

# Python Базовый

Работа с рекурсией


# Python Базовый

## Introduction



**Бондаренко Кирилл**

Senior Data scientist, CreatorIQ

 [profile.php?id=100011447245832](https://www.facebook.com/profile.php?id=100011447245832)

 [kirill-bond/](https://www.linkedin.com/in/kirill-bond/)

 [@bond.kirill.alexandrovich](https://www.telegram.com/@bond.kirill.alexandrovich)



# Python Базовый

## Тема урока

## Работа с рекурсией

# Python Базовый

## План урока

1. Что такое рекурсия в программировании
2. Простые варианты применения рекурсии
3. Бинарное дерево
4. Решение задач

# Python Базовый

## Что такое рекурсия

**Рекурсия** - когда функция вызывает саму себя (простая рекурсия) или когда она вызывает саму себя через другую функцию (косвенная рекурсия).

Два самых важных момента в рекурсии: изменение состояния и условие выхода.

Если первое условие не соблюдено, код теряет логику, если второе условие не соблюдено, то будет переполнение стека и ошибка.

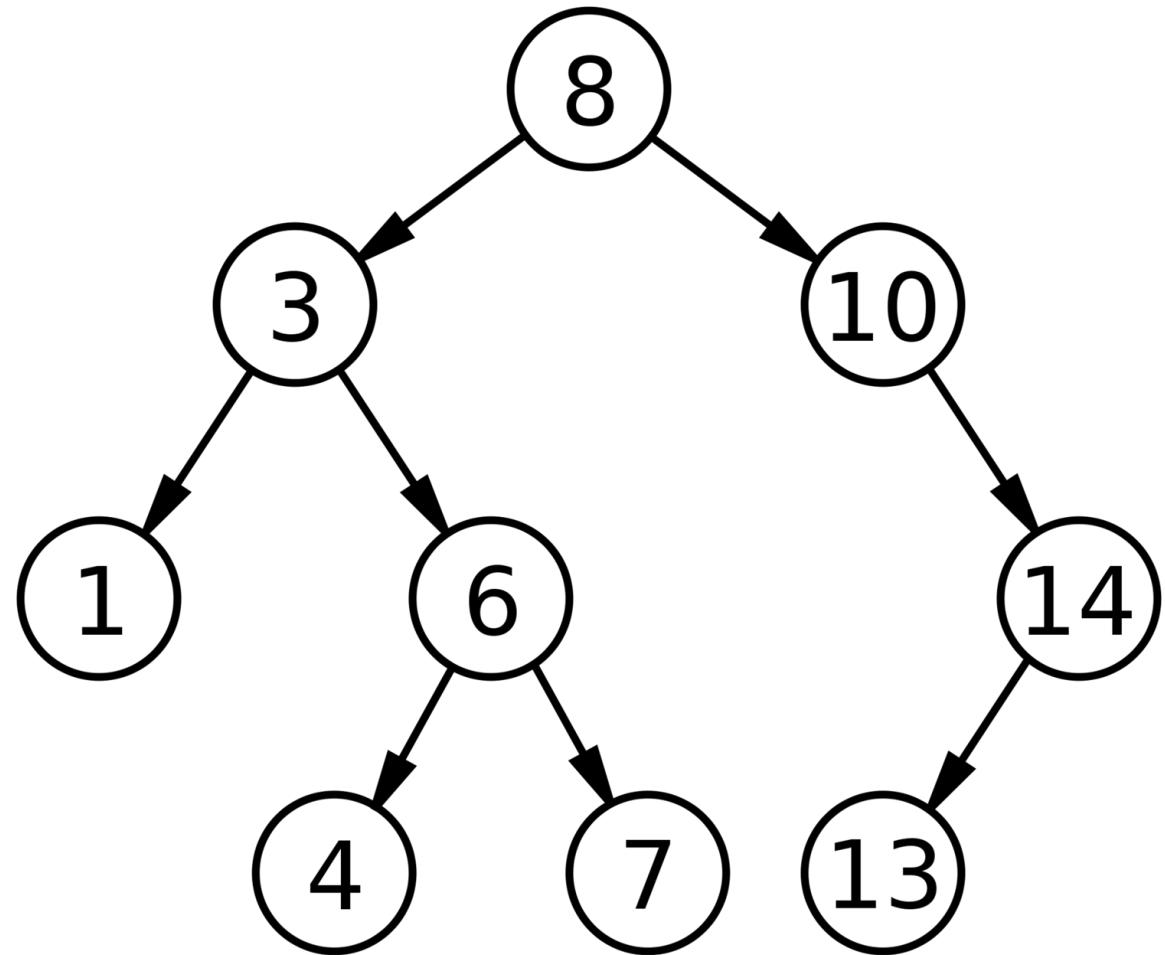
```
1      a = [1, 2, 3, 4, 5]
2
3
4      def iterate(array):
5          for elem in array:
6              print(elem)
7
8
9      def recur_iter(array, index=0):
10         if index < len(array):
11             print(array[index])
12             index += 1
13             recur_iter(array, index)
14
15
16     iterate(a)
17     recur_iter(a)
18
```

# Python Базовый

## Где нужна рекурсия

Структура данных дерево - хороший пример того, зачем нужна рекурсия.

