INT 489 Selected Topics IN IT

Dr. Khaled Mostafa Reda

Sinai University (SU)

Faculty of Information Technology and Computer Science (FIT)

E-Mail: khaled.Mostafa@su.edu.eg

Lecture #3

Last Time

- String object type
- Branching and conditionals
  - if/elif/else
- Indentation
- Iteration and loops
  - while loops
  - for loops

TODAY

- String manipulation.
- Guess and check algorithms.
- Bisection Search

Strings

- Think of as a **sequence** of case-sensitive characters
- can compare strings with ==, >, <, etc.
- `len()` is a function used to retrieve the **length** of the string in the parentheses

```
s = "abc"
len(s) →evaluates to 3
```

Strings, Cont.,

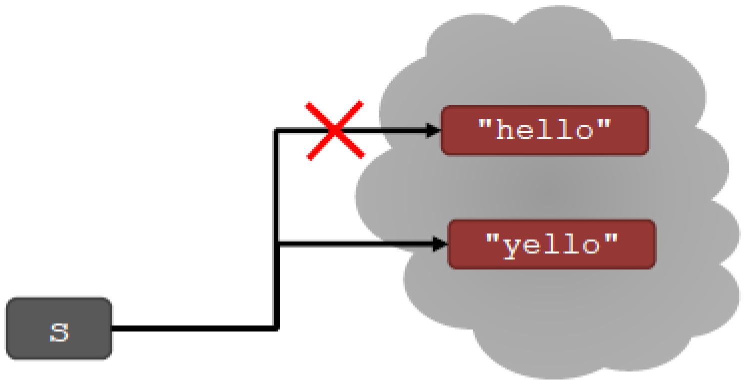
- Square brackets used to perform **indexing** into a string to get the value at a certain index/position
- s = "abc"
  - index: 0 1 2 → indexing always starts at 0
  - index: -3 -2 -1 → last element always at index -1
- s[0] →evaluates to "a"
- s[1] → evaluates to "b"
- s[2] → evaluates to "c"
- s[3] → trying to index out of bounds, error
- s[-1] → evaluates to "c"
- s[-2] → evaluates to "b"
- s[-3] → evaluates to "a"

Strings, Cont.,

- Can **slice** strings using [start:stop:step]
- If give two numbers, [start:stop], step=1 by default
- You can also omit numbers and leave just colons
- s = "abcdefgh“
- s[3:6] → evaluates to "def", same as s[3:6:1]
- s[3:6:2] → evaluates to "df"
- s[::] → evaluates to "abcdefgh", same as s[0:len(s):1]
- s[::-1] → evaluates to "hgfedbca", same as s[-1:-(len(s)+1):-1]
- s[4:1:-2] → evaluates to "ec"

Strings, Cont.,

- strings are "**immutable**" – cannot be modified
- `s = "hello"`
- `s[0] = 'y'` → gives an error
- `s = 'y'+s[1:len(s)]` → is allowed, s bound to new object



for Loops

- for loops have a **loop variable** that iterates over a set of values
- for var in range(4): → var iterates over values 0,1,2,3

    <expressions> → expressions inside loop executed with each value
- for var for var in range(4,6): → var iterates over values 4,5

    <expressions>
- range is a way to iterate over numbers, but a for loop variable can **iterate over any set of values**, not just numbers!

Strings and Loops

- These two code snippets do the same thing
- Bottom one is more "more efficient"

```
for index in range(len(s)):
      if s[index] == 'i' or s[index] == 'u’:
          print("There is an i or u")
for char in s:
      if char == 'i' or char == 'u’:
           print("There is an i or u")
```

Code Example: ROBOT

- `an_letters = "aeiou"`
- `word = input("I will glad for you! Enter a word: ")`
- `times = int(input("Enthusiasm level (1-10): "))`

```
i = 0
while i < len(word):
    char = word[i]     ✗
```

```
for char in word:     ✓
```

```
    if char in an_letters:
        print("Give me an " + char + "! " + char)
    else:
        print("Give me a " + char + "! " + char)
    i += 1     ✗
```

Code Example: ROBOT Cont.,

```
print("What does that spell?")
for i in range(times):
    print(word, "!!!")
```

Guess and Check

- The process below also called **exhaustive enumeration**
- Given a problem…
- You are able to **guess a value** for the solution
- you are able to **check if the solution is correct**
- keep guessing until finding a solution or guessed all values

Guess and Check Cube root

```
cube = 8
for guess in range(cube+1):
    if guess**3 == cube:
        print("Cube root of", cube, "is", guess)
```

