Intro to Computer Science
CS-UH 1001, Spring 2022

Lecture 5 – List Methods, Decision Structures

Today's Lecture

- List Methods
- Decision structure

Recap String Methods

- Replacing substrings
  - .replace(old, new, max)
- Finding substrings
  - .find(substring, start, end)
- Counting substrings
  - .count(substring, start, end)

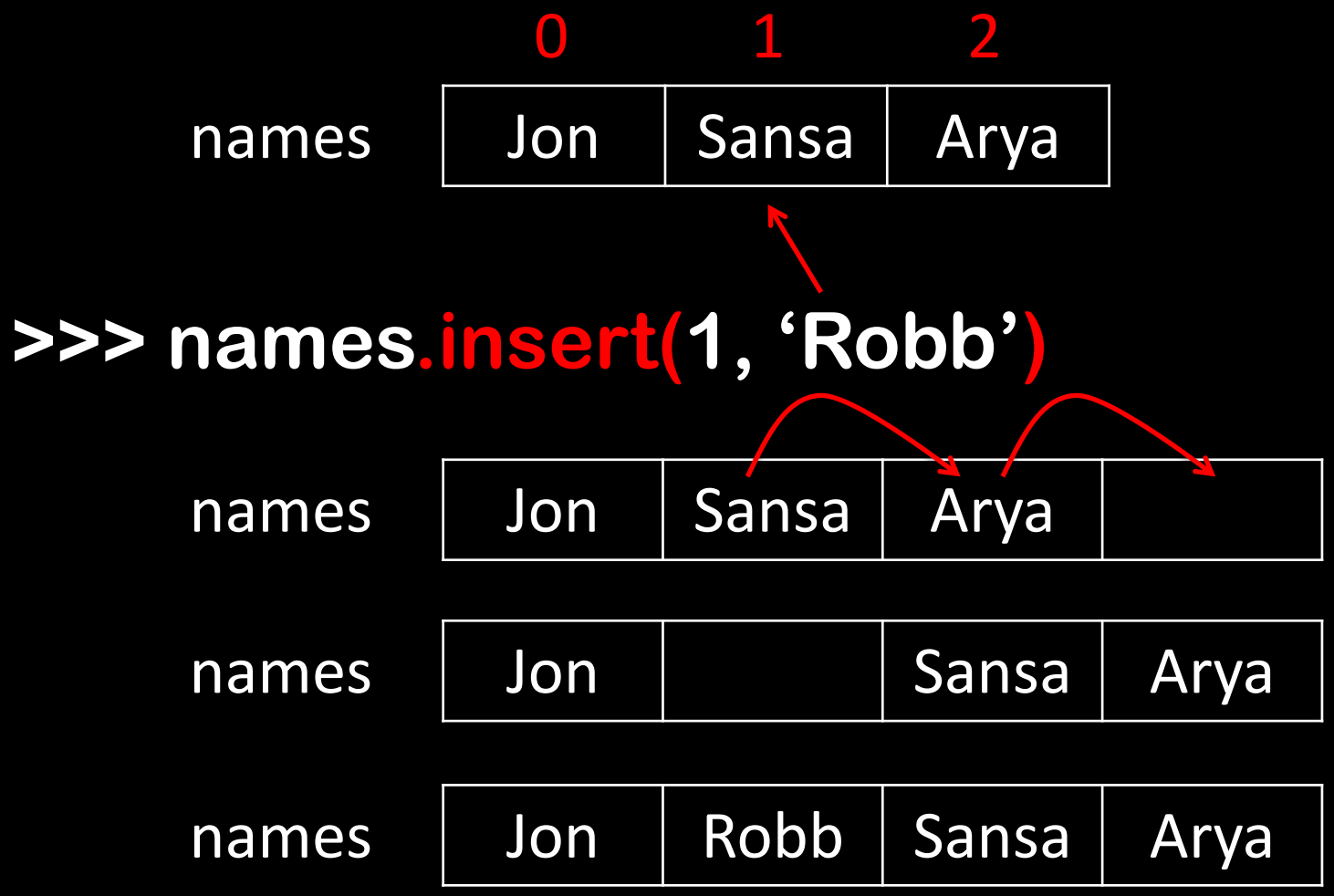
Recap Lists

- Lists are created by []
  - Empty list: my_list = []
  - List with initial values: my_list = [1, 2, 3, 4]
- Lists are mutable
- List items are indexed: my_list[1] -> 1
- List slicing always returns a list
- Adding items to a list:
  - .append(item)

