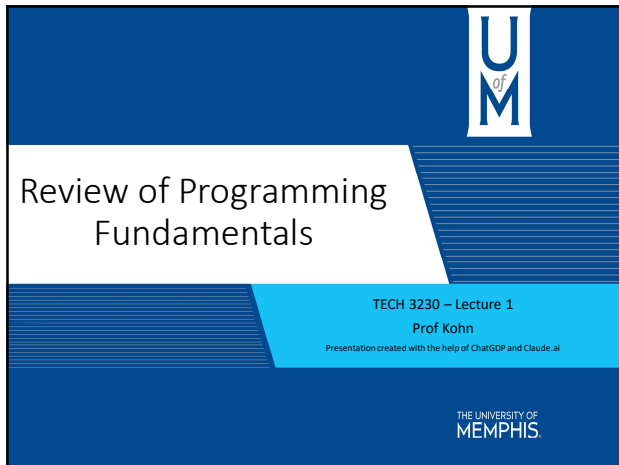
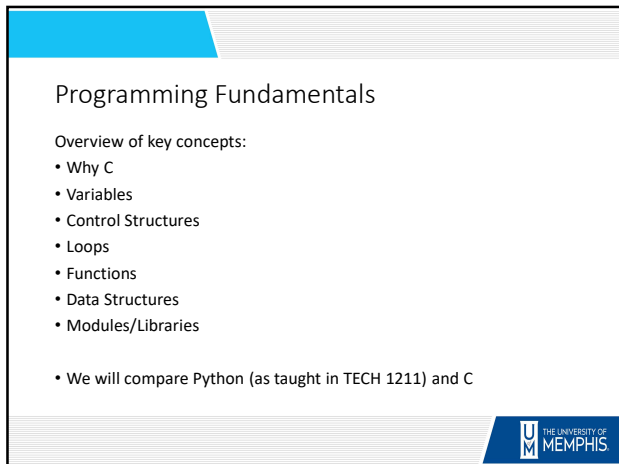
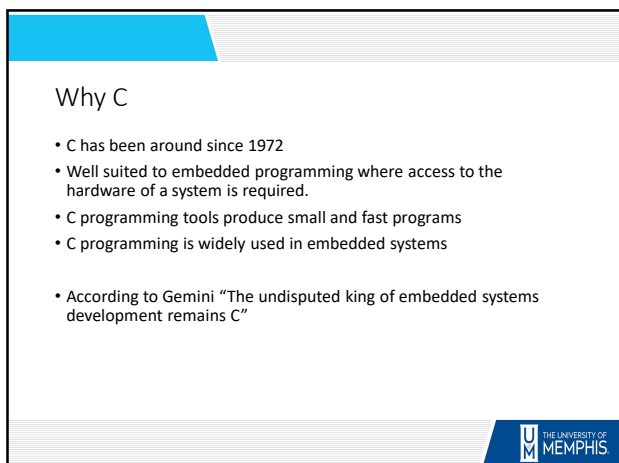




1



2



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")
5 if __name__ == "__main__":
6     main()
        
```

C

```

1 #include <stdio.h>
2 void greet(char *name) {
    printf("Hello, %s\n", name);
}
3 int main() {
4     greet("World");
    return 0;
}
        
```



4

How the Pieces Relate

The same four structural roles, expressed with different syntax

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



5

Variables

Python:

- `x = 10`

C:

- `int x = 10;`

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



6

Printing a Literal String

```
hello.py
print("Hello, world!")
```

```
hello.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

```
great.py
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")
```

```
great.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	<code>#include <stdio.h></code> header
Function	<code>print()</code>	<code>printf()</code>
Line ending	Newline added automatically	Newline (<code>\n</code>) added manually
Variable insertion	f-strings or comma args	Format specifiers (<code>%s</code> , <code>%d</code> , ...)
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
