Algorithms

- **algorithm**: A list of steps for solving a problem.
- Example algorithm: "Bake sugar cookies"
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven temperature.
  - Set the timer for 10 minutes.
  - Place the cookies into the oven.
  - Allow the cookies to bake.
  - Spread frosting and sprinkles onto the cookies.
  - ...

Problems with algorithms

- *lack of structure*: Many steps; tough to follow.
- *redundancy*: Consider making a double batch...
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven temperature.
  - Set the timer for 10 minutes.
  - Place the first batch of cookies into the oven.
  - Allow the cookies to bake.
  - Set the timer for 10 minutes.
  - Place the second batch of cookies into the oven.
  - Allow the cookies to bake.
  - Mix ingredients for frosting.
  - Spread frosting and sprinkles onto the cookies.

Structured algorithms

- **structured algorithm**: Split into coherent tasks.

  1 Make the batter.
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.

  2 Bake the cookies.
  - Set the oven temperature.
  - Set the timer for 10 minutes.
  - Place the cookies into the oven.
  - Allow the cookies to bake.

  3 Decorate the cookies.
  - Mix the ingredients for the frosting.
  - Spread frosting and sprinkles onto the cookies.

  ...

Removing redundancy

- A well-structured algorithm can describe repeated tasks with less redundancy.

1 Make the cookie batter.
- Mix the dry ingredients.
- ...

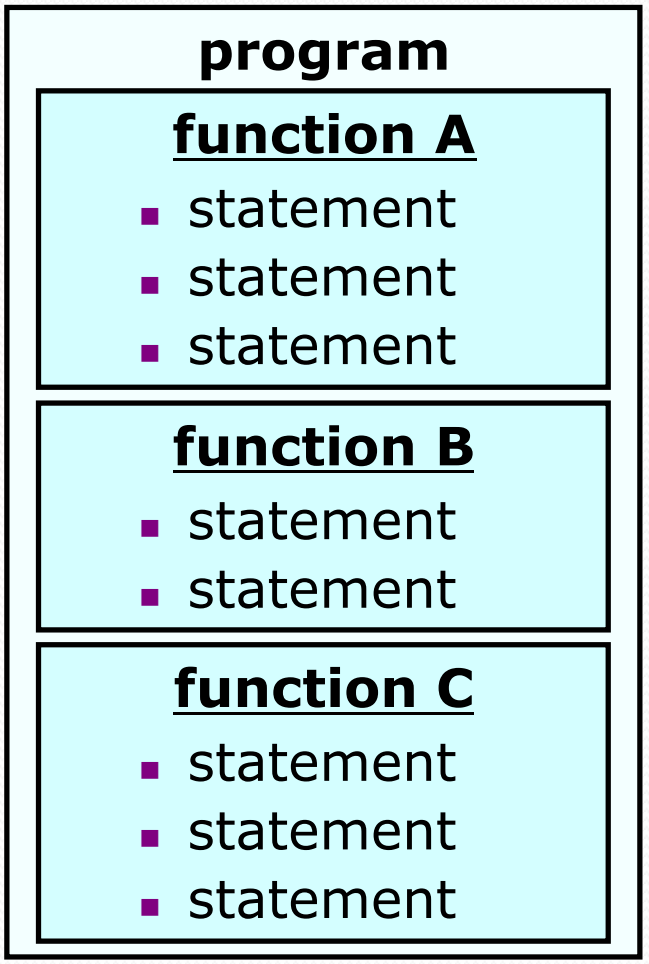
2a Bake the cookies (first batch).
- Set the oven temperature.
- Set the timer for 10 minutes.
- ...

2b Bake the cookies (second batch).
- Repeat Step 2a

3 Decorate the cookies.
- ...

Functions

- **function**: A named group of statements.
  - denotes the *structure* of a program
  - eliminates *redundancy* by code reuse
  - **procedural decomposition**: dividing a problem into sub-problems
- Writing a function is like adding a new command to Python.