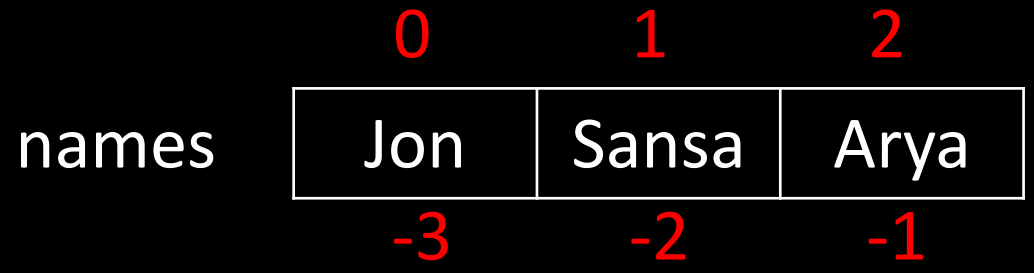
Inserting Items Into Lists

- The insert(index, item) method
  - inserts an item to the list at a specific index
  - does NOT return the new list

Example: Inserting Items



Example: Inserting Items



- **What will happen if you use an invalid index?**

  >>> names.insert(50, 'Robb')

- **What will happen if you use a negative index?**

  >>> names.insert(-3, 'Robb')

Removing an Item From a List

- If you want to remove an item from a list, there are three different ways to do this:
  - list.remove(item) removes the first occurrence of the item within the list (item is NOT returned!)
  - del list[index] removes the item at the specific index from the list
  - list.pop() removes the last item from the list and returns it
    - list.pop(index) removes an item at the specific index from the list and returns it

Example: remove() Method


```
>>> names.remove('Sansa')
```

Never do:

```
>>> names = names.remove("Sansa")
>>> print(names)
None
```

Example: del Statement


```
>>> del names[2]
```


```
>>> del names[5]
```

IndexError

Example: pop() Method


```
>>> name = names.pop()
>>> print(name)
Robb
```

Example: pop() Method with Index


```
>>> print(names.pop(2))
Arya
```

Other Useful List Methods and Functions

Method	Description
.sort()	Sort the items within the list in ascending order (from lower value to upper value)
.reverse()	Reverse the order of the items in the list
.count(item)	Counts how may times an item appears in the list
min (myList)	returns the element with the minimum value in the list
max (myList)	returns the element with the maximum value in the list

You can find more list methods here: https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Convert Strings to Lists

- Remember type casting?
  int(), float() and str()?

- A string can be changed into a list using the list() type cast:

```
>>> name = 'python'
>>> my_list = list(name)
>>> print(my_list)
['p', 'y', 't', 'h', 'o', 'n']
```

Converting Lists to Strings

- How about a string of words?

```
>>> sentence = 'I love python'
>>> print(list(sentence))
['I', ' ', 'l', 'o', 'v', 'e', ' ', 'p', 'y', 't', 'h', 'o', 'n']
```

Splitting Strings

- The split(separator) method splits a string

string.split(separator)

- The separator argument is optional; by default " "
- It returns a list of strings

```
>>> sentence = 'I love python'
>>> print(sentence.split())
    ['I', 'love', 'python']
```

Splitting Strings

- You can also define the separator

```
>>> sentence = 'I-love-python'
>>> sentence.split('-')   ['I', 'love', 'python']
>>> sentence.split('o')   ['I-l', 've-pyth', 'n']
>>> sentence.split('love')   ['I-', '-python']
>>> sentence.split()         ['I-love-python']
```

Lists to Strings

- The join(list) method does the opposite of the split() method
- It joins all items in the list into one string

string.join(list)

- It returns a string by joining all list elements, separated by the string

- Note: list must contain string items!

