Zachary Shin

Unit 8: Programming Languages

CS 101, Fall 2020

Learning Objectives

After completing this unit, you should be able to:

- Explain the difference between an assembler, compiler, and interpreter.
- Name and describe at least six different programming languages.
- Compare and contrast imperative programming with object-oriented.
- Trace the execution of Python if/else statements and while loops.
- Determine the value of logic expressions (using `and`, `or`, `not`) in Python.
- Implement a Python function that computes a mathematical formula.

Textbook Sections

6.1 Historical Perspective
6.2 Traditional Programming Concepts
6.3 Procedural Units

Video Lectures

- More Python
- Grace Hopper on Letterman

Assignments

Exercise08 Programming Languages; Chapter 6 Problems
Lab08 Python: Procedural and Object-oriented

Exercise 8: Programming Languages

Model 1 Low-Level Languages

The following program, shown in three different languages, calculates the sum of numbers from 1 to 10. In other words, it adds $1 + 2 + \ldots + 10 = 55$.

Machine Code
(1st Generation)

```
0x000:
0x000: 700001000000000000

0x100:
0x100:
0x100: 30f00b00000000000000
0x10a: 30f30100000000000000
0x114: 30f10200000000000000
0x11e: 30f20100000000000000

0x128: 2017
0x12a: 6107
0x12c: 736401000000000000

0x135:
0x135: 6013
0x137: 6021

0x139: 2017
0x13b: 6107
0x13d: 743501000000000000

0x146:
0x146: 00
```

Y86-64 Assembly
(2nd Generation)

```
.pos 0 code
    jmp _start

.pos 0x100 code
_start:
    irmovq  $0xb,  %rax
    irmovq  $0x1,  %rbx
    irmovq  $0x2,  %rcx
    irmovq  $0x1,  %rdx

    rrmovq  %rcx,  %rdi
    subq    %rax,  %rdi
    je      done

loop:
    addq    %rcx,  %rbx
    addq    %rdx,  %rcx

    rrmovq  %rcx,  %rdi
    subq    %rax,  %rdi
    jne     loop

done:
    halt
```

Standard C
(3rd Generation)

```
int main()
{
    int upper = 11;
    int sum = 1;
    int val = 2;

    while (val < upper)
    {
        sum = sum + val;
        val++;
    }
}
```

Questions

1. Compare the length of each program. Do not count labels (e.g, `0x000:`, `.pos 0 code`) or punctuation (e.g., {, }).

a) How many instructions of machine code? 12

b) How many instructions of assembly code? 12

c) How many non-blank, non-brace lines of C code? 5

2. All data values for this program are stored in registers named `%rax`, `%rbx`, etc.

a) In which register is the sum stored? %rax

b) In which register is the next value to add stored? %rdi

3. The instruction `irmovq` means "move immediate value to register". Immediate values begin with a dollar sign ($), and registers begin with a percent sign (%).

a) What is the value 11 in assembly code? 0xb

b) Does assembly use decimal or hexadecimal? Hexadecimal

c) Does Standard C use decimal or hexadecimal? decimal

4. In terms of the machine, what does an assignment statement do? As part of your answer, name the instructions in Model 1 that perform assignment.

An assignment statement requests that a value be assigned to a variable. 30f, irmovq, int

5. Consider the line "`rrmovq %rcx, %rdi`". The instruction `rrmovq` means "move (copy) register to register".

a) What is stored in register `%rcx`? 0x2

b) Where is this value copied to? %rdi

6. The instruction `subq` means "subtract". Given two registers R and T, `subq` performs $R - T$ and stores the result in T.

a) What is stored in register `%rax`? 0xb or 11

b) In what case would `%rax - %rdi` be zero? if %rax & %rdi have same value

7. The instruction `je` means "jump if the last operation's result equals 0", and the instruction `jne` means "jump if the last operation's result does not equal 0". Circle the portion of assembly code that corresponds to the `while` loop in C.

Model 2 High-Level Languages

In addition to adding the numbers from 1 to 10, this program prints (displays) the result on the screen using Standard I/O.

Standard C
(3rd Generation)

```
#include <stdio.h>

int main()
{
    int upper = 11;
    int sum = 1;
    int val = 2;

    while (val < upper)
    {
        sum = sum + val;
        val++;
    }

    printf("Sum = %d\n", sum);
}
```

Python
(4th Generation)

```
upper = 11
isum = 1
val = 2

while val < upper:

    isum = isum + val
    val = val + 1


print("Sum = " + str(isum))
```

Questions

8. Compare the C code with the Python code.

   a) Circle the lines of C code that were not present in Model 1.

   b) Which lines of C are not present (i.e., needed) in Python? #include <stdio.h> int main( )

   c) What punctuation used in C is not required in Python? { ; }

9. Without using braces, how does Python know which lines are part of the `while` loop?

Python knows which lines are part of the while loop through indents.

10. Why does Python use the name `isum` instead of `sum`? Hint: type `sum` into a Python shell.

There is a built-in function of sum. in Python.

11. In Python, the `range` function can be used to generate a sequence of numbers. Use a Python shell to answer this question.

a) What is the result of `list(range(5))`? [0, 1, 2, 3, 4]

b) What is the result of `str(range(5))`? 'range(0, 5)'

c) What do the `list` and `str` functions do? list prints numbers up to 5. str prints the words in the range

d) What is the result of `sum(range(5))`? 10

e) What does the `sum` function do? sum adds all numbers together up to the max.

12. Rewrite the entire program of Model 2 using one line of Python code. Hint: you'll need to use `print`, `str`, `sum`, and `range`.

```
print(sum(range(11)))
```

13. Based on Model 1 and Model 2, what does it mean to be low-level vs high-level?

High-level displays the results using standard I/O when low-level does not.

Chapter 6: Programming Languages

Answer the following questions using the textbook, your individual notes, and the Internet.

1. What is the difference between an *assembler*, a *compiler*, and an *interpreter*?

An assembler converts mnemonic expressions into machine language instructions. A compiler made machine instructions into short sequences to stimulate the activity. An interpreter executed the instructions as they were translated in.

2. What is the difference between *declarative* statements, *imperative* statements, and *comments*? Why do programming languages need all three?

Declarative statements defined custom terminology used later in the program. Imperative statements describe steps in the underlying algorithm. Comments enhance the readability of a program by explaining its esoteric features in a more human-computible form. We might all these languages in a program.

3. Draw parentheses to show operator precedence. What is the value of number?

```
number = 1 + 2 * 3 - 4 / (5 + (6 * 7)) - 8 / 9
```

= 4

4. Rewrite the following instructions in Python using a single if-else statement.

```
     if (X = 5) goto 50
     goto 60
50   print the value of Z
     goto 100
60   print the value of Y
100  . . .
```

```
if X=5:
    print(Z), go to 100
else:
    print(Y)
```

5. Why is the "goto" statement no longer popular in high-level programming languages?

High-level programming have other statements like "if-else" that are easier than "goto."