Fundamentals of Programming

Guy W. Dayhoff II
Department of Chemistry
University of South Florida

Part 4: Programming in C and Python

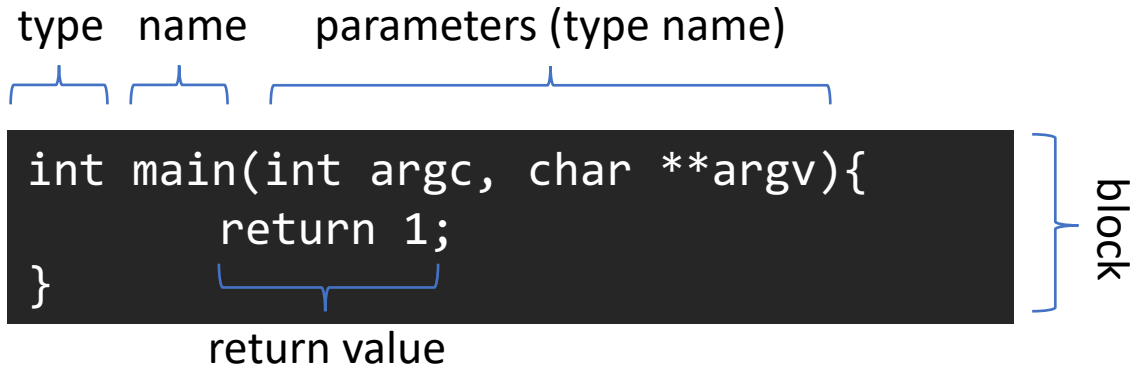
Intro to C

- Procedural programming language
- Programs are compiled
- High performance
- Underpins numerous other languages
- Released in 1972
- Developed to make UNIX portable
- Data types are explicit
- Requires a ‘main’ function
- Most popular language as of January 2021

C Syntax

Functions:

- **Functions* *must* have a data type**
- **Functions *must* have a unique name**
- **Functions can have any # of parameters**
- **Functions with the *void* data type do not require a return statement.**
- **When a function is called, control is given to the function until it is *returned* to the block in which the function was called.**




```
void greet(char *name){
        printf("Hello, %s!\n", name);
}
```

```
char *knock_knock(void){
        return "who's there";
}
```

*All data types are explicit, so when you declare a variable you must specify its expected data type.

C Syntax

Conditionals:

- Conditional statements evaluate to TRUE (1) or FALSE (0)
- Logical operators allow complex conditions to be used
- Conditionals can be *negated* with the ! operator
- Relational operators are also available
  - >
  - <
  - >=
  - <=
  - ==
  - !=

```
if (condition) {
        //do something
}
```

```
if (number_1 > number_2) {
        //do something
}
```

```
If (condition_1 && condition_2) {
        //do something
else if (condition_3) {
        //do something else
}
else {
        //do another thing
}
```

```
If ((!condition_1 && (condition_2 || condition_3)) {
        //do something
}
```

C Syntax

Loops:

- **Three types:**
  - **while**
  - **do/while**
  - **for**

```
while(condition){
        //do something
}
```

```
for(;condition;) {
        //do something
}
```

