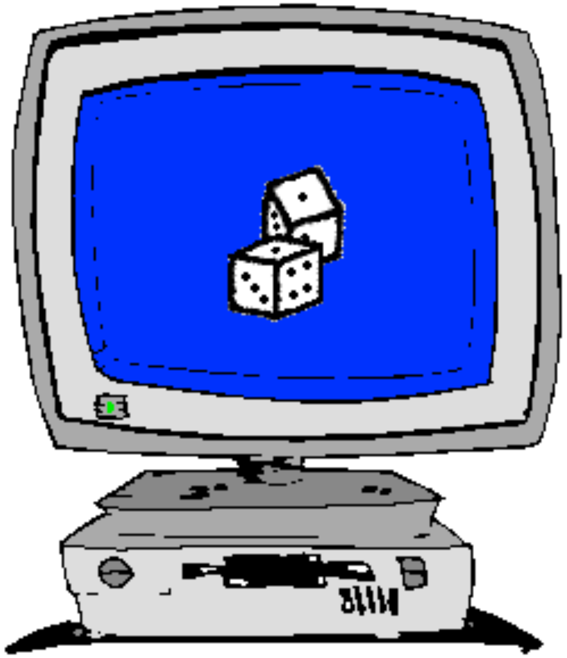
15-112
Fundamentals of Programming

Week 2 - Lecture 1:
Strings part 2 + Monte Carlo method



May 23, 2016

Plan for today

Wrap up strings

Monte Carlo simulation

String literals

x = “#FeelTheBern” → string literal

x = ‘#FeelTheBern’ single-quotes

x = ‘‘‘#FeelTheBern’’’ triple single-quotes

x = “““#FeelTheBern””” triple double-quotes

What are the differences between these?

String literals

Single-quotes and double-quotes work similarly.

print(“hello world”)	hello world
print(‘hello world’)	hello world
print(“He said: “hello world”.”)	Syntax error
print(‘He said: “hello world”.’)	He said: “hello world”.
print(“He said: ‘hello world’.”)	He said: ‘hello world’.
print(“Hello World”)	Syntax error

String literals

Use triple quotes for multi-line strings.

print(““““““hello
world””””””)

hello
world

x = ‘‘‘#FeelTheBern
Hillary’’’

print(x)

#FeelTheBern
Hillary

newline character

What value does x really store?

‘#FeelTheBern\nHillary’

String literals

\n newline **\t tab**

x = “#FeelTheBern**\n**Hillary”

print(x)

#FeelTheBern
Hillary

x = “#FeelTheBern**\t**Hillary”

print(x)

#FeelTheBern Hillary

String literals

Escape characters: use \

print(“The newline character is \n.”) The newline character is
.

print(“The newline character is \\n.”) The newline character is \n.

print(“He said: \“hello world\”.”) He said: “hello world”.

String literals

Second functionality of \ : ignore newline

print(“““#FeelTheBern Hillary”””)	#FeelTheBern Hillary
print(“““#FeelTheBern \\ Hillary”””)	#FeelTheBern Hillary
print(‘#FeelTheBern \\ Hillary’)	#FeelTheBern Hillary

The in operator

The **in** operator returns True or False.

t = “h”

s = “hello”

print(t **in** s) — same as isSubstring(t, s)

print(“h” **in** “hello”)	True
print(“ll” **in** “hello”)	True
print(“H” **in** “hello”)	False
print(“” **in** “hello”)	True
print(“k” **not in** “hello”)	True

Built-in constants

```
import string

print(string.ascii_letters)

print(string.ascii_lowercase)

print(string.ascii_uppercase)

print(string.digits)

print(string.punctuation)

print(string.printable)

print(string.whitespace)

print("\n" in string.whitespace)
```

Example

```
import string

def isLowercase(c):
    return (c in string.ascii_lowercase)
```

Built-in string methods

Method: a function applied "directly" on an object/data

Example: there is a string method called upper( ), it works like toUpper( ) from the HW.

s = "hey you!"

print(upper(s)) ERROR: not used like a function.

print(s.upper()) HEY YOU!

```
s.upper()      is kind of like
upper(s)        (if upper was a function)
```

Built-in string methods

Method: a function applied "directly" on an object/data

Example: there is a string method called count( ):

s = "hey hey you!"

