) after x.
- Pivot could be first element, last element, random element or median (implemented here)
- Other sorting algorithms: Merge & Heap Sort
- QuickSort is faster in practice

Algorithms

Quick Sort - Pseudocode

```
quickSort(arr[])
{
  pivot = middle element
  left = array with values lesser than pivot
  middle = array with values equal to pivot
  right = array with values greater than pivot

  return (concatenate(left, middle, right))
}
```

User-defined Function

Quick Sort

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

print(quicksort([3,6,8,10,1,2,1]))
```

Ans: [1, 1, 2, 3, 6, 8, 10]

Contents

- Numpy in Python
- Algorithms & Functions
- **Practice Problems**
- Rapid Coding
- HackerRank Contest

Valid Palindrome

```
s = 'radkar'

def solution(s):
    for i in range(len(s)):
#         print(s[:i], s[i+1:])
        t = s[:i] + s[i+1:]
        if t == t[::-1]: return True

    return s == s[::-1]

solution(s)
```

Monotonic Arrays

```
# Given an array of integers, determine whether the array is monotonic or not.
A = [6, 5, 4, 4]
B = [1,1,1,3,3,4,3,2,4,2]
C = [1,1,2,3,7]

def solution(nums):
    return (all(nums[i] <= nums[i + 1] for i in range(len(nums) - 1)) or
            all(nums[i] >= nums[i + 1] for i in range(len(nums) - 1)))

print(solution(A))
print(solution(B))
print(solution(C))
```

Ans: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]

Prime Number

```
def isprime(n):
    '''check if integer n is a prime'''

    # make sure n is a positive integer
    n = abs(int(n))

    # 0 and 1 are not primes
    if n < 2:
        return False

    for i in range(2, n):
        if n % i == 0:
            return False
        return True

    return True
```

Prime Numbers Array

```
n = 35
def solution(n):
    prime_nums = []
        for num in range(n):
         if num > 1: #all prime nos are greater than 1
          for i in range(2, num)
# if the modulus == 0 is means that the number can be divided by a number preceding it
           if (num %i) == 0
             break
           else:
                  prime_nums.append(num)
return prime_nums
```

Ans: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]

Contents

- Numpy in Python
- Algorithms & Functions
- Practice Problems
- **Rapid Coding**
- HackerRank Contest

Rapid Coding Tips

BASICS

Make Your Fundamentals Clear. Aim For the Flow.
Learn By Doing, Practicing and Not Just Reading.

TEAM

Team Work: Share, Discuss and Ask For Help.
Practice Coding as a Team. Soft Skills Matter.

CODE

Comment and Reflect. Improve Debugging Skills.
Create the Right Work Environment.

FOCUS

References

1. https://cs231n.github.io/python-numpy-tutorial/
2. https://www.slideshare.net/SupunAbeysinghe/preparing-for-ieeextreme-120-amp-mora-xtreme
3. https://github.com/topics/ieeextreme
1. https://www.bigocheatsheet.com/?fbclid=IwAR2NyCWf4L_6hrC8-WhH8vScaeJy_voSZbHSQoH_ByZEsBnijnV8_ueEPEA
1. https://hackerbits.com/data/top-10-data-mining-algorithms-in-plain-english/

HACKERRANK CONTEST