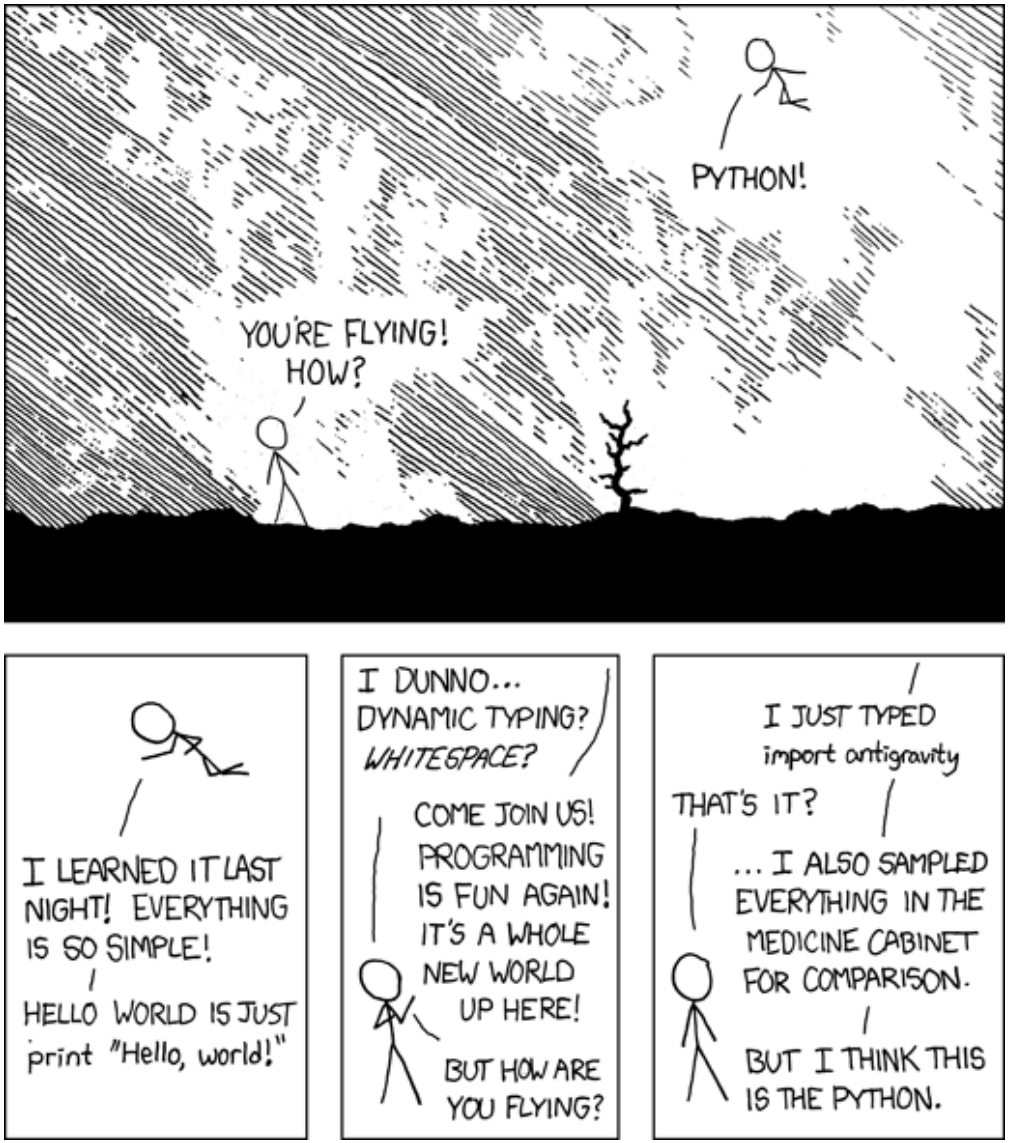
Python

Sukanta Bhattacharjee

Indian Statistical Institute, Kolkata

Contents



http://www.xkcd.com/353

Why Python?

