CMPT 120

Topic: Python’s building blocks
  -> Statements
Last Lecture

• Start learning Python’s building blocks
  • Variables
  • Values
  • Types
  • and Literal values
• Hand tracing
• Execution Flow
• Comments and Comment Header Block
Learning outcomes

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

• Describe and apply (i.e., use in s/w dev.) fundamental concepts and terminology of Python:
  • Expressions
  • Operators
  • Statements

• Create (design) simple algorithms:
  • Solve problems by designing simple algorithms, e.g., basic calculations

• Create (design) small size programs using Python:
  • Hand trace code (programs) and predict results of executing code (determining contents of variables and output)
Today’s Menu

• Continue learning Python’s building blocks
  • Let’s have a look at the 2\textsuperscript{nd} concept “Actions”
    • Statements
    • Expressions

• Categories of Statements
  1. Assignment statement
  2. Input statement
  3. Output statement
  4. Operational statements
Statements

• Like most programming languages, Python programs are composed of statements
  • A Python program may contain many statements

• What is a statement?
  • 1 computer program instruction, made of expressions
  • A Python statement may contain many expressions

• What is an expression?
What is an expression?

**Expression**

**Definition:**

- Fragment of code
- Can be composed of...

**literal values**

examples of literal values: 3, 5.6, “Hello”

**built-in functions**

**methods**

**variables**

operators (+ operands)

examples of operators: +, -, *, /, **, string concatenation
Examples of Expressions

1. width/2
2. width/2.0
3. height/3
4. 1 + 2 * 5
5. delimiter * 5

Variables: width, height, and delimiter
Literal values: 2, 2.0, 3, 1, and 5
Operators: /, +, and *
Expressions are interpreted (i.e., executed or evaluated) by Python interpreter.

Once interpreted, an expression produces a result.

If the expression is part of a statement, its result is used to evaluate the rest of the statement.

Example: \texttt{print(len("Anne") + 26)}

```
<table>
<thead>
<tr>
<th>Expression 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression 2</td>
</tr>
<tr>
<td>Expression 3</td>
</tr>
<tr>
<td>Expression 4</td>
</tr>
</tbody>
</table>
```
Python statements

• Categories:
  1. Assignment statement
  2. Input statement
     • Conversion function
  3. Output statement
  4. Operational statements
     • Mathematical/arithmetic operator
     • String manipulation operator
  5. Conditional statement
  6. Iterative statement

Some of them are built-in function or method

We’ll see Strings in a few lectures!
We’ll see these statements soon!

Today
1. Assignment statement

- **Operator:** =
- **Syntax:** `<variable> = <expression>`

- **Examples:**
  ```python
  gradeMT = 0
  gradeMT = float(input("Please, enter MT grade: "))
  newGradeMT = gradeMT * percentOfFinalGradeMT / gradedOutOfMT
  ```

- **How it works:**
  1. The expression on the right of the assignment operator is evaluated
  2. If the variable on the left of the assignment operator does not exist ...
     - it is then created (given space in memory)
     - the result of the expression is assigned to this variable (stored in the space in memory)
   otherwise
     - its value is changed to (overwritten by) the result of the expression

- **Caution:** In Python, = does not mean equality, it means “assigns”
2. Input statement

- `input()` is a **built-in function**
- **Syntax** → `input(<prompt>)`

  - What is a **function**? It represents an action our program does
  - What is a **prompt**? It is an instruction to the user
    - Prompt must be a string (data of `str` data type)
  - What is a **user**? The person that executes our program and types in data

- **Example:**
  ```python
  gradeMT = float(input("Please, enter MT grade: "))
  ```

- **How it works:**
  1. The prompt is printed on the screen
  2. The user enters the requested data then presses **Enter**
  3. This data is returned from the function `input()` into our program as a **string**

- **GPS:** When prompting the user, we must give her/him clear and unambiguous instructions: the value to enter, the format of this value (if possible) and any other useful information such as the range of the value, etc...
Conversion functions

• `int()`
• `float()` are built-in functions
• `str()`

Syntax → `int(<expression>)`
   `float(<expression>)`
   `str(<expression>)`

• Examples
  `gradeMT = float(input("Please, enter MT grade: "))`
3. Output statement

• **print( )** is a **built-in function**

• **Syntax** -> **print(<argument(s)>)**
  
  • What is an **argument**? Data we want the function to use and act upon

• **Example:**
  
  ```python
  print("Final grade is %0.2f." %finalGrade )
  ```

• **How it works:**
  1. The value of the arguments are printed on the screen
  2. Often, format instructions are included in the arguments as in our example above

• The formatting instructions here are saying:
  - print the value of the expression (variable) **finalGrade** as a float value (**f** in **%0.2f**) with only 2 decimal numbers to the right of the decimal point (**0.2** in **%0.2f**)

• **GPS:** The output the program produces must be labelled clearly
GPS related to input and output statements and user interaction

- From a user perspective, i.e., from the perspective of the person typing her/his name, which user interaction would we prefer?
- Why?

User interaction # 1

```python
>>> name = input("name:")
name: Anne
>>> print(name)
Anne
```

User interaction # 2

```python
>>> name = input("Please, enter your first name: ")
Please, enter your first name: Anne
>>> print("You have entered " + "'" + name + "'" + " as your first name.")
You have entered 'Anne' as your first name.
```
4. Operational statement

• Mathematical/arithmetic operator:
  • Addition: +
  • Subtraction: -
  • Multiplication: *
  • Division: /
  • Floor division: //
  • Modulus: %
  • Exponentiation: **

• Syntax: <operand> <operator> <operand>

• Example:
newGradeMT = gradeMT * percentOfFinalGradeMT / gradedOutOfMT

< ... > signifies “replace with one operator”
< ... > signifies “replace with an operand, i.e., an integer or a float”
Let’s Practise!
Summary

• Python’s building blocks
  • Statements
  • Expressions

• Categories of Statements
  1. Assignment statement
  2. Input statement
  3. Output statement
  4. Operational statements
Next Lecture

• Continue learning Python statements (Python’s building blocks)
  • Categories of Statements
    4. Operational statements
      • Mathematical/arithmetic operators
        • Order of operations
        • Augmented assignment operators
      • Function terminology
    as we practice Step 5 of the software development process -> Testing and Debugging
  • 3 kinds of errors