Example: Lists to Strings

```
>>> separator = " "
>>> word_list = ['I', 'love', 'python']
>>> joined_string = separator.join(word_list)
>>> print(joined_string)
I love python

>>> separator = "_"
>>> joined_string = separator.join(word_list)
>>> print(joined_string)
I_love_python
```

Breakout session I:
Lists

Guessing game (ex_5.1.py)

Write a program that

1. Asks the user to input 4 words separated by comma (all with the same theme, e.g. fruits)
2. Create a list out of all 4 words
3. Then ask the user for one more word. Add the last word to the beginning of the list
4. Ask the user to guess one of the words in the list
   - If the guess is correct, display: True
   - If guess is wrong, display: False

You can play the guessing game with a friend:

- Ask your friend to enter the 5 words without you looking
- Then try yourself to guess one

Hints:

- Use `print('\n' * 100)` to clear the screen before the guessing starts
- Use the `in` operator to check for a match

Extra Task:

- Instead of displaying True or False, display "Congratulations" or "Sorry, wrong guess"
  - Remember: A True has the value of 1, whereas a False has the value of 0. Use them as indices to print the two strings above

Decision Structures

Decision Structure

- In programing it's often required to make decisions
  - The program branches out based on a condition
  - The program executes different code depending on a condition

- Conditions are boolean expressions that always evaluate to either True or False!
  - in operator
  - Comparison operators (==, !=, <=, etc)
  - Logical operators (and, or, not)

Comparison Operators

- Comparison operators are used to compare one value to another (immutable data types only!)
- The result is either True or False

Operator	Description	Example
==	Equality	2 == 4
!=	Inequality	2 != 4
<	Less than	2 < 4
>	Greater than	2 > 4
<=	Less than or equal	2 <= 4
>=	Greater than or equal	2 >= 4

The if statement

- An if statement is used to check a condition
  - if the condition is satisfied (evaluates to True), it executes a certain code block
  - if the condition is not satisfied (evaluates to False), the code block is skipped

- Syntax:

```
if condition_expression:
    # indented code block if condition is True
```

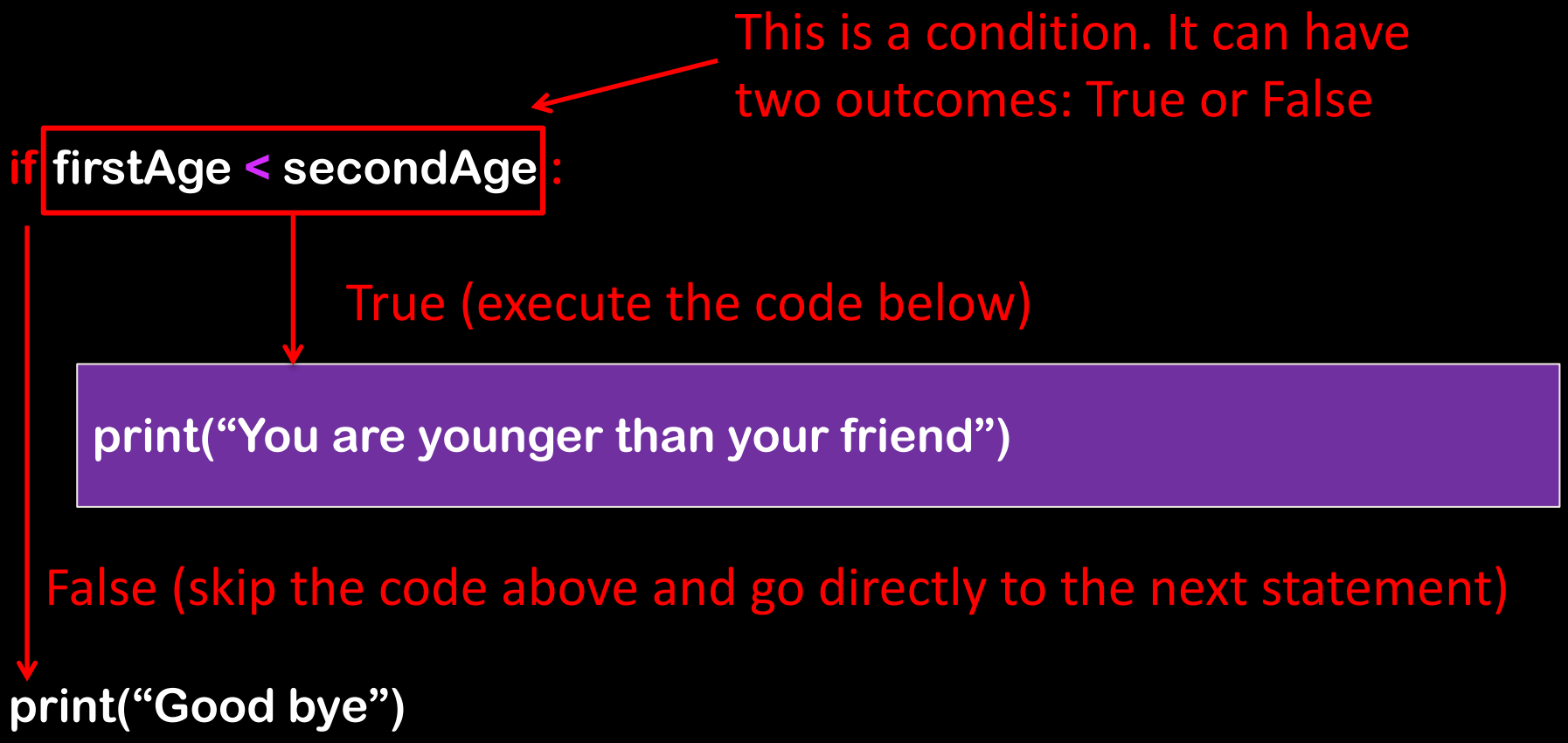
Simple Example

```
word = input(“Enter the word ‘python’: ”)
if word == 'python':
    print('Correct’)

print(‘Good bye')
```

