COMP 141

Functions

Rhodes College

1

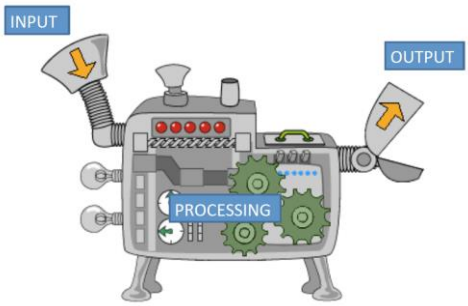
Announcements

Program 2 assigned - due 2/6 by 11:55pm

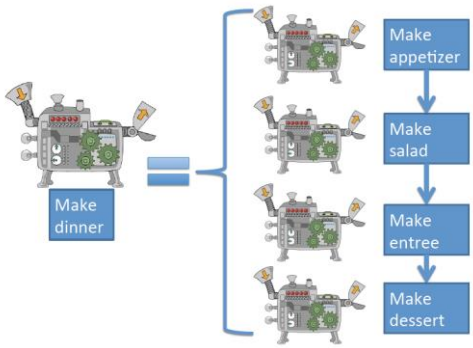
2

Practice from Last Time

Introduction to Functions



4



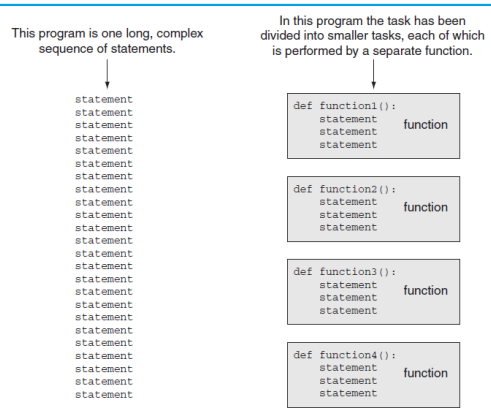
5

Introduction to Functions

- **Function:** group of statements within a program that perform as specific task
  - Usually one task of a large program
    - Functions can be executed in order to perform overall program task
  - Known as *divide and conquer* approach
- **Modularized program:** program wherein each task within the program is in its own function

6

Figure 3-1 Using functions to divide and conquer a large task



Benefits of Modularizing a Program with Functions

- **The benefits of using functions include:**
  - Simpler code
  - Code reuse
    - write the code once and call it multiple times
  - Better testing and debugging
    - Can test and debug each function individually
  - Faster development
  - Easier facilitation of teamwork
    - Different team members can write different functions

8

Defining a Function

Gives your function a name so it can be run later

- Syntax:

Function Header

```
def name():
    statement
    statement
    statement
    …
```

```
# Notice how these
# lines are indented.
# This is how Python knows
# where a function definition
# begins and ends.
```

Pick a name for your function that describes what it does! (Just like you pick variable names that describe what the variable holds.)

9

Defining and Calling a Function

- To use a function, we must ***define*** it first.
- After defining a function, to run the code inside, you ***call*** the function.
  - When a function is called:
    - Interpreter jumps to the function and executes statements in the block
    - Interpreter jumps back to part of program that called the function

10

Calling a Function

Runs the code inside the function definition

- Syntax:

```
name()
```

After defining a function, you can call it any number of times you want.

Each time it is called Python acts as if you had typed in all of the lines of the function definition.

11

The `main()` function

- From this point on, always ***define*** a `main()` function in your programs.
- Always ***call*** the `main()` function as the last line in your program.
- **`main` function: called when the program starts**
  - Calls other functions when they are needed
  - Defines the *mainline logic* of the program

12

Indentation in Python

- **Each block must be indented**
  - Lines in block must begin with the same number of spaces
    - Use tabs or spaces to indent lines in a block, but not both as this can confuse the Python interpreter
    - IDLE automatically indents the lines in a block
  - Blank lines that appear in a block are ignored

13

Example with Functions

- When a function is called, Python will
  - "jump" to the first line of the function's definition,
  - run all the lines of code inside the definition, then
  - "jump" back to the point where the function was called.

```
def twinkle():
  print("Twinkle twinkle little star")
  print("How I wonder what you are")

def main():
   twinkle()        # Call (run) the twinkle function.
   print("Up above the world so high")
   print("Like a diamond in the sky")
   twinkle()        # Call the twinkle function again.

main()              # Call main() to start the program.
```

14

Practice

- You are in charge of desserts at Thanksgiving dinner. You decide to make 2 pumpkin pies and 1 apple pie.
- Write a program that defines these functions:
  - `make_apple()` should print a description of how to make 1 apple pie
  - `make_pumpkin()` should print a description of how to make 1 pumpkin pie
  - `main()` should call `make_apple()` and `make_pumpkin()` appropriately to make all the pies.
- Don't forget to call main() at the end of your code!
- If you get done early, modify main() to ask the user for how many people are coming to dinner. If the number of people is more than 6, make 1 extra apple pie and 1 extra pumpkin pie.

15