Last Few Lectures

• Strings & Lists
  • Indexing
  • Slicing
  • **Strings** -> immutable and **Lists** -> mutable
  • Functions
  • Methods
Learning outcomes

At the end of this course, a student is expected to:

• Describe and apply fundamental concepts and terminology of Python:
  • Conditions and execution flow
• Create (design) small to medium size programs using Python:
  • Use the core features of Python to design programs to solve problems: conditionals
Today’s Menu

• Introduce Python Statements #5

-> Conditional Statements

1. Syntax #1
2. Syntax #2
3. Syntax #3
Python statements

- Categories:
  1. Assignment statement
  2. Input statement
     - Conversion function
  3. Output statement
  4. Operational statements
     - Mathematical/arithmetic operators
     - String manipulation operators
  5. Conditional statement
  6. Iterative statement  **We’ll see this statement soon!**

Some of them are built-in function or method
Introducing Conditional Statements

• So far, the execution of our Python programs has flowed from top to bottom
  • Starting at the first statement and executing each statement all the way down *sequentially*

• Conditional statements allow us to alter the flow of the execution of our Python programs
Conditional statements – In real world!

• Making decisions
  • Every day, we make decisions based on conditions
  • Example:
    • If I am hungry, I eat.
    • If it rains this afternoon, I will take the bus, otherwise, I will walk.
    • On Tuesdays and Thursdays, I can sleep in, but on Mondays, Wednesdays and Fridays, I can’t, I need to wake up early and go to my lecture.
Conditional statements – In computer world!

- **Definition:**
  - A statement that allows our Python program to decide whether to do something or to do something else (but not both), depending on a *condition* hence affecting the sequential flow of execution.
Conditional statements – Syntax #1

Syntax #1:

```python
if <condition>
  <some statement(s)>
```

This small empty space made of a few white space (or blank) characters is called an indentation and its presence is very important in a conditional statement in Python.
How syntax #1 works?

1. The **condition** is evaluated -> **True** or **False**

2. If the condition is **True**, the **indented** statement(s) below the `if <condition> :` is/are executed

3. Then Python interpreter moves to the next statement, i.e., the statement after the `if` statement (statement that is indented at the same level as the `if` statement) and resumes its execution
Let’s practice:

- **Step 1 - Problem Statement**
  - Write a Python program that allows a user to login
    - User enters a password
    - The program displays “Log in successfully” or terminates if not

- **Solution: Login_Syntax_1.py**

- How would we test this program?
Let’s practice:

- **Step 2 – Design Algorithm and Data**

- **Step 3 – Solution Selection** (we’ll do this step later on this semester)

- **Step 4 – Implementation**
  - Solution: Login_Syntax_1.py

- **Step 5 – Testing**
  - How would we test this program?
## Test Cases

<table>
<thead>
<tr>
<th>Test Case #</th>
<th>Test Data</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Improving our solution

• Right now, when the user enters the wrong password, our program simply terminates.

• What can we do to improve the usefulness (i.e., the user interaction) of our program?

• How can we make our solution clearer to our user (i.e., more user-friendly)?
Conditional statements – Syntax #2

**if else statement**

Syntax #2:

```plaintext
if <condition>:
    <some statement(s)>
else:
    <some other statement(s)>
```

Indentation

Boolean expression
Syntax #2

• Can you guess how this second conditional statement works?
How syntax #2 works?

1. The condition is evaluated -> True or False
2. If the condition is True, the indented statement(s) directly below the if <condition> : is/are executed
3. If condition is False, the indented statement(s) directly below the else : is/are executed
4. Once step 2. or 3. above has executed, Python interpreter moves to the next statement, i.e., the statement after the if else conditional statement i.e., the next statement that is indented at the same level as the if else conditional statement, and resumes its execution
OK! Back to the password program!

- Let’s improve our program: Login_Syntax_2.py
Back to Scratch

• Does Scratch have conditional statements?
• If so, what do they look like?
Another example

-> Guessing game:

• **Step 1 - Problem Statement**
  • Write a guessing game in Python in which the computer picks a number (between 1 and 10, inclusively) and the player (i.e., user) tries to guess this number.
  • Once the player has entered a guess, our program displays an appropriate message:
    "Wow! You got it!"
    OR
    "Sorry! You missed! The number was <#>."
Guessing game:

- **Step 2 – Design Algorithm and Data**

- **Step 3 – Solution Selection** (we’ll do this step later on this semester)

- **Step 4 – Implementation**
  - Solution: GuessingGame_1.py

- **Step 5 – Testing**
  - How would we test this program?
# Test Cases

<table>
<thead>
<tr>
<th>Test Case #</th>
<th>Test Data</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Improving our solution

• Let’s improve our guessing game by letting the user know when s/he has entered a guess that is outside the range of numbers s/he could pick from

• Solution: GuessingGame_2.py

• Step 5 – Testing
  • How would we test this program?
Conditional Statements – Syntax #3

nested if statement

```python
if <condition 1>:
    <some statement(s)>
else:
```

Indentation
Can become:

**nested if statement**

```
if <condition 1>
    <some statement(s)>)
else:
    <some statement(s>)
```

This is just like our `if else` statement (see Syntax #2)
OR this:

```python
if <condition 1>:
    <some statement(s)>
else:
    if <condition 2>:
        <some statement(s)>
    else:
        <some statement(s)>
```
OR that:

```
if <condition 1>
   <Some statement(s)>
else:
   if <condition 2>
      <Some statement(s)>
   else:
      if <condition 3>
         <Some statement(s)>
   else:
   ...
```
OR even that:

if <condition 1> :
    if <condition 2> :
        <some statements A>
    if <condition 3> :
        <some statements B>
    else :
        <some statements C>
    if <condition 4> : <some statements D>
else :
    <some statements E>
else :
    <some statements F>
<some statements G>
Bottom Line

• The way we nest our conditional statements in our program is dictated by the design of our solution to the problem we are solving, i.e., our algorithm
Summary

- Introduce Python Statements #5
  -> Conditional Statements
    1. Syntax #1
    2. Syntax #2
    3. Syntax #3
Next Lecture

• Conditional Statements -> Syntax #4
• Introduction to yet another data type -> bool
• How to construct conditions