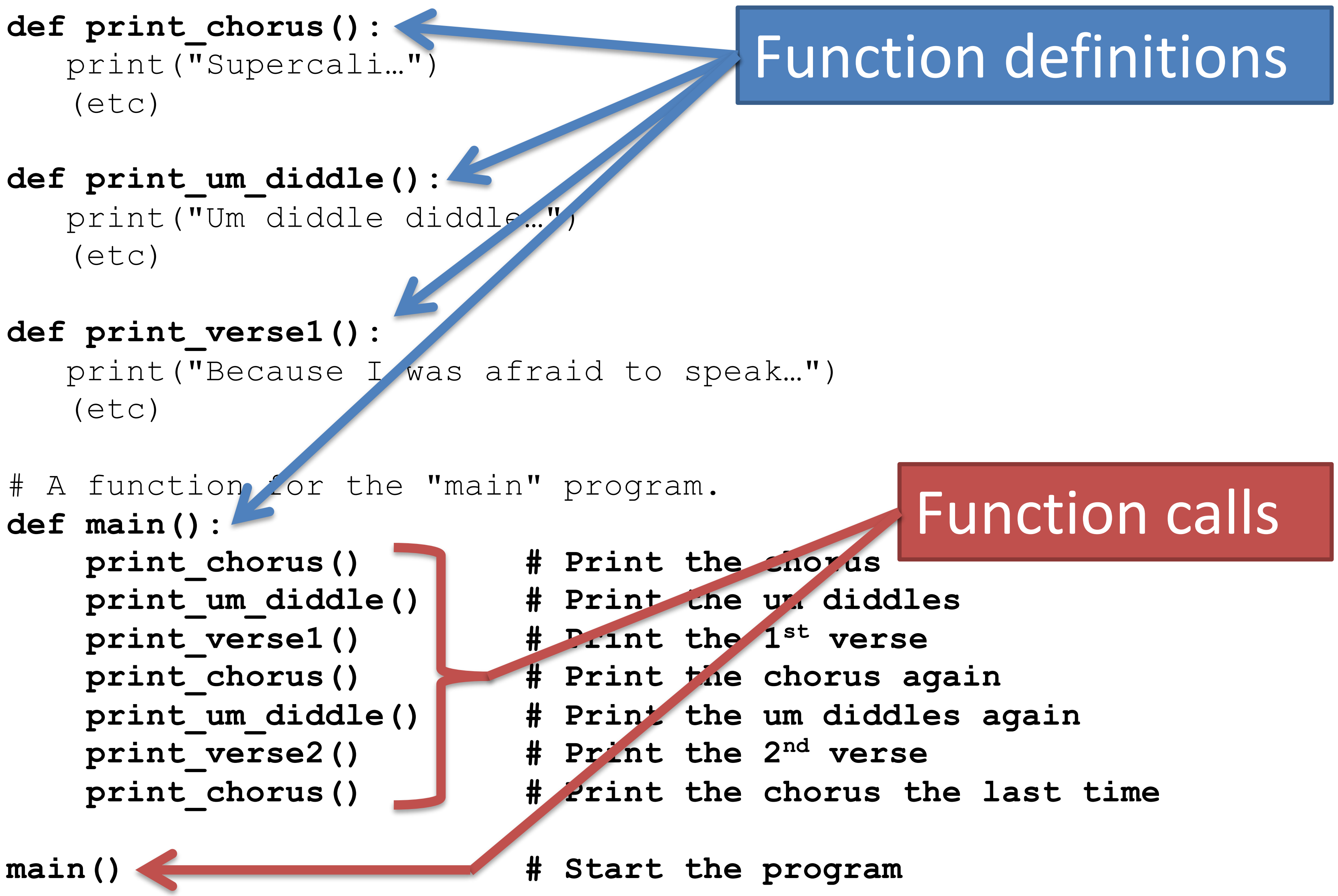
Functions

- Functions are groups of statements to which you give a name.
  - ***Defining*** a function uses the "`def`" keyword.
- That group of statements can then be referred to by that name later in the program.
  - ***Calling*** a function uses the name of the function then an opening/closing set of parentheses.



```
def print_chorus():
    print("Supercali…")
    (etc)

def print_um_diddle():
    print("Um diddle diddle…")
    (etc)

def print_verse1():
    print("Because I was afraid to speak…")
    (etc)

# A function for the "main" program.
def main():
     print_chorus()         # Print the chorus
     print_um_diddle()      # Print the um diddles
     print_verse1()         # Print the 1st verse
     print_chorus()         # Print the chorus again
     print_um_diddle()      # Print the um diddles again
     print_verse2()         # Print the 2nd verse
     print_chorus()         # Print the chorus the last time

main()                      # Start the program
```

