JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

EN.540.635
Software Carpentry

Lecture 3
Introduction to Python
(Hello World, Variables, Conditionals, Loops, and Functions)

The First Computer Programmer

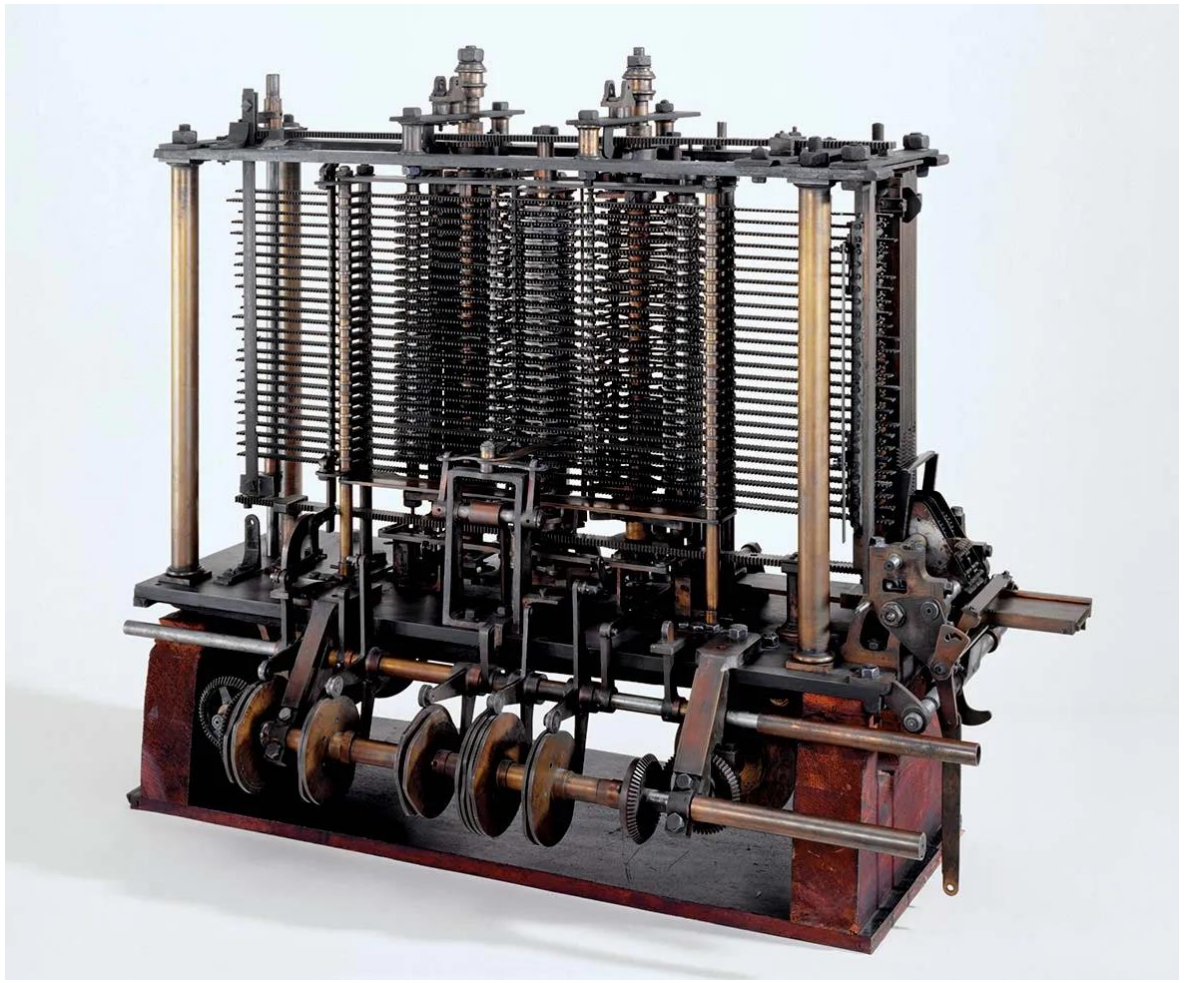


Image credits : Britannica

What is Programming?

