Python Syntax and Dot Notation Reference

The basics

Python is known for its elegant and readable syntax. If you're new to programming or transitioning from another language, you'll quickly appreciate Python's focus on clarity. Let's dive into the core elements that make up Python's structure.

1. Whitespace and Indentation

Many languages use {curly braces} to define blocks of code, Python relies on indentation. Consistent indentation is not just for visual neatness; it tells the Python interpreter how your code is organized. A common standard is to use four spaces for each level of indentation.

```
if score >= 80:
    print("You passed!")
else:
    print("Please try again.")
```

2. Variables

Variables are named containers that hold values. Again, Python does things a little differently than many languages in the fact that you don't need to declare data types in Python.

```
name = "Alice"  # String
age = 25        # Integer
is_new_student = True  # Boolean
grade_average = 89.5  # Float
```

3. Comments

Comments are useful for communicating to yourself or others who may be reading your code what certain code blocks or functions can do. It can be especially useful when returning to a project after a break or when having someone review your code before deploying.

Comments are lines of text within your code that the Python interpreter ignores. You can comment a single line with a # or for multi-line comments, enclose your text within triple quotes.

4. Lists

Lists are used to store sequences of items. They are mutable, meaning you can change their contents after creation.

You'll need to use square brackets and make sure elements in the list are comma-separated.

```
shopping_list = ["milk", "eggs", "bread"]
```

Lists are zero indexed which means that if you want to access something that's stored in a list you need to start with 0 rather than 1.

```
print(my_numbers[0])   # Output: 1 (first element)
print(shopping_list[2])  # Output: bread
```

There are a number of common operations you can use with lists but these are by no means comprehensive.

Common Operations:

- `list.append(item)` - Adds an item to the end.
- `list.insert(index, item)` - Inserts an item at a specific index.
- `list.remove(item)` - Removes the first occurrence of an item.

5. Dictionaries

Dictionaries store unordered collections of data in key-value pairs. Like lists, the dictionary itself is changeable or mutable but the keys used to access the values inside the dictionary must be unique and are not able to be changed (immutable).

To create a dictionary you need to use curly braces {}. The one thing you should be careful with is to not confuse using colons which separate keys from values, and commas which separate pairs.

```
student_info = {"name": "Bob", "age": 22, "major": "Computer Science"}
```

Square brackets are used to access something inside of another variable or object. In this case you're accessing the name key inside of the student info variable. In later videos and activities you will use square brackets to access things inside of arrays or dictionaries like dataframes in this same way.

```
print(student_info["name"])  # Output: Bob
```

Common Operations you can use with dictionaries include but are by no means limited to the following:

- `dict[new_key] = new_value` - Adds a new key-value pair or updates an existing one.
- `del dict[key]` - Removes the key-value pair with the specified key.

Go to next item ✓ Completed

Like Dislike Report an issue

Go to next item →