Types

Adrien Protzel

Exercise 1: Summarize

1. A brief survey of types (such as integer, string, and boolean) and their applications in programming and mathematics, focusing on how they define variables, structures, and behaviors with each other.

Exercise 2: Demonstrate & Explain

1. Type inference is when a language already knows or assumes the type of a variable, see ML code example 1; in ML, almost everything is inferred to be some type, making code smaller, cleaner, and easier to read.

# ML Code Example 1

X = 10;

Y = 1.2;

Z = “hello”;

In ML, the day datatype is inferred to be an enumeration to be used in the isweekend function. The function is type inferred based on the operations inside of it, in this case it would be a bool. This is because isweekend receives and checks d with Sa and Su which are type day, which in turn inferred d to be type day. This then results in a bool of d being something, see ML code example 2.

# ML Code Example 2

datatype day = M | Tu | W | Th | F | Sa | Su;

fun isWeekend d = (d = Sa) orelse (d = Su);

Exercise 3: Investigate

1. Type hints are a way to make typer inference more readable. They label what the variable and functions should be.

def add(a: int, b: int) -> int:

return a + b

def string(s: str) -> str:

return f"{s}"

The example above shows how the function, add explicitly takes in two ints and returns an int, or takes in a string and returns a string. These hints help me as a coder to better keep track of what type a variable should be and what a function should be returning, this can be beneficial for debugging and keeping track of dynamic types throughout the program. The down side to these optional hints is that it takes up more space and time to write out. Personally, I do not make huge python programs that would warrant these hints and even if I do, python does rightfully infer most of the time. However, knowing about them is definitely better than not.