print(s.count("hey")) 2

```
s.count("hey")
count(s, "hey")
```

`s.count("hey")` is kind of like `count(s, "hey")` (if count was a function)

Built-in string methods

- isupper
- islower
- isdigit
- isalnum
- isalpha
- isspace
- upper
- lower
- replace
- strip
- count
- startswith
- endswith
- find

Built-in string methods

split and splitlines

```
names = “Alice,Bob,Charlie,David”

for name in names.split(“,”):
    print(name)
```

Alice
Bob
Charlie
David

returns [“Alice”, “Bob”, “Charlie”, “David”]

Built-in string methods

split and splitlines

```
s.splitlines()  ≈  s.split("\n")
```

```
quotes = """\
Dijkstra: Simplicity is prerequisite for reliability.
Knuth: If you optimize everything, you will always be unhappy.
Dijkstra: Perfecting oneself is as much unlearning as it is learning.
Knuth: Beware of bugs in the above code; I have only proved it correct, not tried it.
Dijkstra: Computer science is no more about computers than astronomy is about telescopes.
"""
```

```
for line in quotes.splitlines():
    if (line.startswith("Knuth")):
        print(line)
```

String formatting

team = “Steelers”

numSB = 6

s = “The ” + team + “ have won ” + numSB + “ Super Bowls.”

String formatting

```
team = “Steelers”
numSB = 6
s =  “The ” + team + “ have won ” + str(numSB) + “ Super Bowls.”
```

```
team = “Steelers”
numSB = 6
s =  “The %s have won %d Super Bowls” % (team, numSB)
```

%s → string

%d → decimal

```
print(s)
```

The Steelers have won 6 Super Bowls

String formatting

print("Miley Cyrus gained %f pounds!" % 2**(-5))

float

Miley Cyrus gained 0.03125 pounds!

print("Miley Cyrus gained %.2f pounds!" % 2**(-5))

Miley Cyrus gained 0.03 pounds!

print("Miley Cyrus gained %10.2f pounds!" % 2**(-5))

Miley Cyrus gained   0.03 pounds!

print("Miley Cyrus gained %-10.2f pounds!" % 2**(-5))

Miley Cyrus gained 0.03   pounds!

String formatting

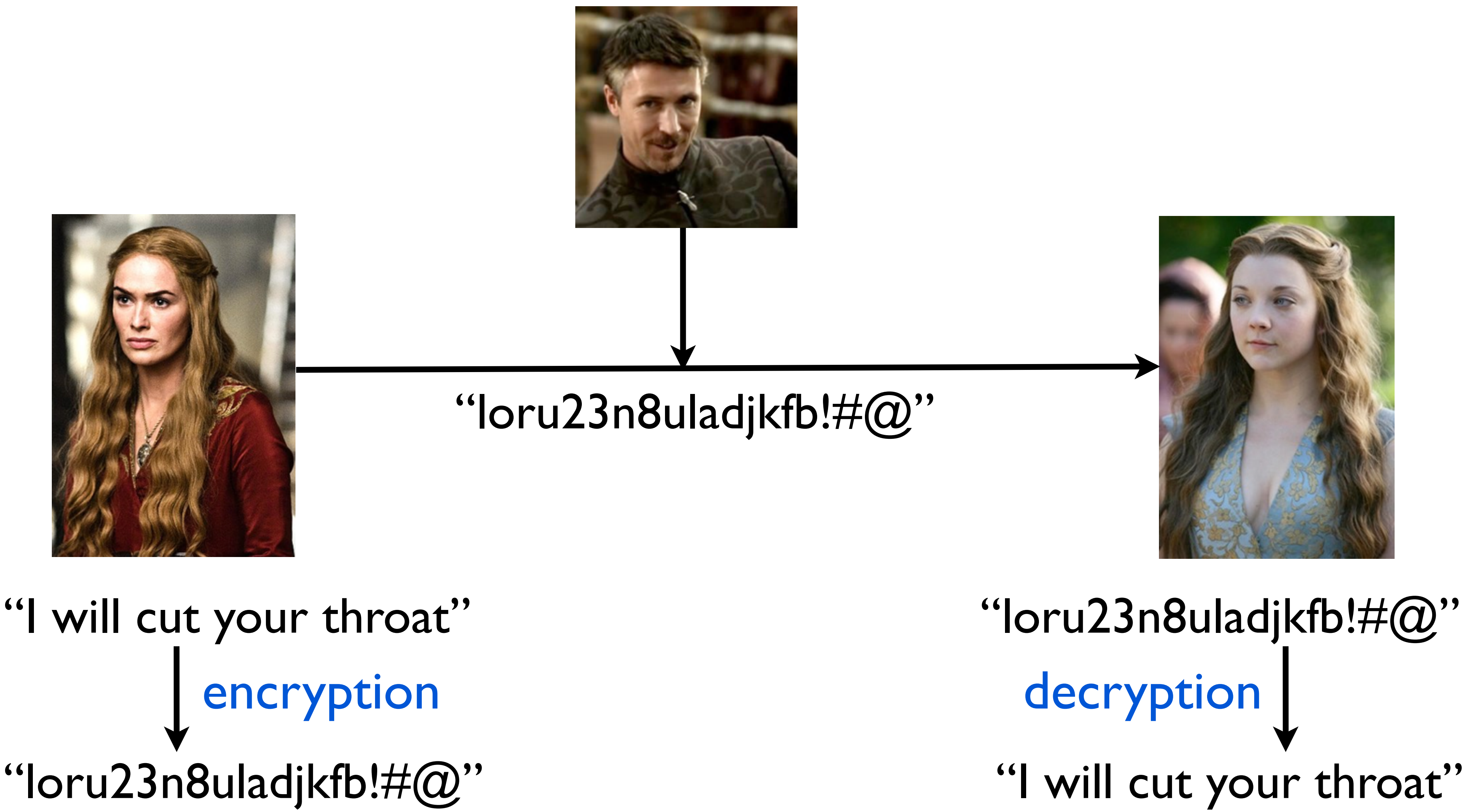
print(“Miley Cyrus gained %-10.2f pounds!” % 2**(-5))