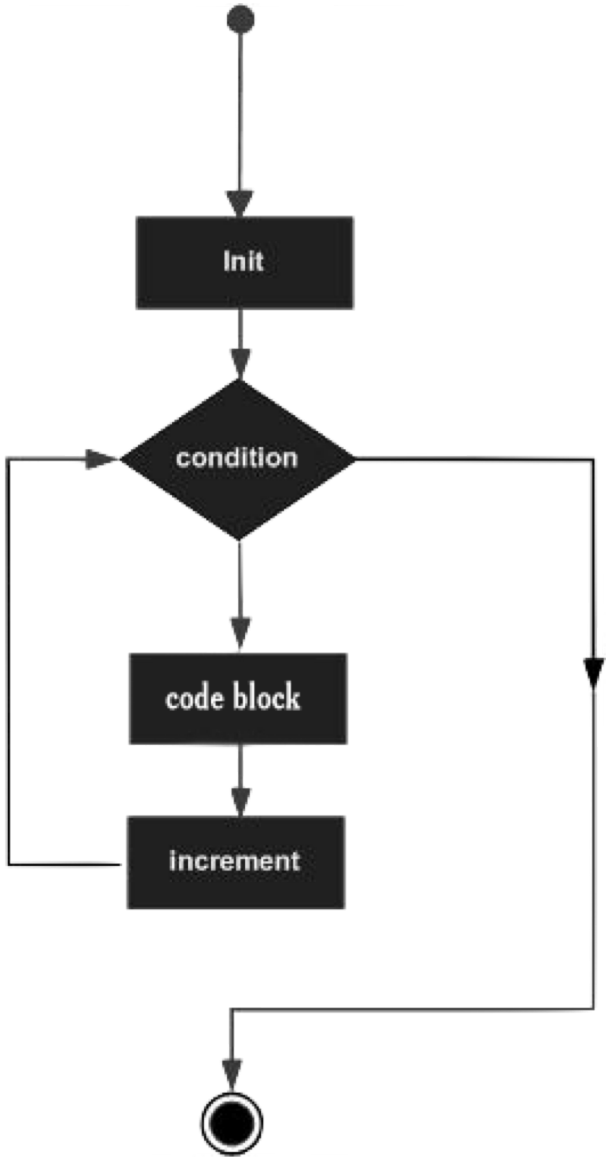
```
do {
        //something
} while (condition);
```

```
for(init; condition; increment) {
        //do something
}
```

C Syntax

```
for(init; condition; increment) {
        //do something
}
```

```
int x;
for(x = 99; x > 0; x--) {
        printf("%d bottles of beer on the wall",x);
}
```



C Example #1: Hello World!

1. **Create hello.c**

```
#include <stdio.h>

int main(int argc, char **argv){
        puts("Hello, World!");
        return 1;
}
```

Use text editor of your choice

2. **Compile program with gcc**

```
gcc hello.c –o hello
```

3. **Execute the program**

```
./hello
```

Use a command-line interface

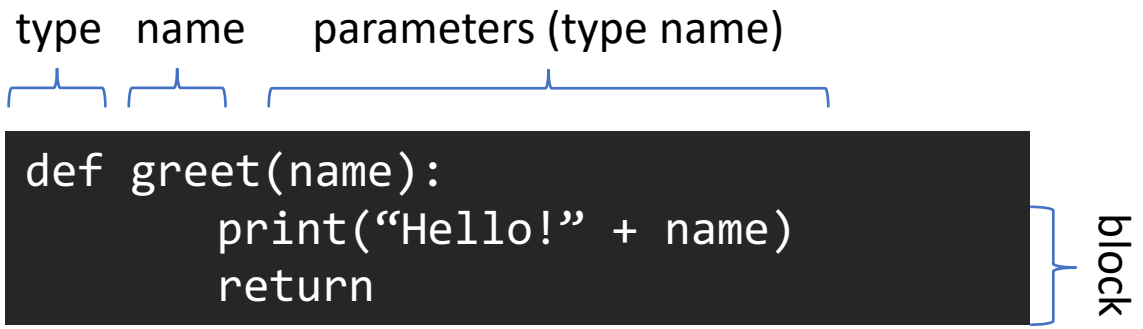
Intro to Python

- Object-oriented programming language
- Programs are interpreted
- Fast development time
- Indentation dependent
- Released in 1991
- Data types are implicit
- Doesn’t require a ‘main’ function
- Third most popular language as of January 2021

Python Syntax

Functions:

- **Functions *must* have a unique name**
- **Functions can have any # of parameters**
- **Functions can return multiple values**
- **When a function is called, control is given to the function until it is *returned* to the block in which the function was called.**
- **Parameters can have default values**




```
def get_name(employee_id):
        #lookup name using ID
        return first_name,last_name
```

```
def repeat(repetitions=100):
        #do something
        return
```

***All data types are implicit, so when you declare a variable you do NOT need to specify expected data type.**

Python Syntax

Conditionals:

```
if x < 100:
        print(“It’s less than 100!”)
elif x > 9000:
        print(“It’s over 9000!”)
else:
        print(“It’s somewhere”)
```

Loops:

- **Two types:**
  - **while**
  - **for**

```
while condition:
        #do something


for x in range(10):
        #do something with x
```

Python Example #1: Hello World!