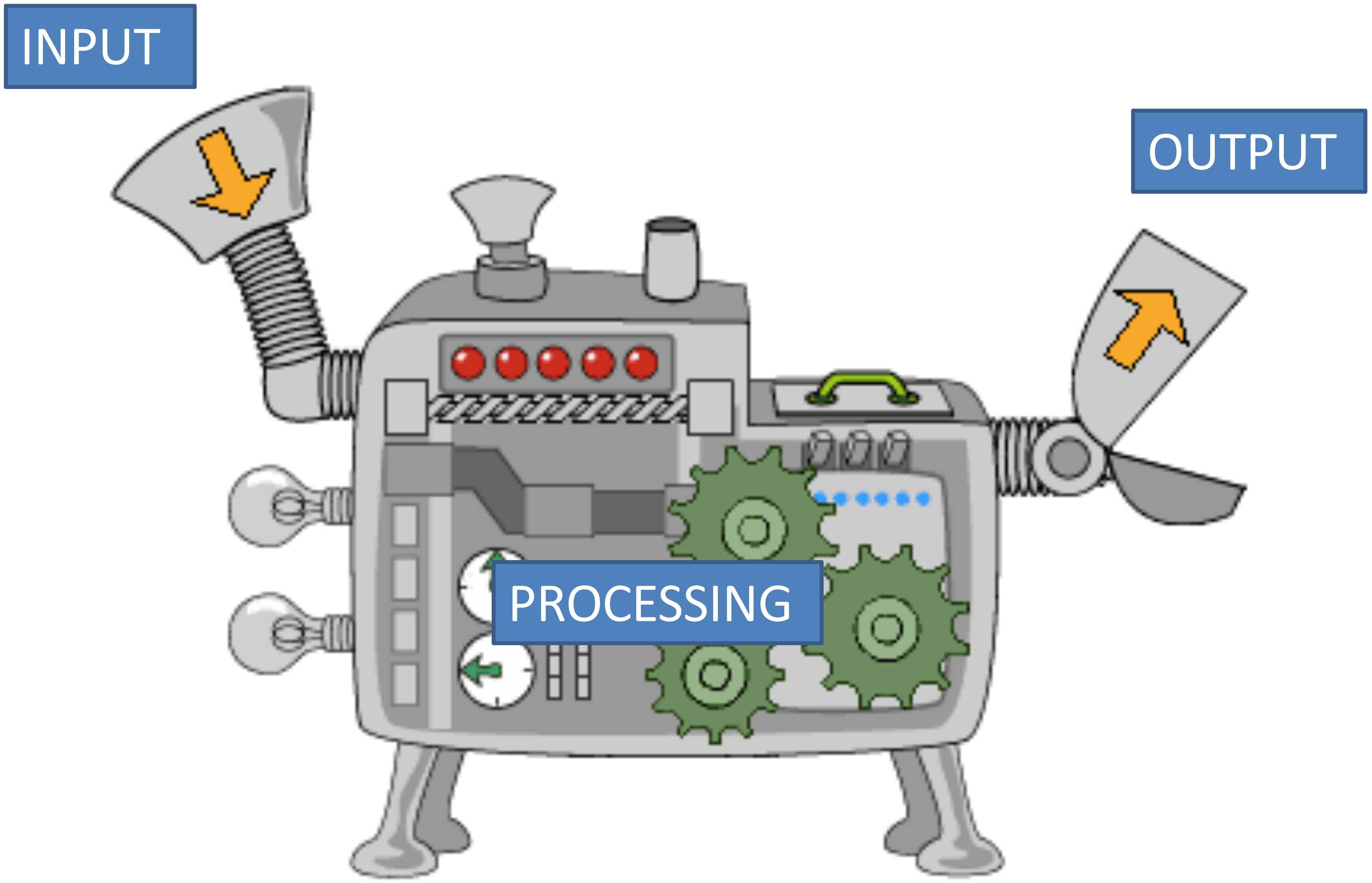
- When a function is called, Python will
  - "jump" to the first line of the function's definition,
  - run all the lines of code inside the definition, then
  - "jump" back to the point where the function was called.

