Review of Programming Fundamentals

TECH 3230 – Lecture 1

Prof Kohn

Presentation created with the help of ChatGDP and Claude.ai

THE UNIVERSITY OF MEMPHIS.

1

Programming Fundamentals

Overview of key concepts:

- Why C
- Variables
- Control Structures
- Loops
- Functions
- Data Structures
- Modules/Libraries

- We will compare Python (as taught in TECH 1211) and C

THE UNIVERSITY OF MEMPHIS.

2

Why C

- C has been around since 1972
- Well suited to embedded programming where access to the hardware of a system is required.
- C programming tools produce small and fast programs
- C programming is widely used in embedded systems

- According to Gemini “The undisputed king of embedded systems development remains C”

THE UNIVERSITY OF MEMPHIS.

3

Anatomy of a Program

Same numbered pieces on each side line up conceptually

Python

```
1  import math

2  def greet(name):
       print(f"Hello, {name}")

3  def main():
4      greet("World")

3  if __name__ == "__main__":
4      main()
```

C

```
1  #include <stdio.h>

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

3  int main() {
4      greet("World");
       return 0;
   }
```

THE UNIVERSITY OF MEMPHIS

4

How the Pieces Relate

The same four structural roles, expressed with different syntax

	PYTHON	C
1 Bring in outside code	`import math` — *imports a module, runs any top-level code in it*	`#include <stdio.h>` — *pastes in declarations before compiling*
2 Define a function	`def greet(name):` — *indentation marks the function body*	`void greet(char *name) { }` — *braces mark the body; return type is explicit*
3 Program entry point	`if __name__ == "__main__":` — *a convention — Python runs top to bottom*	`int main() { }` — *required — the OS always calls main() first*
4 Call a function / exit	`main()` — *just a function call, no return value needed*	`greet("World"); return 0;` — *return 0 reports success to the OS*

THE UNIVERSITY OF MEMPHIS

5

Variables

Python:

- x = 10

C:

- int x = 10;

- Python is dynamically typed.
- C is **statically** typed.

THE UNIVERSITY OF MEMPHIS

6

Printing a Literal String

hello.py — PYTHON

```python
print("Hello, world!")
```

hello.c — C

```c
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

Python's `print()` needs no setup or header. C's `printf()` requires `#include <stdio.h>` and an explicit newline (`\n`).

THE UNIVERSITY OF MEMPHIS

7

Printing a String with Variables

greet.py — PYTHON

```python
name = "Ada"
age = 28

print("Hello,", name, "-", "you are", age,
"years old")

# or with an f-string
print(f"Hello, {name} - you are {age} years
old")
```

greet.c — C

```c
#include <stdio.h>

int main(void) {
    char name[] = "Ada";
    int age = 28;

    printf(
        "Hello, %s - you are %d years old\n",
        name, age
    );
    return 0;
}
```

Python interpolates types automatically. C's `printf` needs a format specifier per variable - `%s` for strings, `%d` for integers - matched by position.

THE UNIVERSITY OF MEMPHIS

8

Print at a Glance

ASPECT	PYTHON	C
Setup	None required	#include <stdio.h> header
Function	print()	printf()
Line ending	Newline added automatically	Newline (\n) added manually
Variable insertion	f-strings or comma args	Format specifiers (%s, %d, ...)
Type declarations	Not required (dynamic typing)	Required (static typing)

THE UNIVERSITY OF MEMPHIS

9

If Statements

Python:

```
if x > 5:
    print('Greater')
elif x == 5:
    print('Equal')
else:
    print('Less')
```

C:

```
if (x > 5)
{
    printf("Greater");
}
else if (x == 5)
{
    printf("Equal");
}
else {
    printf("Less");
}
```

10

Nested If Statements

Python:

```
if x > 0:
    if x < 10:
        print('Single digit')
```

C:

```
if (x > 0)
{
    if (x < 10)
    {
        printf("Single digit");
    }
}
```

11

For Loops

Python:

```
for i in range(5):
    print(i)
```

C:

```
for(int i=0; i<5; i++)
{
    printf("%d", i);
}
```

12

While Loops

Python:

```
i = 0
while i < 5:
    print(i)
    i += 1
```

C:

```
int i = 0;
while(i < 5) {
    printf("%d", i);
    i++;
}
```

13

Do-While Loop (C only)

C:

```
int i = 0;
do
{
    printf("%d", i);
    i++;
} while(i < 5);
```

- Python:
- (No direct equivalent, use while loop)

14

Functions

Python:

```
def add(a, b):
    return a + b
```

C:

```
int add(int a, int b)
{
    return a + b;
}
```

15

Data Structures

Python:
list = [1,2,3]

C:
int arr[3] = {1,2,3};

- Python has built-in high-level structures.

16

Modules vs Libraries

Python:
- import math
- from math import sqrt

C:
- #include <stdio.h>
- #include <math.h>

- Python uses modules/packages.
- C uses header files and libraries.

17

Summary

Python:
- Easier syntax
- Rapid development

C:
- More control
- Better performance

18