The ability to define a set of instructions for a non-living device to carry out.

Examples:

- Joseph Marie Jacquard – Loom that weaves differently depending on input.
- RoboCup – Robot Soccer!
- Computer programs.
- Application Development (apps on phones/laptops).

Computer Programming


```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!\n");
    }
}
```

```
section     .text
global      _start                  ;must be declared for linker (ld)

_start:                             ;tell linker entry point

    mov     edx,len                 ;message length
    mov     ecx,msg                 ;message to write
    mov     ebx,1                   ;file descriptor (stdout)
    mov     eax,4                   ;system call number (sys_write)
    int     0x80                    ;call kernel

    mov     eax,1                   ;system call number (sys_exit)
    int     0x80                    ;call kernel

section     .data

msg     db  'Hello, world!',0xa     ;our dear string
len     equ $ - msg                 ;length of our dear string
```

Scripting Languages

High Level Languages

Low Level Languages

```python
print("Hello World!\n")
```

```c
#include <stdio.h>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

Computer Programming

Task : Add 2 + 3

In Binary

$2 \rightarrow 10$

$3 \rightarrow 11$

$5 \rightarrow 101$

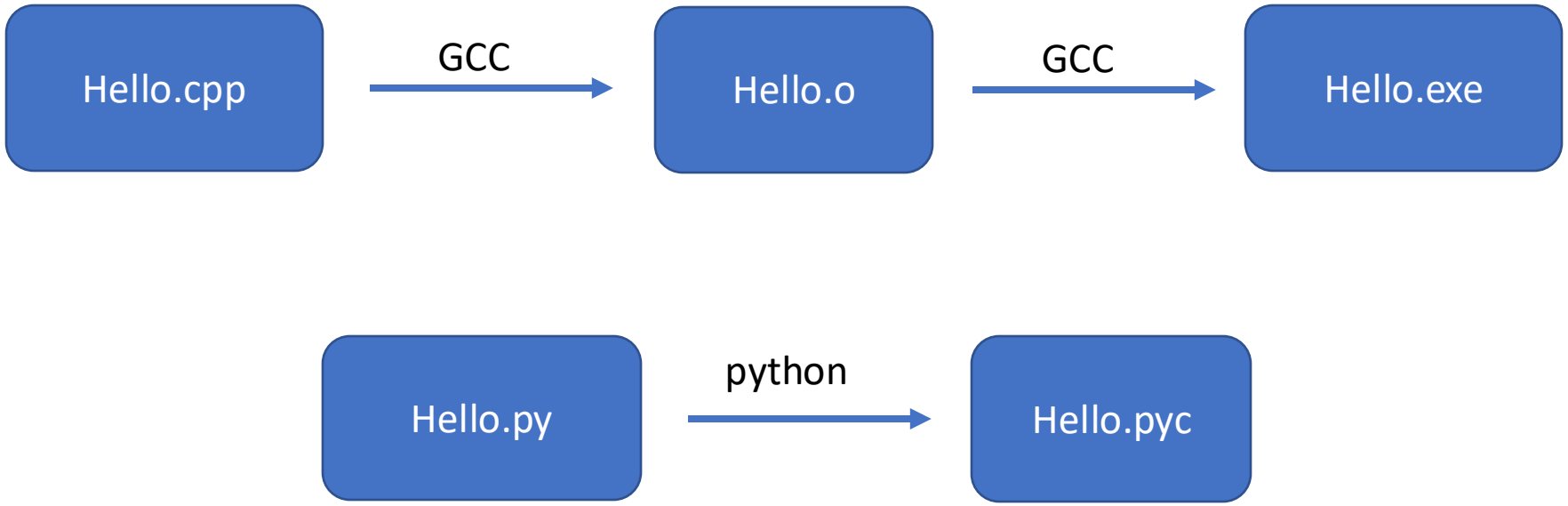
In Assembly

```
1 mov a,#45h
2 mov b,#86h
3 add a,b
4 mov 20h,a
5 end
```

In Python

```
>>> print(2 + 3)
5
>>>
```

How does Python behave differently from C++?



Hello World!

- Python hello world in terminal