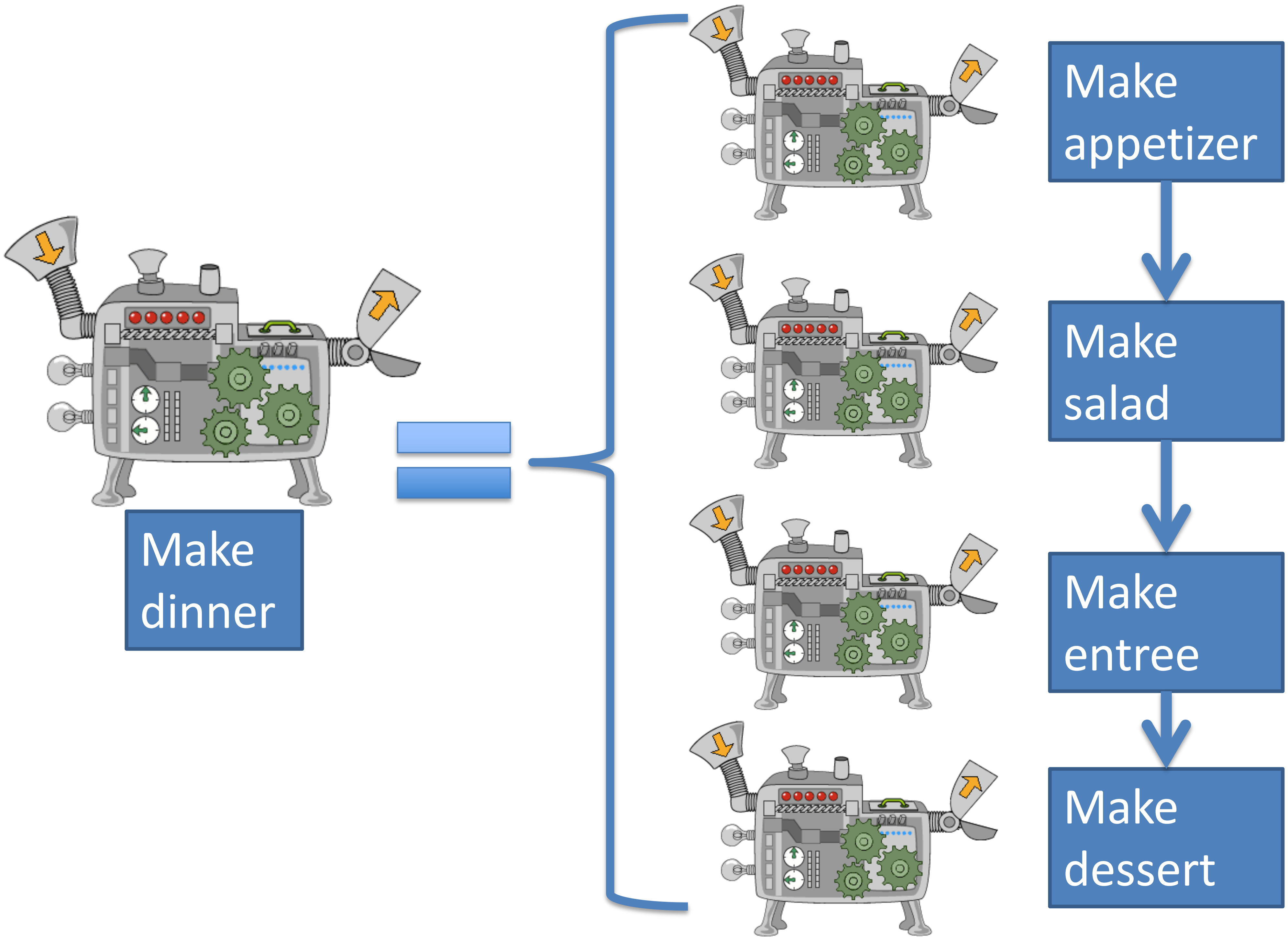
- When a function is called, Python will
  - "jump" to the first line of the function's definition,
  - run all the lines of code inside the definition, then
  - "jump" back to the point where the function was called.

```
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.
```


INPUT
OUTPUT
PROCESSING


Make dinner
Make appetizer
Make salad
Make entree
Make dessert

- So far we know:

- Input:
  - input statement

- Output:
  - print statement

- Processing:
  - math, variable assignments, if statements

- Suppose we want to write a program to sing "Happy Birthday" to the user, who also has a twin sibling.
- If we think of "sing Happy Birthday" as an algorithm, what information does the algorithm require as input?