- Reduced development time
  - Program is 2-10X shorter than C,C++,Java

C

```c
#include <stdio.h>
int main(){
   printf("Hello World\n");
   return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;
int main(){
   cout << "Hello World"<<endl;
   return 0;
}
```

Java

```java
public class Hello{
    public static void main(String[] args){
        System.out.println("Hello, World");
    }
}
```

Python

```python
print 'Hello World'
```

How simply we can program

```
# Swaping
a = 1
b = 'DFS Lab'
print 'Before swap:',a,b
a,b = b,a # swap !!
print 'After swap:',a,b
```

How simply we can program

```
# Sorting
student_tuples = [('john', 'A', 15),('dave', 'B', 10),('guido','C',19)]
sorted(student_tuples, key=lambda student: student[2])   # sort by age
sorted(student_tuples, key=lambda student: student[0])   # sort by name

# Multiplication table
[2*i for i in range(1,11)]
```

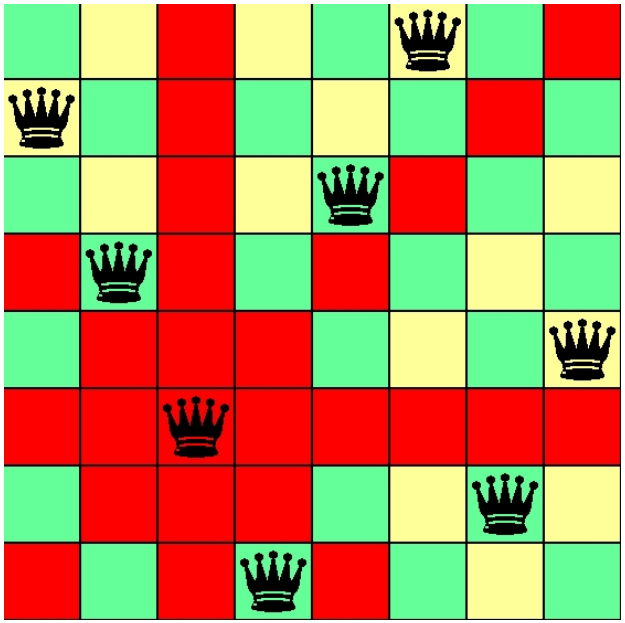
How simply we can program

```
# Swaping
a = 1
b = 'DFS Lab'
print 'Before swap:',a,b
a,b = b,a # swap !!
print 'After swap:',a,b

# Sorting
student_tuples = [('john', 'A', 15),('dave', 'B', 10),('guido','C',19)]
sorted(student_tuples, key=lambda student: student[2])   # sort by age
sorted(student_tuples, key=lambda student: student[0])   # sort by name

# Multiplication table
[2*i for i in range(1,11)]

# Summation of first n natural numbers
def add(x,y):
    return x+y

reduce(add, range(1, 11))
```

Don't worry !!

8-Queen problem (C. F. Gauss, 1850)

Place 8 chess queens on an 8×8 chessboard so that no two queens attack each other.

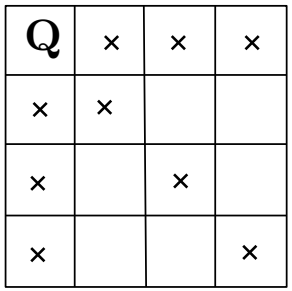


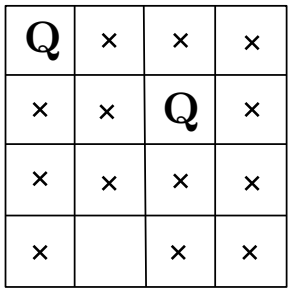
A solution

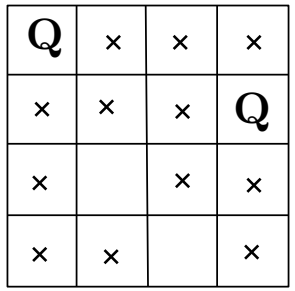
How do we solve?