**Бинарное дерево** — это иерархическая структура данных, в которой каждый узел имеет значение (оно же является в данном случае и ключом) и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне (не являющийся чьим либо потомком) называется корнем. Узлы, не имеющие потомков (оба потомка которых равны NULL) называются листьями.

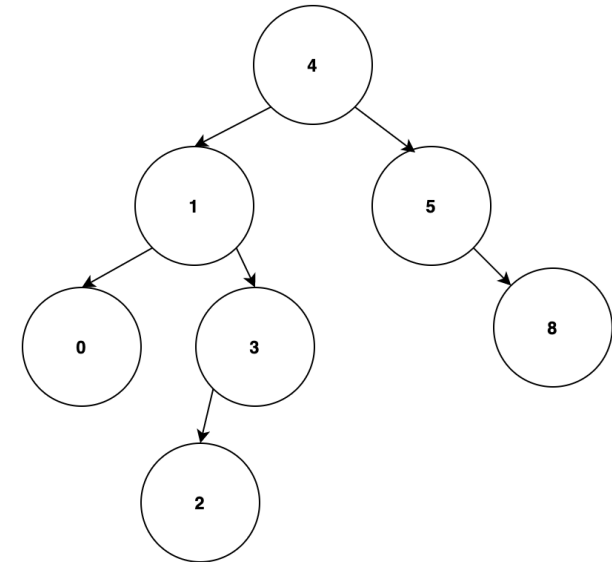


# Python Базовый

## Бинарное дерево на Python

```
1 class Node:
2     def __init__(self, data):
3         self.left = None
4         self.right = None
5         self.data = data
6
7     def insert(self, data):
8         if self.data:
9             if data < self.data:
10                 if self.left is None:
11                     self.left = Node(data)
12                 else:
13                     self.left.insert(data)
14             elif data > self.data:
15                 if self.right is None:
16                     self.right = Node(data)
17                 else:
18                     self.right.insert(data)
19             else:
20                 self.data = data
21
22     def print_tree(self):
23         if self.left:
24             self.left.print_tree()
25         print(self.data)
26         if self.right:
27             self.right.print_tree()
28
29
```

```
30 array = [4, 5, 1, 3, 2, 8, 0]
31 root = None
32 for i in range(len(array)):
33     if i == 0:
34         root = Node(array[i])
35     else:
36         root.insert(array[i])
37 root.print_tree()
38
```



# Python Базовый

## Задачи

1. Реализовать функцию поиска значения в бинарном дереве. Если значение найдено, то функция вернет True и предварительно выведет в консоль, что значение найдено, в обратном случае - False и выведет в консоль, что такого значения в дереве нету.
2. Дано натуральное число N, вывести все натуральные числа от 1 до N.
3. Дано натуральное число N. Выведите слово YES, если число N является точной степенью двойки, или слово NO в противном случае.
4. Дано натуральное число N. Вычислите сумму его цифр.



# Python Базовый

## Решение

1

```
7 def search(self, data):
8     if data == self.data:
9         print("{} in tree".format(data))
10        return True
11    if data < self.data:
12        if self.left is None:
13            print("{} not in tree".format(data))
14            return False
15        return self.left.search(data)
16    if self.right is None:
17        print("{} not in tree".format(data))
18        return False
19    return self.right.search(data)
20
```

3

```
10 def check_pow_2(n):
11     if n == 1:
12         print("YES")
13     else:
14         if n % 2 == 0:
15             check_pow_2(n // 2)
16         else:
17             print("NO")
18
19 check_pow_2(128)
21
```

2

```
1 def values(n, current=1):
2     if current <= n:
3         print(current)
4         values(current=current + 1, n=n)
5
6 values(n=10)
8
```

4

```
23 def sum_val(num, res=0):
24     if not num:
25         return res
26     res += num % 10
27     num //= 10
28     return sum_val(num, res)
29
30
31 s = sum_val(125)
32 print(s)
33
```

# Информационный видеосервис для разработчиков программного обеспечения



# Проверка знаний

TestProvider.com



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://TestProvider.com)

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

# Python Базовый

Спасибо за внимание! До новых встреч!



Бондаренко Кирилл  
Senior Data scientist, CreatorIQ