Arguments and Parameters

- Algorithms described by functions allow for ***input*** via arguments and parameters.
- This method allows you to send information into a function to change its behavior when it runs.

Arguments and parameters

Defining:

```
def name_of_function(param1, param2, …):
    statement
    statement
    statement
```

- ***Parameters*** are variables placed inside the parentheses when a function is ***defined***.
- They should represent pieces of information that the function needs to know ahead of time in order to run.

```
def sing_song(name):
  print("Happy bday to you, happy bday to you!")
  print("Happy bday dear", name, "happy bday to you")
```

- The statements inside a function definition can use the parameters as normal variables.

Arguments and parameters

Defining:

```
def name_of_function(param1, param2, …):
    statement
    statement
    statement
```

Calling:

```
name_of_function(arg1, arg2, …)
```

The values being copied from the calling function are called ***arguments***.
The variables being copied into are called ***parameters***.

You've seen arguments already

- `name = input("What is your name? ")`
- `x = 5`
- `y = 2`
- `print("x is", x, "y is", y)`
- `print("their sum is", x + y)`

Arguments can be variables, literals, or math expressions.

Determining good parameters

- In an algorithm for computing the area of a rectangle, what information is needed?
- In calculating whether a number is even, what information is needed?

- What if we want our program to ask for the user's and twin's names?

```python
def sing_song(name):
  print("Happy bday to you, happy bday to you!")
  print("Happy bday dear", name, "happy bday to you")

def main():
  my_name = input("What is your name? ")
  sing_song(my_name)
  twin_name = input("What is your twin's name? ")
  sing_song(twin_name)

main()
```

```
def sing_song(name):
  print("Happy bday to you, happy bday to you!")
  print("Happy bday dear", name, "happy bday to you")

def main():
  my_name = input("What is your name? ")
  sing_song(my_name)
  twin_name = input("What is your twin's name? ")
  sing_song(twin_name)

main()
```

When Python runs the red line, it copies the value of my_name into `sing_song`'s variable name.

```
def sing_song(name):
    print("Happy bday to you, happy bday to you!")
    print("Happy bday dear", name, "happy bday to you")

def main():
    my_name = input("What is your name? ")
    sing_song(my_name)
    twin_name = input("What is your twin's name? ")
    sing_song(twin_name)

main()
```

When Python runs the blue line, it copies the value of twin_name into sing_song's variable name.

```
def sing_song(name):
  print("Happy bday to you, happy bday to you!")
  print("Happy bday dear", name, "happy bday to you")

def main():
  name = input("What is your name? ")
  sing_song(name)
  name = input("What is your twin's name? ")
  sing_song(name)

main()
```

- You *may* use the same variable names in both places, if desired.
- Each function then has its own copy of the variable.
- There is no permanent link between the variables.

Local variables

- Any variable used as a parameter inside a function is "owned" by that function, and is ***invisible*** to all other functions.
- These are called ***local variables*** because they can only be used "locally" (within their own function).
- Any variable created inside a function is also a local variable and cannot be seen outside of that function.

```
def some_function(x):
   print(“Inside the function, x is”, x)
   x = 17
   print(“Inside the function, x is changed to”, x)

def main():
   x = 2
   print(“Before the function call, x is”, x)
   some_function(x)
   print(“After the function call, x is”, x)

main()
```

Output:

```
Before the function call, x is 2
Inside the function, x is 2
Inside the function, x is 17
After the function call, x is 2
```

Wait. What?

- There is no permanent connection between the `x` in `main` and the `x` in `some_function`.
- Arguments are passed --- one way only --- from `main` to `some_function` when `main` calls `some_function`.
  - This copies main's value of `x` into `some_function`'s `x`.
- Any assignments to `x` inside of `some_function` do not come back to `main` .

- You no longer have a twin. Now you have a sibling that is two years older than you, but you still share the same birthday.
- Edit birthday.py so sing_song now will print the lyrics ***but also print how old the person is***.
- Add a second parameter to sing_song called age.
- Edit main() to ask for your age, as well as your name and sibling's name.
- Edit the two calls to sing_song so appropriate ages are passed as arguments.