Trial and Error, Backtracking

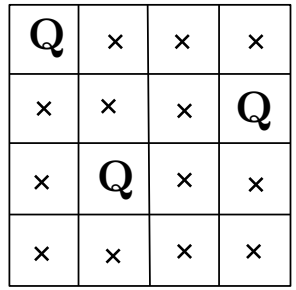
Solving 4-Queen using backtracking

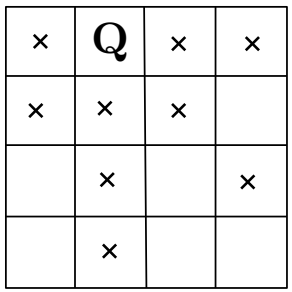




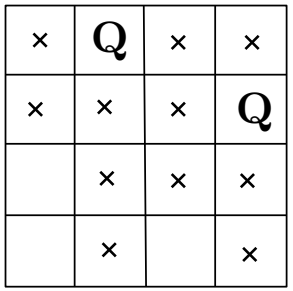


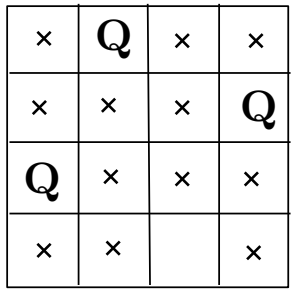
backtrack

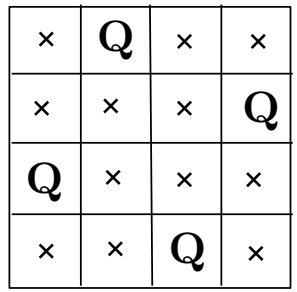




backtrack







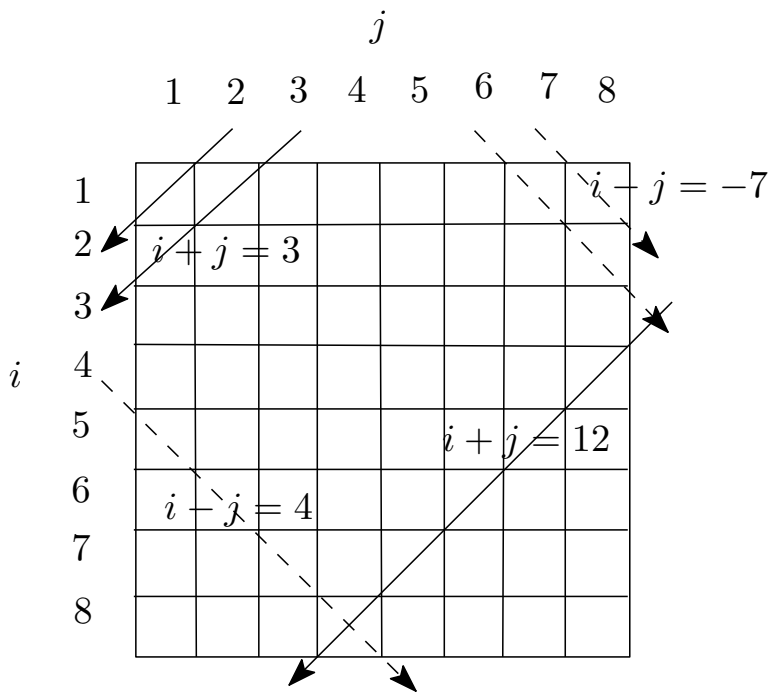
solution

How to design data structures?

- Position of i_{th} queen can be easily determined by its column index j, i.e., $queen[i] = j$
- No two queens can be placed on same column $\forall i \neq j,\ queen[i] \neq queen[j]$
- How to encode diagonals blocked by a queen?

How to design data structures?

- Position of i_{th} queen can be easily determined by its column index j, i.e., $queen[i] = j$
- No two queens can be placed on same column
  $\forall i \neq j$, $queen[i] \neq queen[j]$
- How to encode diagonals blocked by a queen?