1. **Create hello.py**

```
print(“Hello, World!)
```

Use text editor of your choice

2. **Execute the program**

```
python3 hello.py
```

Use a command-line interface

Part 5: Practicing C and Python

*The code examples in this section will most likely not run if copy and pasted, because powerpoint converts Double quotes into unique characters not understood by the C compiler and Python interpreter. I encourage you to write the code yourself, even if you copy it, rather than copy and paste it, however know that if you do copy and paste the code you will have to correct some formatting issues caused by powerpoint such as the the quotes (for C ad Python) and indentations (for python).

Activities

1. Create a flowchart
2. Write pseudocode
3. Implement in C
4. Implement in Python

Activity #1: “Hello, World!”

1. Create a flowchart
2. Write pseudocode
3. Implement in C
4. Implement in Python

Write a program that:

1. **Prints ”Hello, World!” to the screen**
2. **Writes “Hello, World!” to a file**
3. **Greets 10 different worlds, printing to screen.**

   “Hello, 1st World!”

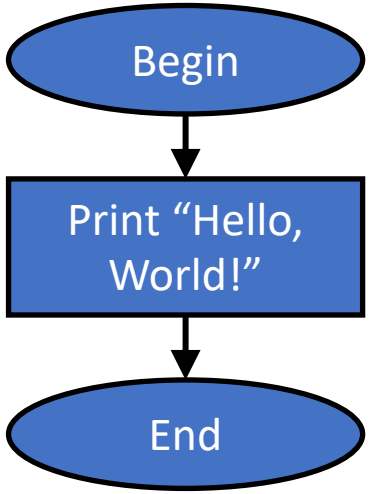
   Hello, 2nd World!”

   …

   ”Hello, 10th World”

Activity #1: “Hello, World!”

- **Flowchart 1**



- **Pseudocode 1**

```
Print “Hello, World!” to the screen.
```

Activity #1: “Hello, World!”

- **C Code 1**

```
#include <stdio.h>

int main(int argc, char **argv){
        puts(“Hello, World!”);
        return 1;
}
```

```
#include <stdio.h>

int main(int argc, char **argv){
        printf(“%s”,“Hello, World!”);
        return 1;
}
```

```
#include <stdio.h>

int main(int argc, char **argv){
        printf(“Hello, World!”);
        return 1;
}
```

```
#include <stdio.h>

int main(int argc, char **argv){
        fprintf(stdout, “Hello, World!”);
        return 1;
}
```

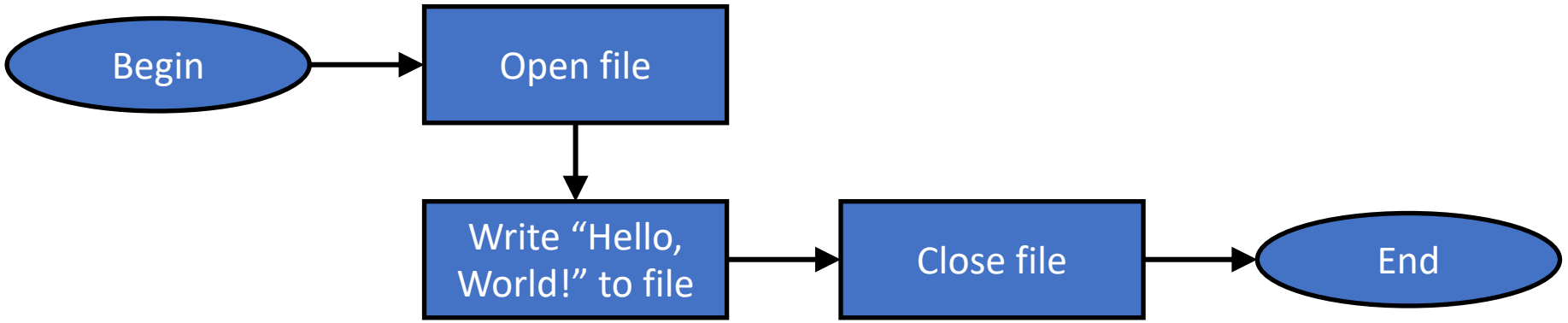
- **Python Code 1**

```
print(“Hello, World!)
```

```
print(“%s” % “Hello, World!”)
```

