Introduction to Python – Day 2

Author: Srishti Sawla
LinkedIn URL: https://www.linkedin.com/in/srishtisawla/

Names and namespaces

Say you are looking for a book, so you go to the library and ask someone for the book you want to fetch. They tell you something like *Second Floor, Section X, Row Three*. So, you go up the stairs, look for Section X, and so on.
It would be very different to enter a library where all the books are piled together in random order in one big room. No floors, no sections, no rows, no order. Fetching a book would be extremely hard. When we write code, we have the same issue: we must try and organize it so that it will be easy for someone who has no prior knowledge about it to find what they're looking for. When software is structured correctly, it also promotes code reuse. On the other hand, disorganized software is more likely to expose scattered pieces of duplicated logic.
First, let's start with the book. We refer to a book by its title and in Python lingo, that would be a name. Python names are the closest abstraction to what other languages call variables. Names basically refer to objects and are introduced by name-binding operations. Let's make a quick example (notice that anything that follows a # is a comment):

```
n = 3 # integer number
address = "221b Baker Street, NW1 6XE, London" # Sherlock Holmes' address
```

We defined variable in the preceding line

- An integer number n (type: int, value: 3)
- A string address (type: str, value: Sherlock Holmes' address)

So, what are n and address? They are names. Names that we can use to retrieve data within our code. They need to be kept somewhere so that whenever we need to retrieve those objects, we can use their names to fetch them. We need some space to hold them, hence: namespaces!

A namespace is therefore a mapping from names to objects. For now, just keep in mind that namespaces are places where names are associated with objects.

Scopes

There are four different scopes in Python.

- **The local scope**, which is the innermost one and contains the local names.
- **The enclosing scope**, that is, the scope of any enclosing function. It contains non-local names and non-global names.
- **The global scope** contains the global names.
- **The built-in scope** contains the built-in names. Python comes with a set of functions that you can use in an off-the-shelf fashion, such as print, all, abs, and so on. They live in the built-in scope.

The order in which the namespaces are scanned when looking for a name is therefore: local, enclosing, global, built-in (LEGB).

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module, or other object.

1. An identifier starts with
   - a letter A to Z or
   - a to z or
   - an underscore (_)
   - followed by zero or more letters, underscores, and digits (0 to 9).
2. Python doesn't allow keywords as identifiers
3. Python does not allow punctuation characters such as @, $, and % within identifiers.
4. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

1. Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
2. Don't use variables with same name and different case. For example, don't use 'Age' and 'age' in the same python file. It may cause confusion.

Different Case Types allowed in Python

1. camelCase

Camel Case is inspired from animal "Camel". Where first word will be small letters and from second word, first character will be capitalized like camelCase.
Some of the big companies also inspired from this. Example like iPhone, eBay
Generally, Camel Case is used for Variable Naming

2. snake_case

Snake Case is naming with words separated by _ (underscore) and all small letters
Generally, Snake Case is used for Variable Naming
Ex: - snake_case, new_word

3. PascalCase

Pascal Case is naming with First letter of each word is Capitalized.
Generally, Pascal Case is used for Class Naming
Ex: - NewWord, PascalCase

Keywords (Reserved Words)

The following list shows the Python keywords. These are reserved words, and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or

break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	Yield
True	False	None

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.
The triple quotes are used to span the string across multiple lines. For example,

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

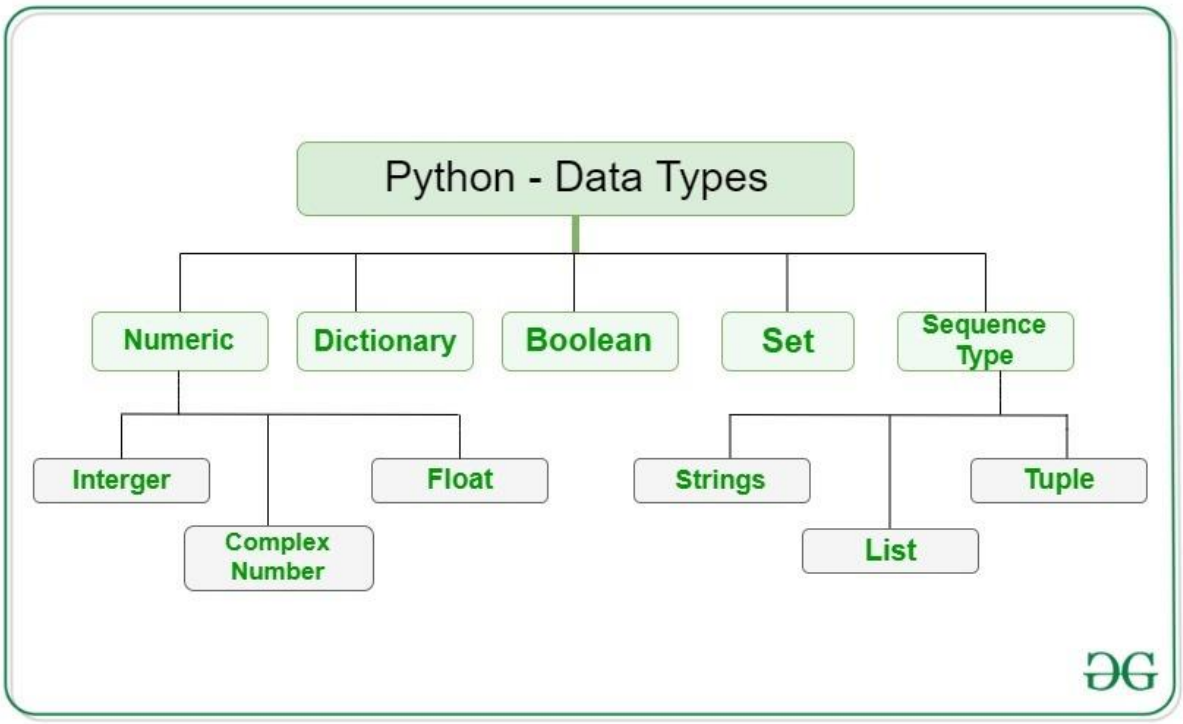
A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

```
print ("Hello, Python!") # Single line comment
```

Triple-quoted string can be used as a multiline comment:

```
'''
This is multiline
comment.
'''
```

Python Built-in Data Types




```
1  message = "Hi Computer Notes"
2  num = 5
3  pi = 3.14159
4  cmp = 2+5j
5
6  print(message, "is of type", type(message))  # This will return a string
7  print(num, "is of type", type(num))  # This will return an integer
8  print(pi, "is of type", type(pi))  # This will return a float
9  print(cmp, "is complex number?", isinstance(1+2j,complex))
```

```
Hi Computer Notes is of type <class 'str'>
5 is of type <class 'int'>
3.14159 is of type <class 'float'>
(2+5j) is complex number? True
```