Miley Cyrus gained 0.03   pounds!

```
% [-] [minWidth] [.precision] type
```

[-], [minWidth], [.precision] → optional

Example: Cryptography



Example: Caesar shift

Encrypt messages by shifting each letter a certain number of places.

Example: shift by 3

a ⟶ d b ⟶ e c ⟶ f ... x ⟶ a y ⟶ b ...

A ⟶ D B ⟶ E ... X ⟶ A Y ⟶ B ...

(other symbols stay the same)

15112 Rocks my world ⟶ 15112 Urfvn pb zruog

Write functions to encrypt and decrypt messages.
(message and shift given as input)

Example: Caesar shift

```
def encrypt(message, shiftNum):
    result = “”
    for char in message:
        result += shift(char, shiftNum)
    return result


def shift(c, shiftNum):
    shiftNum %= 26
    if (not c.isalpha()):
        return c
    alph = string.ascii_lower if (c.islower())  else string.ascii_upper
    shifted_alph = alph[shiftNum:] + alph[:shiftNum]
    return shifted_alph[alph.find(c)]
```

Example: Caesar shift

```
def shift2(c, shiftNum):
    shiftNum %= 26
    if('A' <= c <= 'Z'):
        if(ord(c) + shiftNum > ord('Z')):
            return chr(ord(c) + shiftNum - 26)
        else:
            return chr(ord(c) + shiftNum)
    elif('a' <= c <= 'z'):
        if(ord(c) + shiftNum > ord('z')):
            return chr(ord(c) + shiftNum - 26)
        else:
            return chr(ord(c) + shiftNum)
    else:
        return c
```

Code repetition

Exercise: Rewrite avoiding the repetition

Tangent: Private-Key Cryptography

Cryptography before WWII

Tangent: Private-Key Cryptography

Cryptography before WWII

“#dfg%y@d2hSh2$&”

“I will cut your throat”

“#dfg%y@d2hSh2$&”

“#dfg%y@d2hSh2$&”

“I will cut your throat”

Cryptography before WWII

there must be a secure way of exchanging the key

Tangent: Public-Key Cryptography

Cryptography <u>after</u> WWII





Tangent: Public-Key Cryptography

Cryptography after WWII

“#dfg%y@d2hSh2$&”

“I will cut your throat”

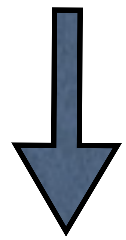
“#dfg%y@d2hSh2$&”

“#dfg%y@d2hSh2$&”

“I will cut your throat”