Activity #1: “Hello, World!”

- **Flowchart 2**



- **Pseudocode 2**

```
Open a file named “hello.txt”.
Write ”Hello, World!” to the file
Close the file.
```

Activity #1: “Hello, World!”

- **C Code 2**

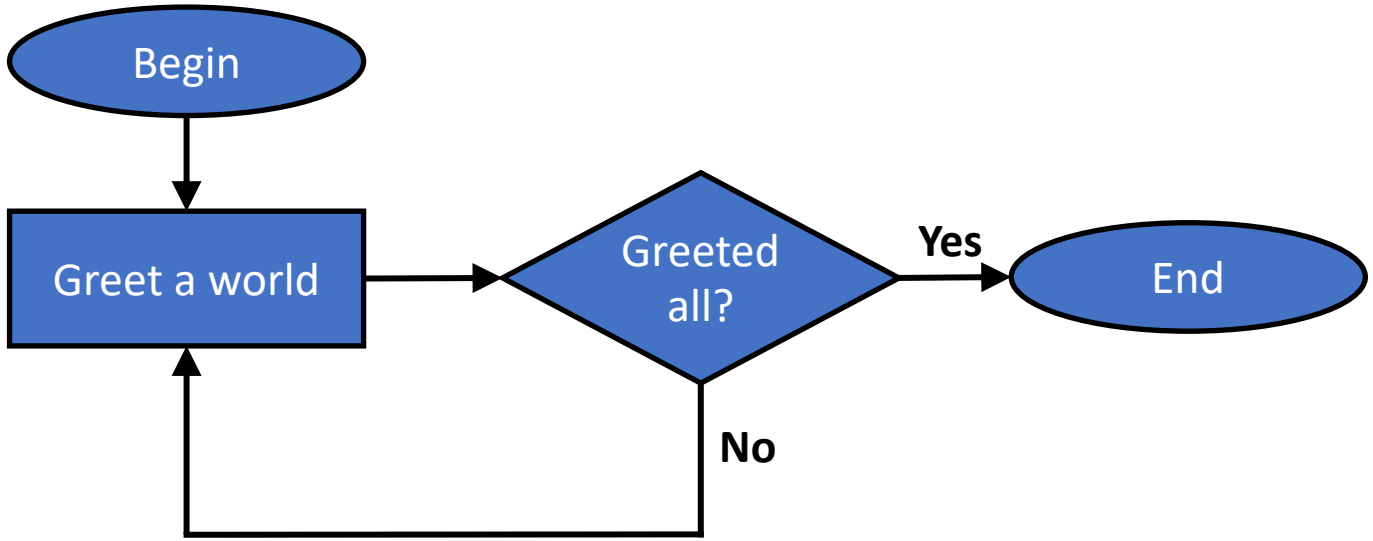
```
#include <stdio.h>

int main(int argc, char **argv){
        FILE *fp = fopen(“hello.txt”,”w”);
        fprintf(fp, “Hello, World!”);
        fclose(fp);
        return 1;
}
```

- **Python Code 2**

```
f = open(“hello.txt”,”w”)
f.write(“Hello, World!)
f.close()
```

Activity #1: “Hello, World!”

- **Flowchart 3**



- **Pseudocode 3**

```
For each world
        Greet the world
```

Activity #1: “Hello, World!”