Another Example

```
first_age = int(input("Please enter your age: "))
second_age = int(input("Please enter your friends age: "))

if first_age < second_age:
    print("You are younger than your friend.")

print("Good bye")
```

Let's take a closer look



Let's take a closer look

```
if first_age < second_age :
    print("You are younger than your friend")
print("Good bye")
```

Notice the colon at the end of the if statement, this causes the indentation.

This is called indentation, it tells the Python interpreter that the statements here belongs to a block (group) which is part of the if statement (if evaluated to True)

Let’s take a break!

Being a Programmer

Mom said: "Please go to the shop and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottle of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS"

The if statement

- You noticed in the previous example that it only prints if the first age is smaller than the second
- "Good bye" appears in both cases (True or False) since its not part of the if statement
- Now, how to print the False case?
  - if first age is larger than the second age

The if/else Statement

- An if statement can also have an else code block (optional)
  - if the condition was not satisfied (False), the else block is executed

- Syntax:

```
if condition_expression:
    # indented code block if condition is True
else:
    # indented code block if condition is False
```

Simple Example

```
word = input(“Enter the word ‘python’: ”)
if word == ‘python’:
    print(‘Correct’)
else:
    print(‘Wrong’)

print(‘Good bye')
```

Remember the Guessing Game?

Without if condition:

```
guess = input("Please guess a word: ")
message = ["Sorry, wrong guess", "Congratulations"]
print(message[guess in word_list])
```

With if condition:

```
guess = input("Please guess a word: ")
if guess in world_list:
    print(“Congratulations”)
else:
    print(“Sorry, wrong guess”)
```

