Last Lecture

• We had a look at some algorithms commonly used in CS
Learning outcomes

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

• Create (design) small to medium size programs using Python:
  • Decompose a Python program into **functions**
• Use the core features of Python to design programs to solve problems: variables, expressions, terminal input and output, type conversion, conditionals, iteration, **functions**, standard library modules
• Design programs requiring approximately 100 lines and 6 **functions** (of well-designed code)
• Describe the benefits of using **functions**
• Construct **functions** such that:
  • Functions have a single purpose (decomposition)
  • Functions are reusable (generalisation)
  • Functions include parameters and local variables
  • Functions return values
• etc...
Today’s Menu

• Functions
  • Built-in
  • User-defined

• Create our own functions
  • Design function interface
  • Syntax
So far …

… we have used functions that were already built into Python by calling them

- **Built-in functions (some came from modules)**
  - For example: `print(...)`, `input(...)`, `type(...)`, `random.randint(1, 10)`

- **Built-in methods**
  - For example: `<string>.upper()`, `<string>.isalpha()`
Remember (from Lecture 8): Terminology related to functions

1. We **call** a function by name
   - Example of a **function call**:
     ```python
     userName = input("Please, enter your name: ")
     ```

2. The **name of the function** is **input**

3. The **expression(s)** in parentheses is (are) called the **argument(s) of the function**
   ```python
   userName = input("Please, enter your name: ")
   ```

4. We say that a function **takes** an argument(s)

5. The **result of the function** is what the function produces
   - In the above example, the **result of the function** is what the user has typed, i.e., her/his name, which is assigned to **userName**
     ```
     'Anne'
     ```

6. We say that a function **returns** a result and the result is called the **returned value**
Different kinds of functions

**Activity:** Fill this table with examples of built-in functions!

<table>
<thead>
<tr>
<th>Function takes arguments</th>
<th>Function does not take arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function returns a result (i.e., a returned value)</td>
<td></td>
</tr>
<tr>
<td>Function does not return a result (i.e., a returned value)</td>
<td></td>
</tr>
</tbody>
</table>
So, what is a function?

- **Description**: (adapted from our textbook)
  - A **named** sequence of Python statements that has **one purpose**
Are there “functions” in the real world?

1. Mom says to Louise “Go make your bed!“
   - **Purpose**: Make bed
   - **Parameter(s)**: None required
   - **Returned value**: None returned

2. Mom says to Louise “Here’s $5, go buy a bag of apples!“
   - **Purpose**: 
   - **Parameter(s)**: 
   - **Returned value**: 
Are there “functions” in the real world?

3. Mom says to Louise “Go find my cell phone!“
   - **Purpose:**
   - **Parameter(s):**
   - **Returned value:**

4. Mom says to Louise “Go put these bags on the kitchen counter!“
   - **Purpose:** Put bags on kitchen counter
   - **Parameter(s):**
   - **Returned value:**
Back to the computer world, let’s create our own functions

- **User-defined functions** (as opposed to **built-in functions**)

Step 1 - Problem Statement

• Let us create a function that returns the largest number out of 3 numbers

• Requirements:
  • cannot use the max( ) built-in function
  • cannot use the word max to name your function
Step 2 – Solution design (algorithm and data)

1. Purpose of function
2. Algorithm –> how the function achieve its purpose
3. Data –> Data needed by the function to achieve its purpose:
   • Input
     • Does the function need data from the main program? –> arguments/parameters
   • Local variables
     • Does the function need to create variables?
   • Output
     • Does the function need to return its result to the main program? –> returned value
Function Interface

1. Function name (purpose)
2. Parameters
3. Returned value
Step 4 – Implementation

Syntax of function definition

```python
def <functionName>( [parameter(s)] ) :
    < 1 or more statements >
    return [expression]
```

- `def` -> means “here is the definition of a function”
- `<functionName>` (Good Programming Style - GPS)
  - Function name is descriptive -> it describes its purpose
  - Function name syntax: same as for variable name syntax
- Function definition ->
- Header of a function ->
- Body of a function ->
- One `return` statement
Step 4 – Implementation

• Solution
  See LargestOf3.py on our course web site
Summary

• Functions
  • Built-in
  • User-defined

• Create our own functions
  • Design function interface
  • Syntax
Next Lecture

• Create our own functions
  • Docstring
  • Location of functions in program
• Guideline
  • Generalization
• Why creating functions