Foundations of Python for AI: Revisiting Data Types, Conversions, and Operators Through a Case Study

Minor In AI, IIT Ropar

19th March, 2025

Welcome to the wonderful world of Python and Artificial Intelligence!

This module is designed for beginners, aiming to gently introduce you to the foundational concepts of Python programming that are crucial for your journey into AI. We'll explore data types, conversions, and operators, not just with abstract definitions, but through an engaging case study that will make learning fun and intuitive.

1 The Alien Message

Imagine this: An alien is trying to communicate with us. Not with advanced technology or complex equations, but with a series of seemingly random numbers: **73, 80, 76.**

What could these numbers possibly mean? Is it a secret code? A countdown to something? Maybe it's directions to their home planet!

This, my friend, is our first AI challenge. We will use Python to "decode" this message, revealing the hidden meaning and, along the way, solidify our understanding of Python's basic building blocks.

Figure 1: "Random fact: Gnorts, Mr. Alien" backwards is "Neil Armstrong"!!

2 Understanding Data Types

Before we crack the alien's code, we need to understand how Python organizes information. These are called **data types**. Think of them as different containers for storing different kinds of things.

2.1 Integers (int)

Integers are whole numbers, like -2, 0, 73, 80, or 76. They're perfect for representing counts, quantities, or any value that doesn't require decimals.

In Python, we can assign these numbers to variables:

```
m1 = 73
m2 = 80
m3 = 76
```

Here, `m1`, `m2`, and `m3` are variables holding integer values.

2.2 Booleans (bool)

Booleans represent truth values: `True` or `False`. They're essential for decision-making in programs. Imagine asking a question: "Is the number greater than 50?" The answer will always be either True or False.

```
is_greater = m1 > 50  # is_greater will be True
print(is_greater)
```

A fascinating thing about booleans is that Python understands how to work with them and numbers as well. Let's see what happens when we convert our numbers to booleans.

```
print(bool(73))
print(bool(0))
print(bool(-1))
```

All the numbers convert to *True* when converted to boolean, except for 0, which results in *False*.

2.3 Floats (float)

Floats are numbers with decimal points, like 3.14, -2.5, or 73.0. They're useful for representing measurements, percentages, or any value that requires precision.

2.4 Strings (str)

Strings are sequences of characters, enclosed in single quotes (' ') or double quotes (" "). They are used to represent text, names, sentences, or any combination of characters.

```
name = "Alien"
message = 'Hello, Earth!'
```

One peculiar thing about an empty string is that it acts as *False* when converted to boolean.

```
print(bool("alien"))
print(bool(""))
```

3 Type Conversion

Sometimes, we need to change the data type of a value. This is called **type conversion**. Python provides built-in functions for this.

- `int()`: Converts a value to an integer.
- `float()`: Converts a value to a float.

- `str()`: Converts a value to a string.
- `bool()`: Converts a value to a boolean.
- `chr()`: Converts an integer to its corresponding character (based on ASCII).

Back to our alien message! Let's see what happens if we convert our numbers to Booleans:

```
print(bool(m1)) # True
print(bool(m2)) # True
print(bool(m3)) # True
```

True, True, True... Doesn't seem to reveal much, does it? Let's try converting them to floats:

```
print(float(m1)) # 73.0
print(float(m2)) # 80.0
print(float(m3)) # 76.0
```

Still not helpful! This is where `chr()` comes in. `chr()` converts an integer to a corresponding character. There are codes (also known as ASCII codes) for every character that's used on the computer. Let's take a look at what `chr` does.

```
print(chr(m1))
print(chr(m2))
print(chr(m3))
```

The output will be:

Output

```
I
P
L
```

Aha! Suddenly, the alien message is clear. The numbers 73, 80, and 76 represent the letters I, P, and L. Could the aliens be trying to tell us something? Maybe they are into *IPL*.

4 Operators

Operators are symbols that perform operations on values. They are essential for manipulating data and performing calculations.

4.1 Arithmetic Operators

These are the basic operators for performing mathematical calculations:

- `+` (Addition)
- `-` (Subtraction)
- `*` (Multiplication)
- `/` (Division)
- `//` (Floor Division - returns the integer part of the division)
- `**` (Exponentiation - raises a number to a power)
- `%` (Modulo - returns the remainder of a division)

```python
a = 10
b = 3

print(a + b)    # 13
print(a / b)    # 3.3333333333333335
print(a // b)   # 3
print(a ** b)   # 1000
```

4.2 String Concatenation

The + operator can also be used to join strings together. This is called string concatenation.

```python
str1 = "Hello"
str2 = "Alien"
combined_string = str1 + " " + str2  # "Hello Alien"
print(combined_string)
```

One thing to remember here is that you cannot add a string and a number directly, and it can result in an error if you're not careful. The other way of concatenating a string is as follows:

```python
num = 2024
print("Alien " + str(num))
```

4.3 String Repetition

The * operator can be used to repeat a string multiple times.

```python
alien_repeat = "Alien" * 5  # "AlienAlienAlienAlienAlien"
print(alien_repeat)
```

4.4 The `join()` method

The *join()* method of the string class can be very powerful to concatenate iterables such as lists, tuples and sets. See the examples below.

```python
str_list = ["hello", "alien"]
print(" ".join(str_list))

str_tuple = ("hello", "alien")
print(" ".join(str_tuple))
```

5 Back to the Message - String Concatenation

Let's say we want to combine the IPL. How can we do that?

```python
m1 = chr(73)
m2 = chr(80)
m3 = chr(76)

alien_message = m1 + m2 + m3
print(alien_message) # Output: IPL
```

The message is still IPL, but this time we can confirm the characters are concatenated.

6

Congratulations! You've taken the first steps in your Python for AI journey. We learned about data types, conversions, and operators, and used them to decode a secret message from an alien! This is just the beginning. As you continue your journey, you'll discover even more powerful tools and techniques for solving complex AI problems. Remember to have fun and stay curious. There's a whole universe of knowledge waiting to be explored!