The if/else statement

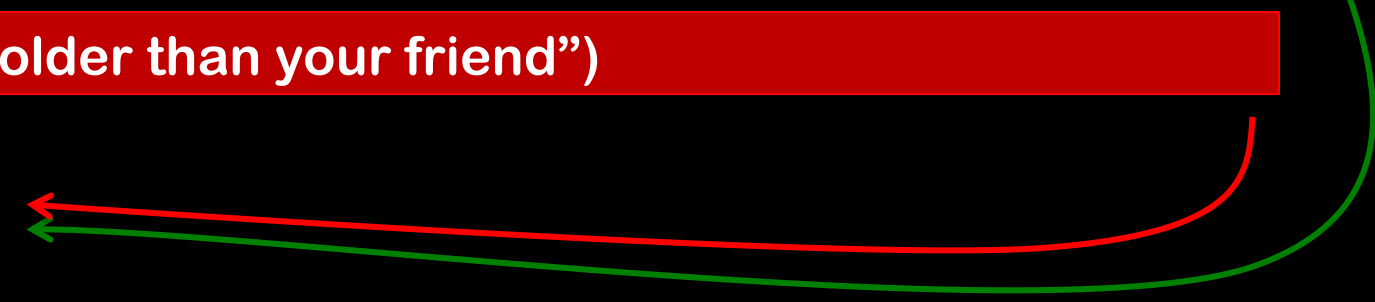
```
first_age = int(input("Please enter your age: "))
second_age = int(input("Please enter your friends age: "))

if first_age < second_age :

    print("You are younger than your friend.")

else:

    print("You are older than your friend.")

print("Good bye")
```

The if/else statement

```
first_age = int(input("Please enter your age: "))
second_age = int(input("Please enter your friends age: "))

if first_age < second_age :
    print("You are younger than your friend")
else:
    print("You are older than your friend")
print("Good bye")
```

True

False



The if/else statement

- What will happen if both have the same age?
- We will still get “you are older”
- How to fix this?
  - elif (using else if)

The if/elif/else Statement

- if statements are used to check a condition
  - if the condition is satisfied (True), it executes a certain code block
- elif statements are used to check for further conditions (optional)
  - if the elif condition is satisfied (True), it executes a certain code block
- else is executed if none of the above are True (optional)

- Syntax:

```
if condition_expression:
    # indented code block if condition is True
elif condition_expression:
    # indented code block if the if condition is False but elif condition is True
else:
    # indented code block if all above conditions are False
```

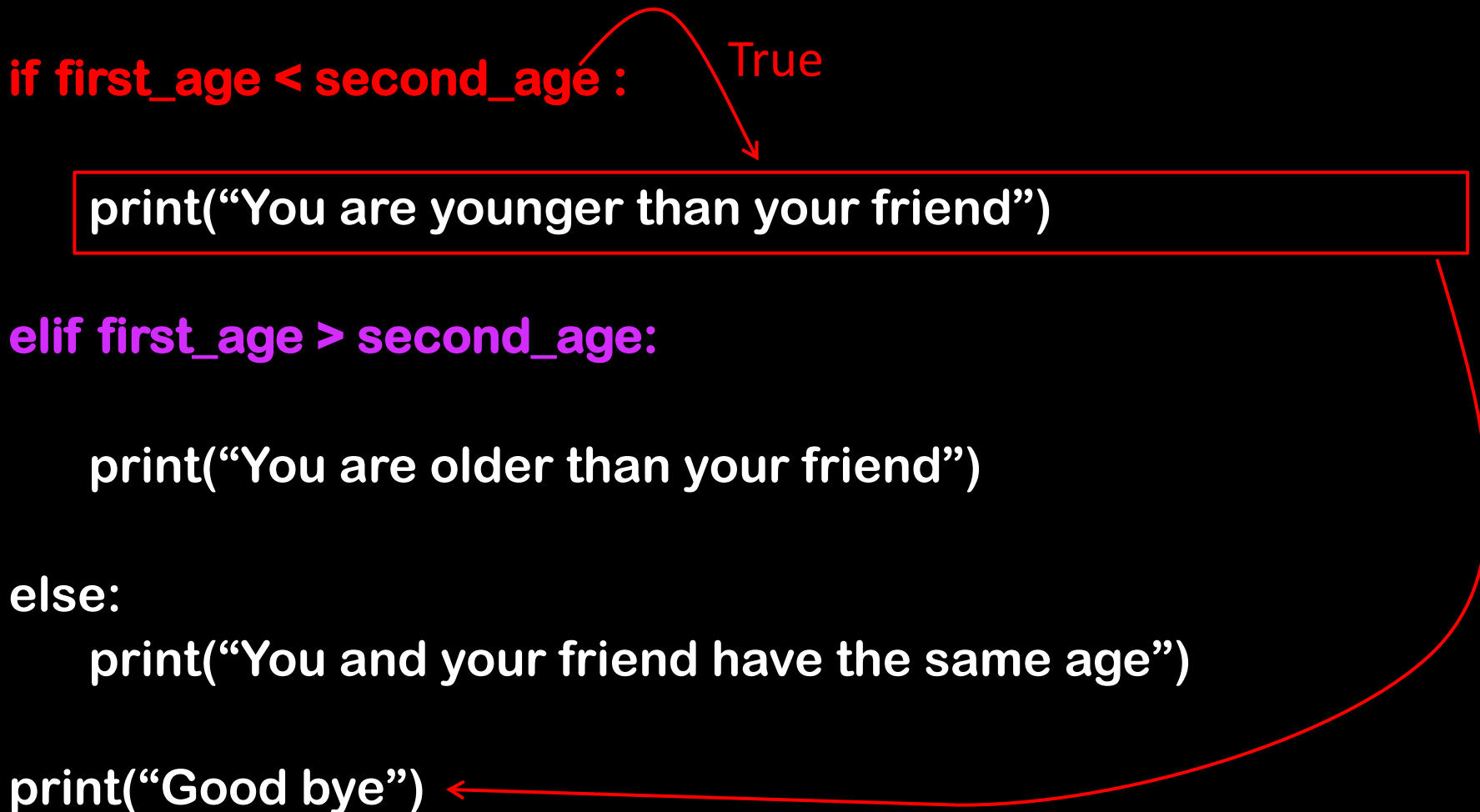
Simple Example

```
word = input("Enter the word 'python': ")
if word == 'python':
    print('Correct')
elif word == '':
    print('Nothing entered')
else:
    print('Wrong')

print('Good bye')
```