```
print("Hello World")
```

Variables

- int – An integer (-2, 3, 0, 1230, …)
- float – A floating point number (3.234, -1.312, 68.342, …)

- str – A sequence of characters that we want to let the program know is text
  - “This is a string”
  - “3.233”

- Special cases of str (quotation marks):
  - “\”Hello World\”” will let you print Hello World with “ ”
  - ‘”Hello World”’ will do the same thing!

Variables

- Lists are arrays (think back to math, <x1, x2, x3, …, xn>)
- Lists can hold any kind of variable! This is a nice way of storing data!
- Quickly make ranges of data using the range function:

```
1  # Get a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2  a = range(0, 10)
3
```

- Note, in Python 3, range returns a ***generator***, not a ***list***. We can use typecasting to change the variable type.

Variables

- Booleans:
  - A bool holds one of two values: True or False
  - Casting to an integer, it holds: 1 or 0

- Booleans help us simplify logic using conditionals such as:
  - If …
  - If … else…
  - If … elif… else…
  - If … elif … elif … elif … (etc…) … else …

```
1  a = bool(True)
2  b = bool(False)
3
4  print a, int(a)
5  print b, int(b)
6
```

```
C:\Users\hherbol\Downloads>python demo.py
True 1
False 0

C:\Users\hherbol\Downloads>
```

Conditionals

- Joe and Jenn are in prison, trading letters in secret. Using a computer, they have access to functions like “send_letter()” and “is_guard_watching()”. How do they write their code to only send a message when the guard is not looking?

- Hint – You can invert a bool with the **not** keyword. That is:

```
1  if not is_guard_watching():
2      send_letter()
3
```

Loops

- What if we want to print the lyrics to “99 bottles of beer on the wall”?
  - We can print them out entirely.

    OR
  - We can be lazy (like good programmers) and print out using a loop!

- Loops:
  - While loops
  - For loops

While Loops


```
1  while some_condition:
2      ...
3      run code
4      ...
5
```

Examples of “some_condition”

- var1 < var2
- bool returned from a function (like “is_guard_watching”)

For Loops


```
1 for i in range(0, 100):
2     ...
3     run code
4     ...
5
```

range(0, 100) is a function that returns the list [0, 1, 2, 3, …, 98, 99]. NOTE! It is only lower-bound inclusive. That is, it gives the list [0, 100).

Examples of other "ranges"

- Lists (of any data type): ["a", "b", "c", "d"]
- Dictionaries (of any data type)

Loops

- Looping is vital to coding!
- Loops help remove tedious tasks from data manipulation.
- Loops let us do iterative algorithms.

- Fun case:
  - Joe and Jenn can't keep re-running code until the guard isn't watching, that would be suspicious! Let them enter the code into a loop instead!

While Loops

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

```python
import random


def is_guard_watching(rate=0.9999999):
    return random.random() < rate


def send_letter(msg="Default Message"):
    print(msg)


# Method 1 - Let the loop watch the guard, and when the
# guard is no longer watching, the loop ends, allowing us
# to send the letter
while is_guard_watching():
    continue
send_letter("Let's break out of here")

# Method 2 - Make an infinite loop (DANGEROUS... but common
# coding practice).  In the loop, check if the guard isn't
# watching.  If so, send the letter and break out of loop.
while True:
    if not is_guard_watching():
        send_letter("Watch out, our messages might be read")
        break

```

For Loops


```
1  import random
2
3
4  # Guards got lazy, rate decreased here
5  def is_guard_watching(rate=0.9999):
6      return random.random() < rate
7
8
9  def send_letter(msg="Default Message"):
10     print(msg)
11
12
13 # Method 3 - Jenn want's to only try sending the message 10000
14 # times.  She's worried the constant computing might give away
15 # their letter swapping.
16 for i in range(0, 10000):
17     if not is_guard_watching():
18         send_letter("The guard's are ugly.")
19         break
20
```

Functions

- A block of code that only runs when it is called.

- It can have input arguments that you specify and can also have an output result.

```
2  def f(x):
3      return x ** 2
4
5
6  f5 = f(5)
7  print(f5)
```