Implementation of 8-Queen using python

```
import sys

rightDiagonal = [False] * 15
leftDiagonal = [False] * 15
placed = [False] * 8
queen = [0] * 8

def placeQueen(i):
  global rightDiagonal, leftDiagonal, placed, queen
    for j in range(8):
      # safe condition for placing queen i to j column
        if ( ?? ):
            # set queen i to location j
            # ??
            #
            if i < 7:
                placeQueen(i + 1)
            else:
                # print Solution
                printSolution(queen)
                sys.exit(0)  # Comment this line for all solutions
            # Backtrack
            # remove queen i from column j
            # ??
            #
placeQueen(0)
```

Implementation of 8-Queen using python

```
import sys

rightDiagonal = [False] * 15
leftDiagonal = [False] * 15
placed = [False] * 8
queen = [0] * 8

def placeQueen(i):
  global rightDiagonal, leftDiagonal, placed, queen
    for j in range(8):
      # safe condition for placing queen i to j column
        if ( placed[j] == False and rightDiagonal[i - j + 7] == False
                                and leftDiagonal[i + j - 2] == False ):
            # set queen i to location j
            # ??
            #
             if i < 7:
                 placeQueen(i + 1)
             else:
                 # print Solution
                 printSolution(queen)
                 sys.exit(0)  # Comment this line for all solutions
            # Backtrack
            # remove queen i from column j
            # ??
            #
placeQueen(0)
```

Implementation of 8-Queen using python

```
1  import sys
2
3  rightDiagonal = [False] * 15
4  leftDiagonal = [False] * 15
5  placed = [False] * 8
6  queen = [0] * 8
7
8  def placeQueen(i):
9    global rightDiagonal, leftDiagonal, placed, queen
10     for j in range(8):
11         if (placed[j] == False and rightDiagonal[i - j + 7] == False
12                                and leftDiagonal[i + j - 2] == False):
13             queen[i] = j
14             placed[j] = True
15             rightDiagonal[i - j + 7] = True
16             leftDiagonal[i + j - 2] = True
17             if i < 7:
18                 placeQueen(i + 1)
19             else:
20                 # print Solution
21                 printSolution(queen)
22                 sys.exit(0)  # Comment this line for all solutions
23             # Backtrack
24             placed[j] = False
25             rightDiagonal[i - j + 7] = False
26             leftDiagonal[i + j - 2] = False
27
28 placeQueen(0)
```

Why Python?

- Improved program maintenance
  - Code is extremely readable
- Less training
  - Language is very easy to learn

What is it used for?

- Web and Internet development
- Database access
- Desktop GUIs
- Scientific and numeric
- Network programming
- Software development
- Game and 3D graphics

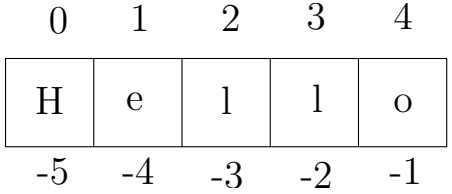
`http://www.python.org/about/apps/`

Important facts about Python

- Open source
- General purpose
- Interpreted rather than compiled
- Supports Procedural, Object Oriented, functional (not fully supported) programming paradigm
- Dynamic typing, static scoping
- Garbage collected

An informal introduction to python

- Numbers
  - $2 + 2 = 4$
  - $(50 - 5 * 6)/3 = 6$
  - $a = 3.0 + 4.0j$, a.real = ?, a.imag = ?
- Strings
  - str = 'Hello'
  - len(str) = 5



  - s = str + ' ' +'World'?
  - s = s*2?
  - s = str[1:3]?, s = str[2:]?, s = [-1:]?, s = [:100]?

Quiz? s[:5] + s[5:] = ?

Informal introduction (Cont. . .)

