# An evil-genius guide to computer programming

**Daniel Lemire** 

Montreal 🙌

blog: https://lemire.me

twitter: @lemire

GitHub: https://github.com/lemire/

### Week 2

Get Smarter by Programming

#### **Numbers**

- 15 digits (64-bit, binary64)

### Non-ASCII URL

é U+00E9 C3 A9

```
import json
from urllib.request import urlopen

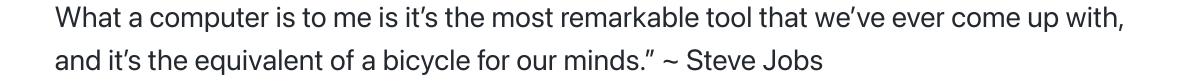
def getjson(url):
    return json.loads(urlopen(url).read().decode("utf-8"))

data = getjson("https://api.agify.io/?name=éric")
print(data)
print(data["age"])
```

```
import json
from urllib.request import urlopen

def getjson(url):
    return json.loads(urlopen(url).read().decode("utf-8"))

data = getjson("https://api.agify.io/?name=%C3%A9ric")
print(data)
print(data["age"])
```



### **Branches**

```
if 1 == 2 :
   print ("equal")
else:
   print ("not equal")
```

# Compare and jump

```
if condition :
   jump new adress
continue
```

## Loops

```
w = 0
while 2 * w + 3 < 100:
    w = 3 * w + 1</pre>

for i in range(10):
    print(w)
```

## Loops are just branches

```
beginning
something
something
if condition : go elsewhere
go to beginning
```

### **Advanced arithmetic**

```
x // y
x % y
x>>6
x<<3
```

```
>>> 10//3
3
>>> 10%3
1
>>> 10>>1
5
>>> 10<2
40
```

```
>>> (-1)%3
2
```

### Random numbers

Pseudo !!!

really hard to do well.

## D. H. Lehmer (1951)

```
seed = 1234

def random():
    global seed
    seed = seed * 0xda942042e4dd58b5
    value = seed >> 64
    value = seed % 2**64
    return value

for i in range(10):
    print(random())
```

```
for i in range(10):
    print(random()%3)
```

### **Towers of Hanoi**


```
state = [[5,2,1], [], []]
while True:
  start = random.choice([0,1,2])
  if len(state[start]) == 0:
      continue
  end = random.choice([0,1,2])
  startvalue = minimum(state[start])
  endvalue = minimum(state[end])
  if endvalue != 0 and startvalue > endvalue:
      continue
  state[start] remove(startvalue)
  state[end].append(startvalue)
  steps = steps + 1
  if len(state[2]) == 3:
      break
```

# Solve puzzling probability problems



```
import random
times = 1000000
hit = 0
for x in range(times):
   treasure = random.choice([1,2,3])
   mychoice = random.choice([1,2,3])
   if mychoice == treasure:
       hit += 1
print(hit / times)
# 0.333
```

```
import random
times = 1000000
hit = 0
for x in range(times):
   treasure = random.choice([1,2,3])
   mychoice = random.choice([1,2,3])
   if mychoice 1= treasure:
       hit += 1
print(hit / times)
# 0.667
```

Solve Peg solitaire

https://en.wikipedia.org/wiki/Peg\_solitaire

```
def find playables(peg):
    playables = []
    for r in range(len(peg)):
        for c in range(len(peg[0])):
             if peg[r][c] == 1:
                 # search neighbor
                 if (r > 1 \text{ and } peg[r-1][c] == 1 \text{ and } peg[r-2][c] == 0):
                      playables.append(((r,c),(r-2,c)))
                 if (c > 1 \text{ and } peg[r][c-1] == 1 \text{ and } peg[r][c-2] == 0):
                      playables.append(((r,c),(r,c-2)))
                 if (r+2 < len(peg) and peg[r+1][c] == 1 and peg[r+2][c] == 0):
                      playables.append(((r,c),(r+2,c)))
                 if (c+2 < len(peg[0])) and peg[r][c+1] == 1 and peg[r][c+2] == 0):
                      playables.append(((r,c),(r,c+2)))
    return playables
```

```
def tryme():
    game = []
    peg = [[2,2,1,1,1,2,2],
           [2,2,1,1,1,2,2],
           [1,1,1,1,1,1,1]
           [1,1,1,0,1,1,1],
           [1,1,1,1,1,1,1],
           [2,2,1,1,1,2,2]
    count = 0
    for r in range(len(peg)):
      for c in range(len(peg[0])):
          if peg[r][c] == 1:
              count += 1
```

```
while count > 1:
    playables = find_playables(peg)
    if len(playables) == 0 :
        return False, game
    source, dest = random.choice(playables)
    game.append((source,dest))
    peg[source[0]][source[1]] = 0
    peg[(source[0]+dest[0])//2][(source[1]+dest[1])//2] = 0
    peg[dest[0]][dest[1]] = 1
    count -= 1
return True, game
```

```
count = 0
result, g = tryme()
while(not result):
    count += 1
    if(count % 1000 ==0):print(count)
    result, g = tryme()

# [((3, 1), (3, 3)), ((5, 2), (3, 2)), ((2, 2), (4, 2)),...
```

#### Homework

Find a simple problem that you can solve with random numbers.