- **C Code 3**

```
#include <stdio.h>

int main(int argc, char **argv){
        for(int i=1;i<11;i++){
           printf(“Hello, World %d!\n”, i);
        }
        return 1;
}
```

```
#include <stdio.h>

int main(int argc, char **argv){
        for(int i=1;i<11;i++){
           char *suffix;
           if (i==1)
                suffix = “st”;
           else if (i==2)
                 suffix = “nd”;
           else if (i==3)
                suffix = “rd”;
           else
                suffix = “th”;
           printf(“Hello, %d%s World!\n”,
                 i,suffix);
        }
        return 1;
}
```

Activity #1: “Hello, World!”

- **C Code 3 [continued]**

```
#include <stdio.h>

int main(int argc, char **argv){
        for(int i=1;i<11;i++){
           char *suffix;
           switch(i){
                case 1:  suffix=“st”; break;
                case 2:  suffix=“nd”; break;
                case 3:  suffix=“rd”; break;
                default: suffix=“th”; break;
           }
           printf(“Hello, %d%s World!\n”,
                 i,suffix);
        }
        return 1;
}
```

In C, switch works on integer values.

```
#include <stdio.h>

int main(int argc, char **argv){
        for(int i=1;i<11;i++){
           char *suffix;
           if (i==1)
                suffix = “st”;
           else if (i==2)
                 suffix = “nd”;
           else if (i==3)
                suffix = “rd”;
           else
                suffix = “th”;
           printf(“Hello, %d%s World!\n”,
                 i,suffix);
        }
        return 1;
}
```

Activity #1: “Hello, World!”

- **Python Code 3**

```
for i in range(1,11):
        print(“Hello, World %d” % i)
```

```
for i in range(1,11):
        if i == 1:
            suffix=“st”
        elif i == 2:
            suffix=“nd”
        elif i == 3:
            suffix=“rd”
        else:
            suffix=“th”

        print(“Hello, World %d%s” % (i,suffix))
```

In Python, there is no switch statement.

Activity #2: Interactive & Batch Averaging

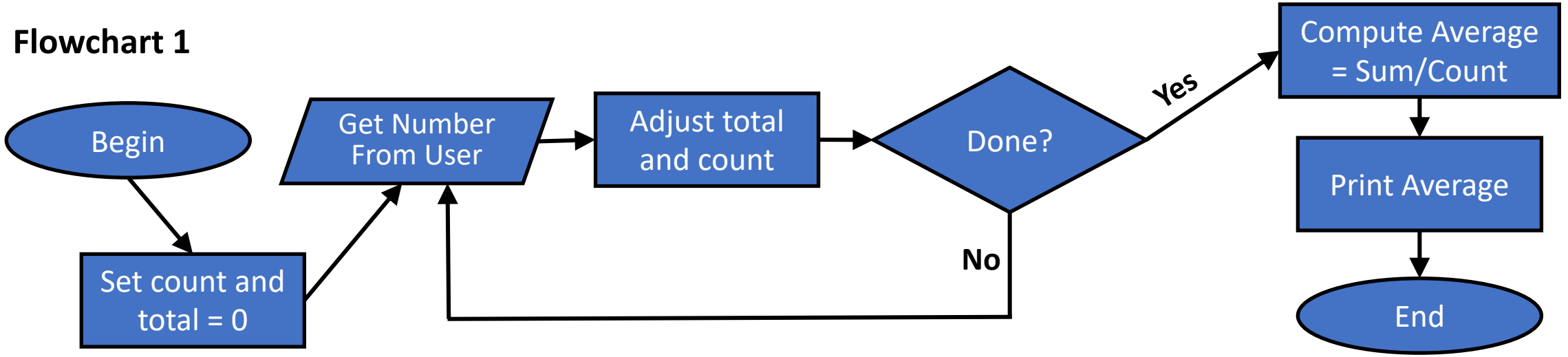
1. Create a flowchart
2. Write pseudocode
3. Implement in C
4. Implement in Python

Write a program that:

1. **Takes some number, n, of user inputs , computes their average and prints it to the screen**
2. **Reads a file containing a column of values and**
   a) **computes their average and prints it to the screen**
   b) **computes a running average and writes it to a file (view this as a plot vs number of samples)**