If, elif and else

```
if first_age < second_age:

    print("You are younger than your friend")

elif first_age > second_age:

    print("You are older than your friend")

else:

    print("You and your friend have the same age")

print("Good bye")
```

Case 1: first person is younger



Case 2: first person is older

```
if first_age < second_age:
    print("You are younger than your friend")
elif first_age > second_age:
    print("You are older than your friend")
else:
    print("You and your friend have the same age")

print("Good bye")
```

False

True

Case 3: both have the same age

```
if first_age < second_age:
    print("You are younger than your friend")
elif first_age > second_age:
    print("You are older than your friend")
else:
    print("You and your friend have the same age")
print("Good bye")
```

False

False

Breakout session II:
Selection Statement

Pass or fail (ex_5.2.py)

Write a program that requests the user to input a grade (in numerical form, e.g. 90). Then the program will display whether the class is PASSED or FAILED.

Use if and else statements to do this.

The table below can help you do the mapping.

Score	Result
>= 60	Passed
< 60	Failed

Test Score (ex_5.2.py)

Write a program that requests the user to input a grade (in numerical form, e.g. 90). Then your program will display the corresponding letter grade.

Use if, elif and else. The table below can help you do the mapping.

Test Score	Grade
90 and above	A
80 - 89	B
70 - 79	C
60 - 69	D
Below 60	F

Nested Decision Structure

- Decision structure can be nested inside each other:

```
if grade > 60:
    print ("You have passed")
    if grade < 70:
        print ("Your grade is D")
    elif grade < 80:
        print ("Your grade is C")
    elif grade < 90:
        print ("Your grade is B")
    else:
        print("Your grade is A")
else:
    print("You have failed the class")
```

Test Score	Grade
90 and above	A
80 - 89	B
70 - 79	C
60 - 69	D
Below 60	F

Do you remember this example?

```
>>> names = [‘Jon’, ‘Sansa’, ‘Arya’]
>>> print(names.index(‘Cersei’))
>>> ValueError: ‘Cersei’ is not in list
```

Write a program that prevents this error (ex_5.3.py)

Next Class

- Logical operators
- Count-controlled loops
- Two dimensional Lists
- Nested loops