- Lists
  - Flexible array ▶ L = [99, 'bottles of beer', ['on', 'the', 'wall']]
  - Same operators as for strings ▶ L + M, L*3, L[2:], len(L)
  - Item and slice assignment
    L[0] = 98, L[1:2] = ['bottles','of','beer']
    ▶ L = [99, 'bottles', 'of', 'beer', ['on', 'the', 'wall']]
    del L[-1] ▶ L = [99, 'bottles', 'of', 'beer']
  - More list operations
    L = range(5) ▶ L = [0,1,2,3,4]
    L.append(5) ▶ L = [0,1,2,3,4,5]
    L.pop() ▶ L = [0,1,2,3,4]
    L.insert(0,5.5) ▶ L = [5.5,0,1,2,3,4]
    L.pop(0) ▶ L = [0,1,2,3,4]
    L.reverse() ▶ L = [4,3,2,1,0]
    L.sort() ▶ L = [0,1,2,3,4]

Control statements I

```
# if statement

x = int(raw_input("Please enter an integer: "))

if x < 0:
   x = 0
   print 'Negative changed to zero'
elif x == 0:
   print 'Zero'
elif x == 1:
   print 'Single'
else:
   print 'More'

#for statement

words = ['cat', 'window', 'defenestrate']
for w in words:
   print w, len(w)

# continue statement

for num in range(2, 10):
   if num % 2 == 0:
      print "Found an even number", num
      continue
   print "Found an odd number", num

```

Control statements II

```
30
#The range() function
32
range(10)                #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
34 range(5, 10)           #[5, 6, 7, 8, 9]
range(0, 10, 3)          #[0, 3, 6, 9]
36 range(-10, -100, -30) #[-10, -40, -70]

38 #break statement, and else clauses on Loops

40 for n in range(2, 10):
  for x in range(2, n):
42   if n % x == 0:
       print n, 'equals', x, '*', n/x
44     break
  else:
46   # loop fell through without finding a factor
     print n, 'is a prime number'
```

Variables

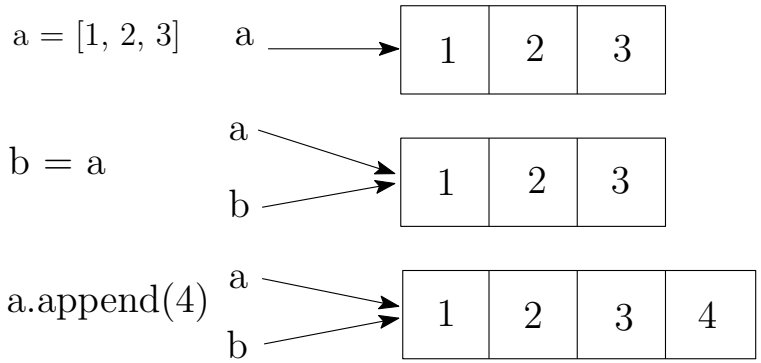
- No need to declare
- Should be initialized
  - Use of uninitialized variables raises exception
- No type

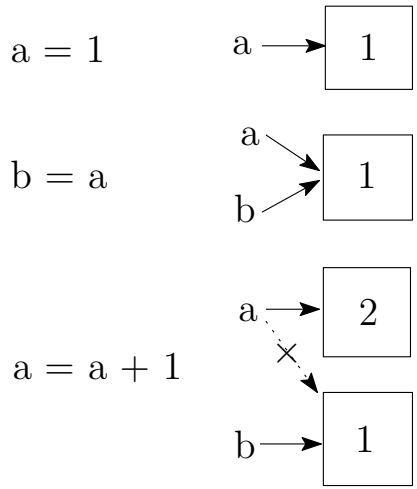
```
1 if friendly:
    greeting = 'Hello World'
3 else:
    greeting = 12**144
5 print greeting
```

- Everything is a variable
  - Functions, modules, classes

Reference semantics

- Assignment manipulates references
  - x = y does not make a copy of y
  - x = y makes x reference the object y references
- Very useful, be aware !!
- Changing a shared list/variable