Using functions

1. **Design** (think about) the algorithm.
   - Look at the structure, and which commands are repeated.
   - Decide what are the important overall tasks.

2. **Define** (write down) the functions.
   - Arrange statements into groups and give each group a name.

3. **Call** (run) the functions.

Defining a function

Gives your function a name so it can be run later

- Syntax:

```
def name():
    statement          # Notice how these
    statement          # lines are indented.
    ...                # This is how Python knows
    statement          # where a function definition
                       # begins and ends.
```

- Example:

```
def printWarning():
    print("This product causes cancer")
    print("in lab rats and humans.")
```

Calling a function

Executes (runs) the function's code

- Syntax:

```
name()
```

  - You can call the same function many times if you like.

- Example:

```
printWarning()
```

  - Output:

```
This product causes cancer
in lab rats and humans.
```

Program with functions

```
# This is a function to print the lyrics to my favorite song.
def rap():
    print("Now this is the story all about how")
    print("My life got flipped turned upside-down")

# A function for the "main" program.
def main():
    rap()                    # Call (run) the rap function.
    print()                  # Print a blank line.
    rap()                    # Call the rap function again.

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

Output:

```
Now this is the story all about how
My life got flipped turned upside-down

Now this is the story all about how
My life got flipped turned upside-down
```

Control flow

- When a function is called, the program's execution
  - "jumps" into that method, executing its statements, then
  - "jumps" back to the point where the method was called.

```
1  def rap():
2    print("Now this is the story all about how")
3    print("My life got flipped turned upside-down")

4  def main():
5     rap()                    # Call (run) the rap function.
6     print()                  # Print a blank line.
7     rap()                    # Call the rap function again.

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

Control flow

```
1  def rap():
2    print("Now this is the story all about how")
3    print("My life got flipped turned upside-down")

4  def main():
5     rap()                    # Call (run) the rap function.
6     print()                  # Print a blank line.
7     rap()                    # Call the rap function again.

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

- Program starts on line 8. Sees a function call…
  - Calls main(), jumps to line 5. Sees a function call…
    - Calls rap(), jumps to line 2. Runs statements 2 and 3.
  - Jumps back to after line 5. Runs statement 6. Sees a call…
    - Calls rap(), jumps to line 2. Runs statements 2 and 3.
  - Jumps back after line 7 [end of main()]
- Jumps back to after line 8 [end of entire program]

When to use functions

- Place statements into a function if:
  - The statements are related structurally, and/or
  - The statements are repeated.

Remember the cookies…

- Unstructured algorithm for a double batch:
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven temperature.
  - Set the timer for 10 minutes.
  - Place the first batch of cookies into the oven.
  - Allow the cookies to bake.
  - Set the timer for 10 minutes.
  - Place the second batch of cookies into the oven.
  - Allow the cookies to bake.
  - Mix ingredients for frosting.
  - Spread frosting and sprinkles onto the cookies.

Remember the cookies…

- Unstructured algorithm for a double batch:
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven temperature.
  - Set the timer for 10 minutes.
  - Place the first batch of cookies into the oven.
  - Allow the cookies to bake.
  - Set the timer for 10 minutes.
  - Place the second batch of cookies into the oven.
  - Allow the cookies to bake.
  - Mix ingredients for frosting.
  - Spread frosting and sprinkles onto the cookies.

Structured algorithms

- **structured algorithm**: Split into coherent tasks.

1 Make the batter.
- Mix the dry ingredients.
- Cream the butter and sugar.
- Beat in the eggs.
- Stir in the dry ingredients.

2 Bake the cookies.
- Set the oven temperature.
- Set the timer for 10 minutes.
- Place the first batch of cookies into the oven.
- Allow the cookies to bake.
- Set the timer for 10 minutes.
- Place the second batch of cookies into the oven.
- Allow the cookies to bake.

3 Decorate the cookies.
- Mix the ingredients for the frosting.
- Spread frosting and sprinkles onto the cookies.

Ideas for functions:
- Make one function for each of steps 1, 2, and 3.
- Make one function for the red/blue text.