Tangent: The factoring problem

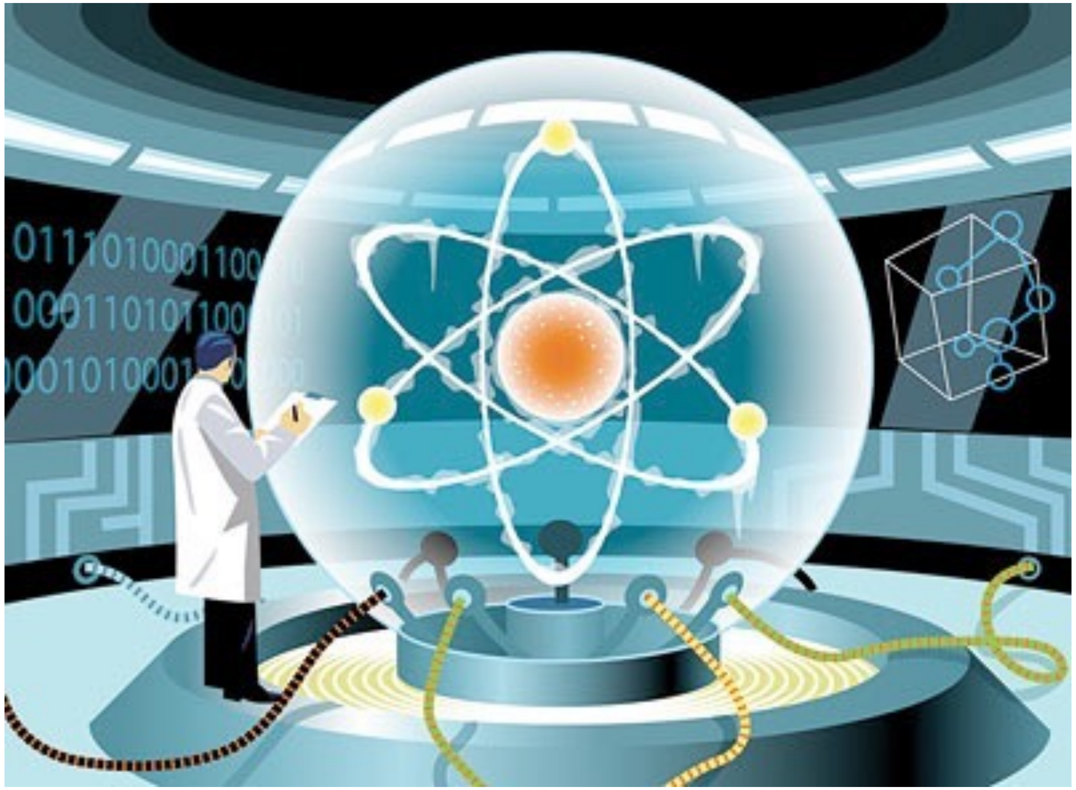
If there is an efficient program to solve the factoring problem



can break public-key crypto systems used over the internet

Fun fact: *Quantum computers* can factor large numbers efficiently!

Tangent: What is a quantum computer?



Information processing using quantum physics.

Plan for today

Wrap up strings

Monte Carlo simulation

Origins of Probability

France, 1654

Let's bet:

I will roll a dice four times.
I win if I get a 1.

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

Hmm.

No one wants to take this bet anymore.

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

New bet:

I will roll two dice, 24 times.

I win if I get double-1's.

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

Hmm.

I keep losing money!

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

"Chevalier de Méré"

Antoine Gombaud

Alice and Bob are flipping a coin.

Alice gets a point for heads.

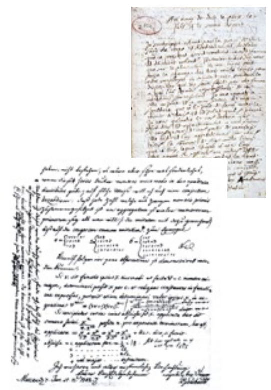
Bob gets a point for tails.

First one to 4 points wins 100 francs.

Alice is ahead 3-2 when gendarmes arrive to break up the game.

How should they divide the stakes?

Origins of Probability



Pascal

Fermat

Probability Theory is born!

Monte Carlo Method

Estimating a quantity of interest (e.g. a probability) by simulating random experiments/trials.