Activity #2: Interactive & Batch Averaging

- **Flowchart 1**



- **Pseudocode 1**

```
Set count and total to zero.
Get a number from the user.
Add the number to the total.
Increment the count.
If the user wants to enter another number return to the first step.
Compute the average as total/count.
Print the average.
```

Activity #2: Interactive & Batch Averaging

- **C Code 1**

```
#include <stdio.h>

int main(int argc, char **argv){
  //declare variables
  int count, total, number;

  //prompt user for a number, then read numbers, one by one until we get a letter
  printf("Enter a number:");
  for (count = total = 0; scanf("%d",&number); count++){
    total += number;
    printf("Enter another number (or any letter to end):");
  }

  //compute then print the average as a float
  float average = (float)total/(float)count;
  printf("The average is: %f\n", average);
  return 1;
}
```

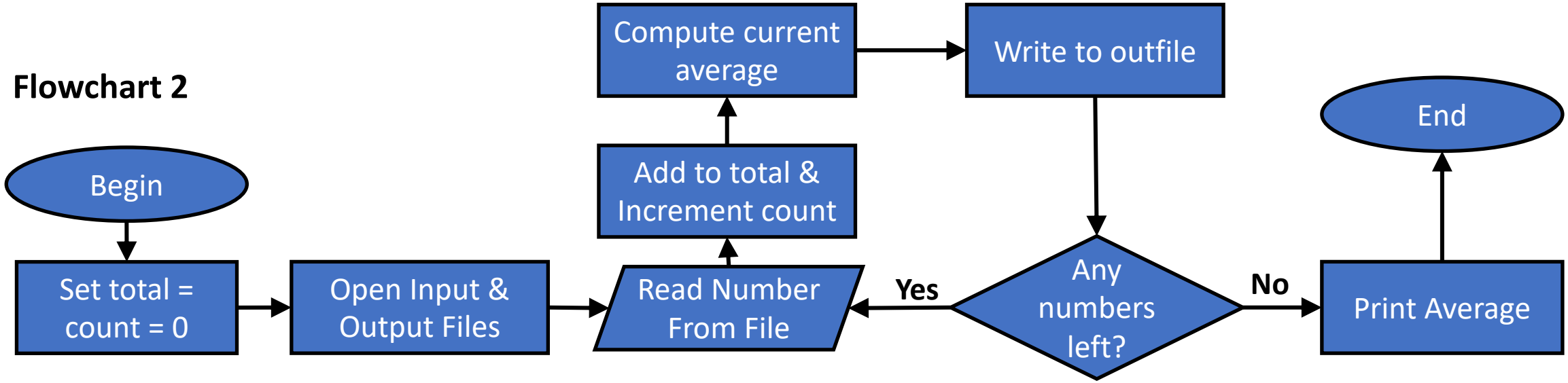
Activity #2: Interactive & Batch Averaging

- **Python Code 1**

```
total = count = 0
while(True):
  try:
    if count == 0:
      print("Enter a number:")
    else:
      print("Enter another number or any letter to finish:")
    number = int(raw_input())
    total += number
    count += 1
  except ValueError:
    break
print("The average is %f\n" % (total/count))
```

Activity #2: Interactive & Batch Averaging

- **Flowchart 2**



- **Pseudocode 2**

```
Set count and total both to zero.
Open the input and output files.
For every number in the input file:
        Read the number.
        Add the number to the total.
        Increment the count by one.
        Compute current average = total/count.
        Write count and current average to the outfile.
Print the final average.
```

Activity #2: Interactive & Batch Averaging

- **C Code 2**

```
#include <stdio.h>

int main(int argc, char **argv){
  int count, total, number; float average;

  FILE *infile = fopen("input.dat","r");
  FILE *outfile = fopen("output.dat","w");

  for (count = total = 0; fscanf(infile,"%d",&number) != EOF;){
    count++; total += number;
    average = (float)total/(float)count;
    fprintf(outfile,"%d %f\n",count,average);
  }
  printf("The average is: %f\n", average);

  fclose(infile);
  fclose(outfile);
  return 1;
}
```