Functions, procedures

def name (*arg*1, *arg*2, ...):
    ''' Documentation ''' # optional
    statements
return # from procedure
return expression # from function

```
1 def gcd(a,b):
    '''greatest common divisor'''
3   while a != 0:
      a,b = b%a,a   # parallel assignment
5   return b
```

Scope of variables

- Static scope
- Two types of variables
  - Local variables - defined inside a function
  - Global variable - defined outside of all functions

```
1  total = 0 # This is global variable.
   def sum( arg1, arg2 ):
3     #global total
      total = arg1 + arg2 # Here total is local variable.
5     print "Inside sum: total = %d" %total
      return total
7
   sum( 10, 20 )
9  print "Outside sum: total = %d" %total

11 #Output
   #Inside sum: total =  30
13 #Outside sum: total =  0
```

- Accessing global variable - use *global variable-name*

Parameter passing

- Python uses "Call by object reference" parameter passing mechanism !!
- Passing immutable arguments - integers, strings e.t.c
  - Acts like "call by value"

```
1 def func(x):
      print "Before assignment: id(x) = %d" %id(x)
3     x = 4
      print "After assignment: id(x) = %d" %id(x)
5
x = 5
7 print id(x)    # 35940184
  func(x)        # Before assignment: id(x) = 35940184
9                # After assignment: id(x) = 35940208
  print id(x)    # 35940184
```

Parameter passing

- Passing mutable arguments - integers, strings e.t.c
  - Acts like “call by reference”

```
1  def func( mylist ):
       mylist.append([1,2,3,4])
3      print "Inside function (1): " + str(mylist)
       mylist += [7,8,9,10]
5      print "Inside function (2): " + str(mylist)
       mylist = [12,13,14,15]
7      print "Inside function (3): " + str(mylist)
       return
9
   mylist = [10,20,30]
11 func( mylist )
   print "Outside function: " + str(mylist)
13
   # Inside function (1): [10, 20, 30, [1, 2, 3, 4]]
15 # Inside function (2): [10, 20, 30, [1, 2, 3, 4], 7, 8, 9, 10]
   # Inside function (3): [12, 13, 14, 15]
17 # Outside function: [10, 20, 30, [1, 2, 3, 4], 7, 8, 9, 10]
```

Quiz? What about function passed as parameter?

- Home works
  - Default arguments, Variable length arguments, Anonymous Functions

Tuples

- Immutable list, i.e. a tuple cannot be changed in any way once it has been created
- t = ("tuples", "are", "immutable")
- It is not possible to assign to the individual items of a tuple
- It is possible to create tuples which contain mutable objects, such as lists
- t = (1, 2, [1])
- t[0] = -1

  # TypeError: 'tuple' object does not support item assignment
- t[2].append(2) # (1, 2, [1, 2])

Dictionaries

- Hash tables, "associative arrays"
  - d = dict('jack': 4098, 'sape': 4139)
- Lookup
  - d['jack'] # 4098
  - d['guido'] # raises KeyError exception
- Insert, delete, overwrite
  - d['guido'] = 4067

  # {'sape': 4139, 'jack': 4098, 'guido': 4067}
  - del d['jack'] # {'sape': 4139, 'guido': 4067}
  - d['sape'] = 5005 # {'sape': 5005, 'guido': 4067}

More dictionary operations

- Keys, values, items
  - d.keys() # [‘sape’, ‘guido’]
  - d.values() # [5005, 4067]
  - d.items() # [(‘sape’, 5005), (‘guido’, 4067)]
- Presence check
  - ‘guido’ in d # True
- Values of any type, keys almost any
  - d = {‘name’: ’DFS Lab’, ‘year’: 2013, ‘instructors’: [‘ArB’,‘AnB’,‘SCG’,‘SB’]}

More dictionary operations

- Keys must be immutable
  - Numbers, strings, tuples of immutables
  - These cannot be changed after creation
  - Reason is *hashing* (fast lookup technique)
  - Not lists or other dictionaries
  - these types of objects can be changed "in place"
  - No restriction on values
- Keys will be listed in arbitrary order - again, because of hashing

FIN

Thank you