Guess and Check Cube root

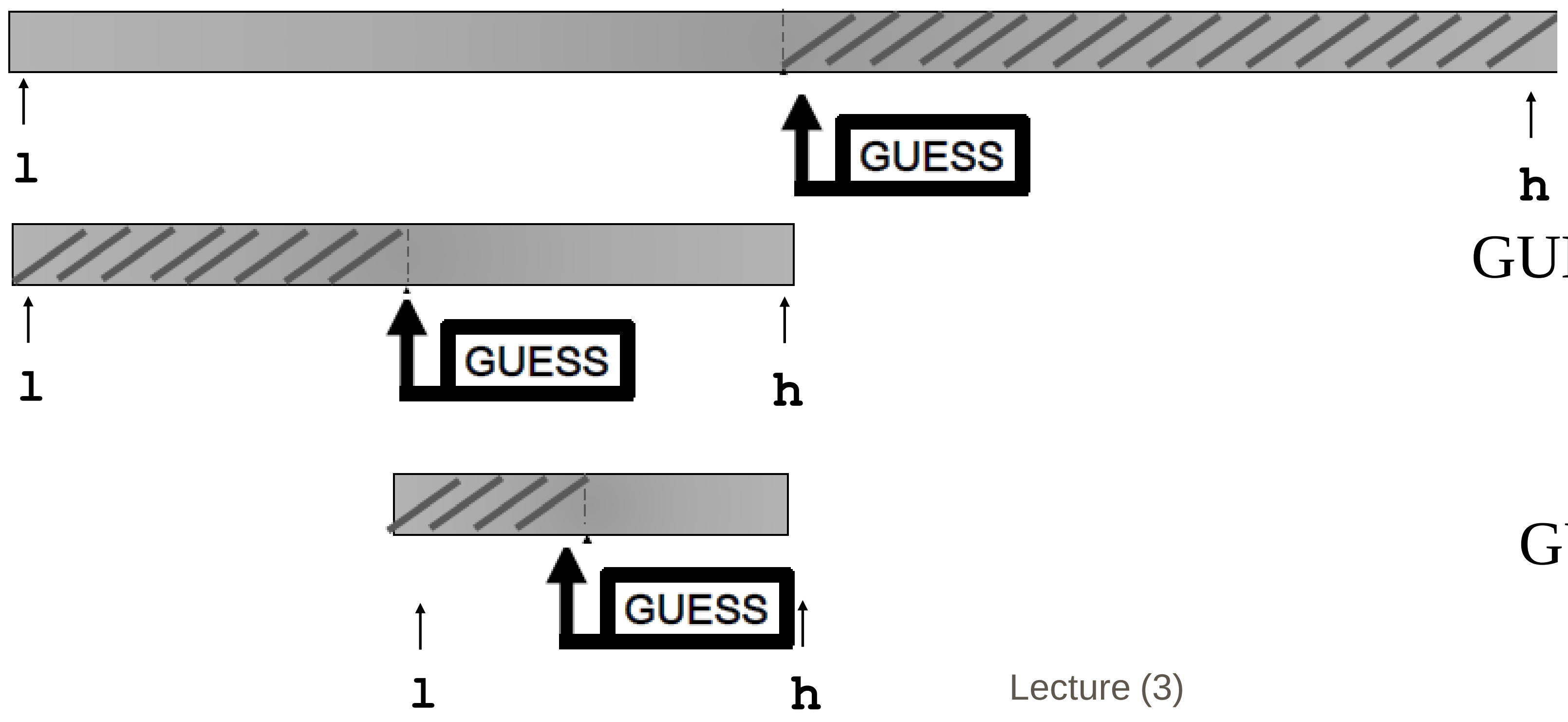
```
cube=int(input(" Enter the number need to be check
as prefect cube = ") )

for guess in range(abs(cube)+1):
    # passed all potential cube roots
    if guess**3 >= abs(cube):
        # no need to keep searching
        break
```

Guess and Check Cube root

```
if guess**3 != abs(cube):
    print(cube, 'is not a perfect cube')
else:
    if cube < 0:
        guess = -guess
    print('Cube root of ' + str(cube) + ' is ' + 
str(guess))
```

Bisection Search

- Half interval each iteration
- New guess is halfway in between
- To illustrate, let’s play a game!



GUESS = ((h –l) /2) + l

response = high

h = GUESS

GUESS = ((h – l)/2) + l

response = low

l = GUESS

GUESS = (h – l)/2 + l

response = c

Bisection Search

```
ans = False
low = int(input("Please enter starting point: "))
high = int(input("Please enter ending point: "))
input (f"Think of a number from {low} to {high}.
Press enter to continue")
high+=1
```

Bisection Search

```
while not ans:
    guess = (high - low) // 2 + low
    print (f"Is your number {guess}?")
    reps = input(""" Enter h to indicate the guess
is too high
 Enter l to indicate the guess is too low
 Enter c to indicate the guess is the correct
 Enter answer:  """).lower()
```

Bisection Search

```
if reps == 'h':
        high = guess
elif reps == 'l':
        low = guess
elif reps == 'c’:
        ans = True
        print ("thank for Playing" )
```

Bisection Search

- Search space
  - first guess: N/2
  - second guess: N/4
  - kth guess: $N/2^k$
- Guess converges on the order of $\log_2 N$ steps
- Bisection search works when value of function varies monotonically with input

Assignment

1. Analyze and design an algorithm by drawing flowcharts and writing pseudo-code to accept a word and compare its letters with vowel letters using for loop and print “an” before each vowel letter and “a” before each constant letter and, accept definite times to repeat print the word. Finally, write a Python program to express your design.

2. Analyze and design an algorithm by drawing flowcharts and writing pseudo-code to find the guessing element within a sorted list of numbers using Bisection Search or Logarithmic Search. Finally, write a Python program to express your design.

Thank You