Activity #2: Interactive & Batch Averaging

- **Python Code 2**

```
total = count = 0.0
infile = open("input.dat","r")
outfile = open("output.dat","w")
for line in infile:
  try:
    number = int(line)
    total += number
    count += 1
    outfile.write("%d %f\n" % (count,total/count))
  except ValueError:
    break
print("The average is %f\n" % (total/count))
infile.close()
outfile.close()
```

Activity #3: Filtering & Sorting

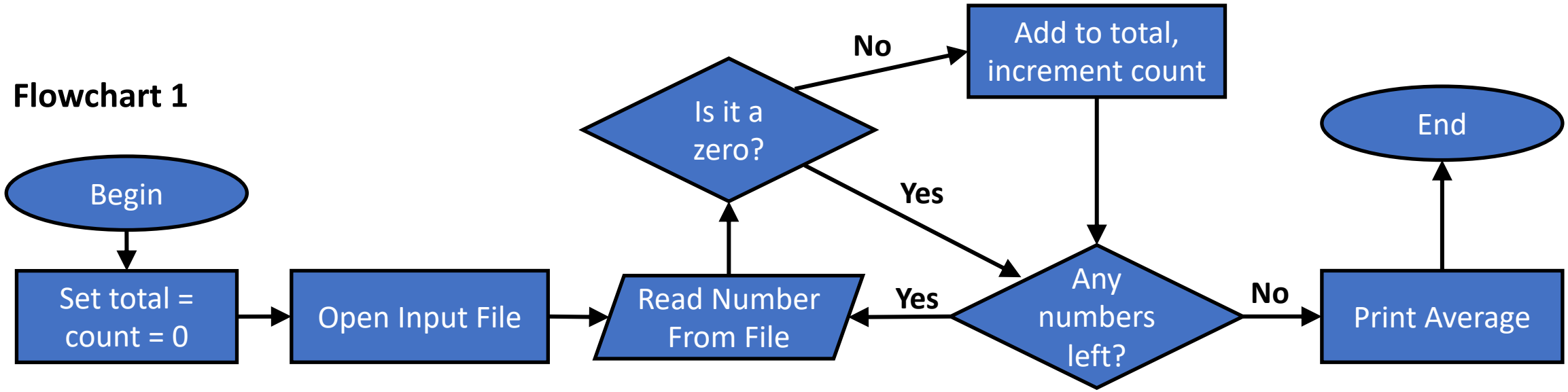
1. Create a flowchart
2. Write pseudocode
3. Implement in C
4. Implement in Python

Write a program that:

1. **Reads values from a file and computes the average of non-zeroes entries and prints it to the screen**
2. **Reads values from a file, removes negative values, and writes the remaining values to a file in descending order**

Activity #3: Filtering & Sorting

- **Flowchart 1**



- **Pseudocode 1**

```
Set count and total to zero.
Open input file
For each number in the file
        if the number is zero do nothing
        else add the number to the total and increment the count
Print the average (i.e. total/count).
```

Activity #3: Filtering & Sorting

- **C Code 1**

```
#include <stdio.h>

int main(int argc, char **argv){
  int count, total, number; float average;

  FILE *infile = fopen("input.dat","r");

  for (count = total = 0; fscanf(infile,"%d",&number) != EOF;){
    if (number == 0):
        continue;
    count++; total += number;
  }

  average = (float)total/(float)count;
  printf("The average is: %f\n", average);

  fclose(infile);
  return 1;
}
```