General approach:

Run trials

In each trial, simulate event (e.g. coin toss, dice roll, etc)

Count # successful trials

$$\text{Estimate for probability} = \frac{\#\ \text{successful trials}}{\#\ \text{trials}}$$

Law of Large Numbers:

As trials —> infinity, estimate —> true probability

Odds of Méré winning

```
def mereOdds():
    trials = 100*1000
    successes = 0
    for trial in range(trials):
        if(mereWins()):
            successes += 1
    return successes/trials


def mereWins():
    for i in range(4):
        dieValue = random.randint(1,6)
        if(dieValue == 1): return True
    return False
```

Example 2: Birthday problem

- Let n = # people in a room.
- Assume people have random birthdays (discard the year).
- What is the minimum n such that:

Pr[any 2 people share a birthday] > 0.5

(ignore Feb 29)

What is the probability if n = 366?

What is the probability if n = 1?

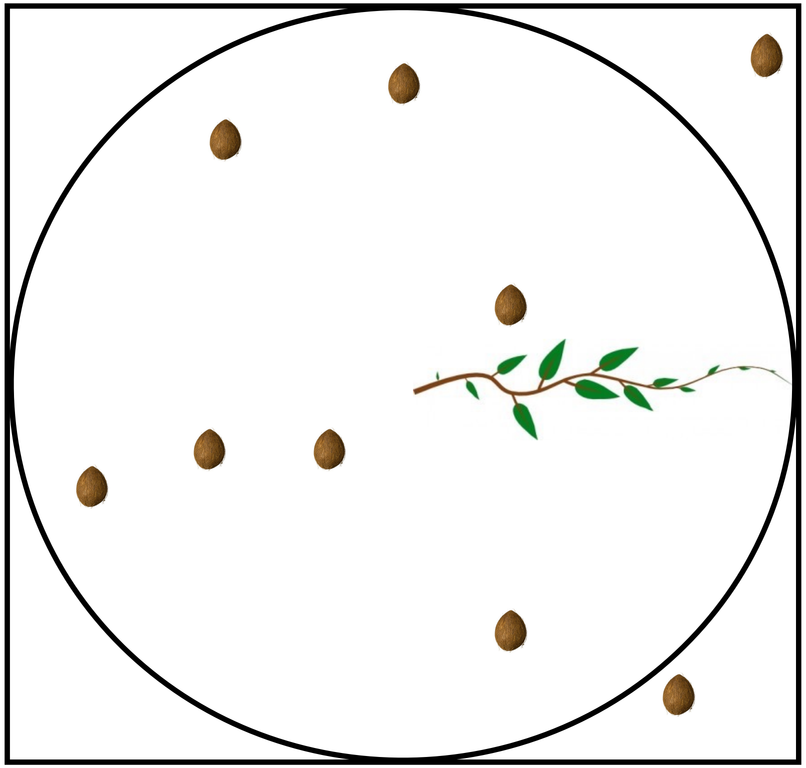
Example 2: Birthday problem

```
def birthdayOdds(n):
    trials = 10*1000
    successes = 0
    for trial in range(trials):
        if trialSucceeds(n):
            successes += 1
    return successes / trials
```

```
def trialSucceeds(n):
    seenBirthdays = “”
    for person in range(n):
        birthday = “$” + str(random.randint(1, 365)) + “$”
        if (birthday in seenBirthdays): return True
        else: seenBirthdays += birthday
    return False
```

Example 3: Estimating Pi

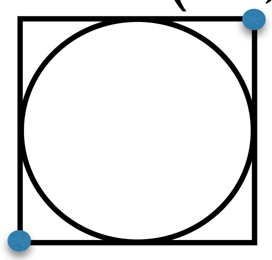
Example 3: Estimating Pi



Pr [random coconut lands in circle] =

$$\frac{\text{area of circle}}{\text{area of square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Example 3: Estimating Pi




```
def findPi(throws):                # throws = # trials
    throwsInCircle = 0             # throwsInCircle = # successes
    for throw in range(throws):
        x = random.uniform(-1, +1)
        y = random.uniform(-1, +1)
        if (inUnitCircle(x,y)):
            throwsInCircle += 1
    return 4*(throwsInCircle/throws)
```

```
def inUnitCircle(x,y):
    return (x**2 + y**2 <= 1)
```

Example 4: Monty Hall problem