ComIT

Program Execution Flow

Program Execution Flow

- Using the syntax we have seen up until now, we can write simple programs that accept user input, compute simple expressions using various data types, and print output to the screen.
- Without the addition of any other types of statements, Python cannot be considered to be “Turing Complete”.
- Being “Turing Complete” means that a system of data-manipulation rules is capable of simulating any Turing Machine.

Turing Machine

- A Turing Machine is “a mathematical model of computation describing an abstract machine that manipulates symbols on a strip of tape according to a table of rules. Despite the model's simplicity, **it is capable of implementing any computer algorithm.**” - Wikipedia
  - Invented by Alan Turing, who used math to beat Nazis (oversimplification).
- There are many theories and mathematical proofs out there that define what minimal sets of rules are required for something to be complete.
- Python is indeed a Turing Complete language. What statements or rules are we missing so far?

Structured Program Theorem

- The structured program theorem, also called the Böhm–Jacopini theorem, states that a class of control-flow graphs (historically called flowcharts in this context) can compute any computable function if it combines subprograms in only three specific ways (control structures):
  - Executing one subprogram, and then another subprogram (sequence)
  - Executing one of two subprograms according to the value of a boolean expression (selection)
  - Repeatedly executing a subprogram as long as a boolean expression is true (iteration)

Selection Statements

- Python allows us to manipulate program execution flow based on Boolean values.
- We know these as “if statements”, and in Python they look like this:

```
if <Boolean Expression>:
        <Python Statements>
elif:
        <Python Statements>
else:
        <Python Statements>
```

Selection Statements

- The statements in the “elif” and “else” blocks are only executed if the preceding blocks are not executed.
- “elif” and “else” blocks are not required.
- You can chain as many “elif” blocks as needed.

Iteration

- Iteration is a programming concept where a set of commands can be completed repeatedly until a certain condition is reached.
  - Each execution of the set of commands is referred to an “iteration”.
- There are two ways to perform iterations in Python:
  - “For Loops” – Iterates over each element in a sequence
  - “While Loops” – Iterates until a conditional statement becomes false

For Loops

- The Syntax looks like this:

for <var> **in** <sequence>**:**
    <Python Statements>

  - Where <var> is the name of a variable, new or previously defined, and <sequence> is the identifier of any “sequence” type object.
    - Lists, Dictionaries, etc.
  - In each iteration of the loop, the <var> is populated with the next element in the sequence.

While Loops

- The Syntax looks like this:

```
while <bool>:
        <Python Statements>
```

  - Where the python statements continue until the boolean expression is evaluated as false.
  - If the conditional statement is never true, none of the code will execute.

Additional Control Statements

- There are a few other Python reserved words that help us control loops and other control flow statements:
  - break – Exits the current block. Program execution resumes after the iteration or selection statement.
  - continue – Exists the current iteration. Program execution resumes in next iteration.

Questions?