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

What are the differences between these?
String literals

**Single-quotes** and **double-quotes** work similarly.

```python
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.

```python
print(""""""hello
world"")
```

```python
x = """"#FeelTheBern
Hillary"
```

```python
print(x)
```

What value does `x` really store?

```
'#:FeelTheBern
Hillary'
```
String literals

```
x = "#FeelTheBern\nHillary"
```

```
print(x)
#FeelTheBern
Hillary
```

```
x = "#FeelTheBern\tHillary"
```

```
print(x)
#FeelTheBern   Hillary
```
String literals

**Escape characters:** use \n
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

```python
print('''#FeelTheBern Hillary''')
print('''#FeelTheBern 
Hillary''')
print('``#FeelTheBern \nHillary'')
print('#FeelTheBern Hillary
Hillary')
```
The `in` operator returns True or False.

```python
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

```python
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)
```
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 `toUpperCase()` from the HW.

```python
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)
**Method**: a function applied “directly” on an object/data

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

```python
s = "hey hey you!"

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

`s.count("hey")` is kind of like `count(s, "hey")` *(if count was a function)*
Built-in string methods

isupper  replace
islower  strip
isdigit   count
isalnum   startswith
isalpha   endswith
isspace   find
upper
lower
names = “Alice,Bob,Charlie,David”

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

returns [“Alice”, “Bob”, “Charlie”, “David”]
Built-in string methods

split and splitlines

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

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

print(s)  The Steelers have won 6 Super Bowls
String formatting

\[
\text{print(“Miley Cyrus gained } %f \text{ pounds!” } \% 2**(-5))
\]

\[
\text{float Miley Cyrus gained 0.03125 pounds!}
\]

\[
\text{print(“Miley Cyrus gained } %.2f \text{ pounds!” } \% 2**(-5))
\]

\[
\text{Miley Cyrus gained 0.03 pounds!}
\]

\[
\text{print(“Miley Cyrus gained } %10.2f \text{ pounds!” } \% 2**(-5))
\]

\[
\text{Miley Cyrus gained 0.03 pounds!}
\]

\[
\text{print(“Miley Cyrus gained } %-10.2f \text{ pounds!” } \% 2**(-5))
\]

\[
\text{Miley Cyrus gained 0.03 pounds!}
\]
String formatting

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

Miley Cyrus gained 0.03   pounds!
```

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

optional
Example: Cryptography

“I will cut your throat”

“loru23n8uladjkfb!#@”

encryption

“loru23n8uladjkfb!#@”

decryption

“I will cut your throat”
Example: Caesar shift

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

Example: shift by 3

\[
\begin{align*}
a &\rightarrow d & b &\rightarrow e & c &\rightarrow f & \ldots & x &\rightarrow a & y &\rightarrow b & \ldots \\
A &\rightarrow D & B &\rightarrow E & \ldots & X &\rightarrow A & Y &\rightarrow B & \ldots
\end{align*}
\]

(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

```python
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

```python
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
```

Exercise: Rewrite avoiding the repetition

Code repetition
Tangent: Private-Key Cryptography

Cryptography before WWII

“I will cut your throat”

“#dfg%y@d2hSh2$&”

“#dfg%y@d2hSh2$&”

“I will cut your throat”

“#dfg%y@d2hSh2$&”

“#dfg%y@d2hSh2$&”
Tangent: Private-Key Cryptography

Cryptography before WWII

there must be a secure way of exchanging the key
Tangent: Public-Key Cryptography

Cryptography after WWII
Tangent: Public-Key Cryptography

Cryptography after WWII

“#dfg%y@d2hSh2$&”

“I will cut your throat”

Unlock

“#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

"Chevalier de Méré"
Antoine Gombaud

Let’s bet:
I will roll a dice four times.
I win if I get a 1.
Origin of Probability

France, 1654

“Chevalier de Méré”
Antoine Gombaud

Hmm.
No one wants to take this bet anymore.
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

“Chevalier de Méré”
Antoine Gombaud

Hmm.
I keep losing money!
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

Probability Theory is born!

Fermat
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**

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

**Law of Large Numbers:**

As **trials** \( \rightarrow \) infinity, estimate \( \rightarrow \) true probability
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:
  \[ \Pr[ \text{any 2 people share a birthday} ] > 0.5 \]
  (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
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

```python
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)
```

(-1,-1) to (+1,+1)

# throws = # trials
# throwsInCircle = # successes

(-1,-1) to (+1,+1)
Example 4: Monty Hall problem