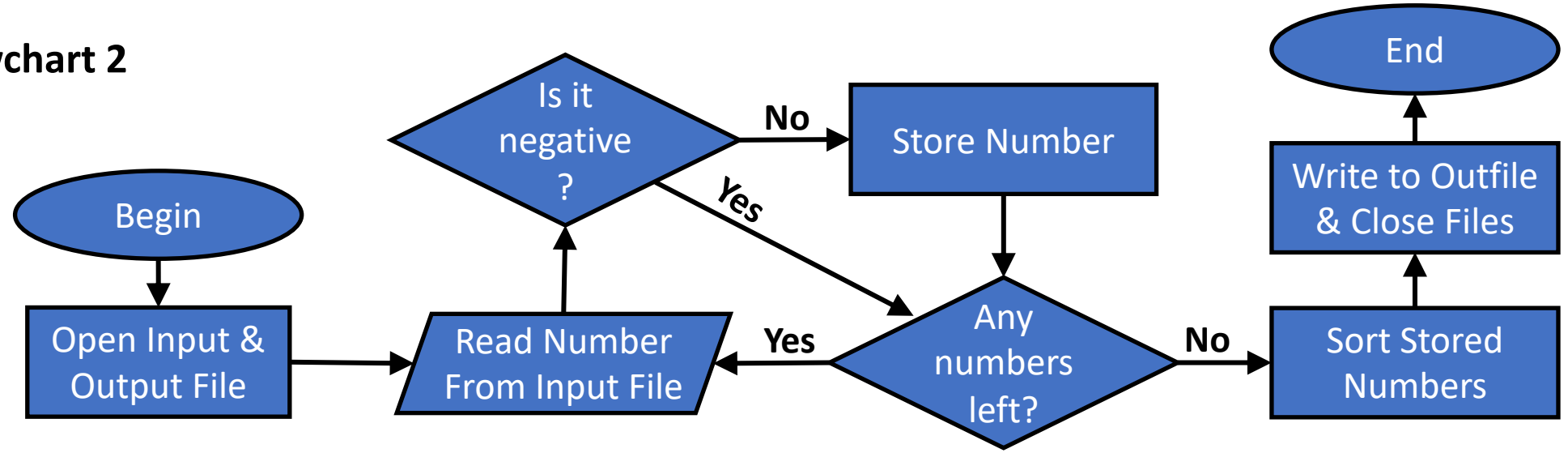
Activity #3: Filtering & Sorting

- **Python Code 1**

```
total = count = 0.0
infile = open("input.dat","r")
for line in infile:
  try:
    number = int(line)
    if number is 0:
      continue
    total += number
    count += 1
  except ValueError:
    break
print("The average is %f\n" % (total/count))
infile.close()
```

Activity #3: Filtering & Sorting

- **Flowchart 2**



- **Pseudocode 2**

```
Open input and output file.
For each number in the input file
        if the number is negative do nothing
        else store the number for later use
Sort stored numbers
Write sorted numbers to the outfile.
Close input and output files
```

Activity #3: Filtering & Sorting

- **C Code 2**

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){
  int count, total, number;
  int *numbers = malloc(0);

  FILE *infile = fopen("input.dat","r");
  FILE *outfile = fopen("output.dat","w");

  for (count = total = 0; fscanf(infile,"%d",&number) != EOF;){
    if (number < 0)
      continue;

    count++;
    numbers = realloc(numbers,sizeof*numbers*count);
    numbers[count-1] = number;
  }
```

Activity #3: Filtering & Sorting

- **C Code 2 [continued]**

```
  int *sorted = malloc(sizeof*sorted*count);
  int i,j,k;

  for (i=0;i<count;i++){
    sorted[i] = -1; //set current slow to definite loser
    for (j=0;j<count;j++){
      if (numbers[j] > sorted[i]){
        sorted[i] = numbers[j]; //store the winning value
        k = j; //track the index of the winner
      }
    }
    numbers[k] = -1; //remove the winner from further consideration
  }
  for (i=0;i<count;i++)
    fprintf(outfile,"%d\n",sorted[i]);

  free(sorted); fclose(infile); fclose(outfile);
  return 1;
}
```

Activity #3: Filtering & Sorting

- **Python Code 2**

```
infile = open("input.dat","r")
outfile = open(“output.dat”, “w”)
numbers = []

for line in infile:
  try:
    number = int(line)
    if number < 0:
      continue
    numbers.append(number)
  except ValueError:
    break

numbers.sort(reverse=True)
for number in numbers:
    outfile.write(number)

infile.close()
outfile.close()
```

Fin

Questions?