CSTWPY (Winter 2022)

Python for Beginners

An Intro and Variables

1 Introduction to Python & Lesson Structure

Python has gradually gained high popularity among programmers over the years, making it the preferred language for beginners and machine language. Throughout these lessons, you will learn about the basics of Python. For consistency sake, each lesson will follow the same or very similar format, as outlined below:

1. Explanation of concepts.
2. Examples to further strengthen your understanding in the concept.
3. Practice exercises with solutions to ensure you understand the concept more deeply.

2 Learning Objectives

By the end of this week, students should be able to:

1. Understand the general syntax of Python
2. Define the following terms: **variables**, **strongly-typed** and **weakly-typed** languages, **dynamically** and **statically-typed** languages.
3. Understand the downfalls and benefits of **Dynamically-typed** languages
4. Declare and **re-use variables**.
5. Identify errors involving variables: including `NameErrors`.
6. Understand that operators act as functions.

3 This Lesson: **Variables**

Definition 3.1 (What are variables?)

Variables reference specific parts of memory where content is stored. The name ***variable*** implies that these objects may change depending on the type itself.

In either case, variable information is **checked** when the program gets executed by the user.

Now to understand how Python computes the variables, i.e., where does Python exactly check these types? Is it while the user is writing code or when the user runs the code. For that to be more clear, we need to discuss **typed languages**. See **Definition 3.2** for a more precise definition.

Definition 3.2 (Typed Languages)

Ways programming languages are typed affect their general syntax and how the program can run. Here are the **four** common types of programming languages:

1. **Strongly-typed languages** are defined by "fixed" types. This means at compile time, types of variables are already defined and **cannot** change.
2. **Weakly-typed languages** can have the ability to **coerce** types at runtime. For example, in javascript, this is completely valid code: `"1" + 1`. This is because at runtime, the `"1"` is type casted into an integer, then the addition is performed, which is what we call **type coercion**.
3. **Dynamically-typed languages** are languages where types are not checked until runtime (i.e., when the program is ran.)
4. **Statically-typed languages** are languages that check the type information when the program is compiled, not when the program is run.

Note: A weakly-typed language cannot be strongly typed and a **dynamically-typed** language cannot be **statically-typed**.

You can tell that Python is not statically typed from the examples below since it is an **interpreted** language. This means the program complies when the user calls it, not when the user ***compiles*** the code. Python is fixed types because it checks type information at runtime, and there are ***fixed types***.

Unlike **statically-typed** languages, dynamically-typed languages allow more flexibility in terms of changing variable types more freely. This means, a variable can represent two things; an integer and a float. **Concept 3.1** shows how you can declare variables in Python:

Concept 3.1 (Declaring Variables in Python)

```
# Declaring an Int
custom_int = 0

# Declaring a String
custom_str = ''          # this indicates an empty string

# In Python, we still use floats.
# Declaring a float
custom_float = 0.40
```

Example 3.1 demonstrates the dynamically-typing nature of Python, specifically how variables can be changed:

Example 3.1 (Demonstration of the Assignment of Different Variable Types in Loosely Typed Languages)

Suppose the following is entered into a python console.

```
>>> x = 5
>>> y = 'hello'
>>> x = y
>>> x
'hello'
```

3.1 The Downfall of Python & Java comparison

Though **Example 3.1** demonstrates that Python variables can be reassigned, there is one major downfall with Python's dynamic type. Though we haven't learned functions, **you do not need** to understand **Example 3.2**.

Example 3.2 (The Downfall of Python)

Suppose you have the following variables defined. Notice here that the variable `x` is ***redefined*** to contain a different type.

The following code snippet uses material to be taught later. You do not need to understand this.

```
x = 4
x = [3+4] + [34]
y = x + [True]
x = [x[i] for i in range(len(x))]

def foo(x: str) -> str:
    return x + "4"
```

Interestingly, since `type(x) == list`, calling the function with `x` as a parameter would fail because you cannot add a list to a string. This behaviour is to be expected in all dynamically typed languages, as the variable name can be reassigned any time. The only time the type of the variable name is checked is at runtime.

This is in direct contrast to a language such as Java, which checks the variable type at compile time and enforces a type structure. Essentially, when a variable gets assigned a type, the variable must stay that type unless you assign another variable name. For example:

Example 3.3 (Java Example)

```
String x = ""; // only can be a string
int y = 223; // likewise, but an integer
```

3.2 Undefined variable names

Most programming languages do not let developers to retrieve a variable that has not been defined. That is, the user must declare all variable names before their retrieval.

Example 3.4 (Undefined Variable Example)

The following code snippet uses material to be taught later. You do not need to understand this.

```
x = 4
x = [3+4] + [34]
y = x + [True]
x = [x[i] for i in range(len(x))]

def foo(x: str) -> str:
    return x + "4"

y = foo("3")
x = 2

# here we are trying to reference an undefined variable
z
```

In the case of **Example 3.4**, running it through a Python interpreter will yield a `NameError` because `z` is not defined.

Note that there are many other exceptions Python can output, some of which are user created.

Congratulations!

You are now one step closer to becoming an expert in Python!