ComIT

Talking to Python: Basic Syntax

The Python Interpreter

- We know that a computer program is essentially just a series of commands sent to the computer.
- We also know that computers only speak in machine code, 1’s and 0’s.
- We need a translator to take human readable commands and convert them into something the computer can understand.
- The Python interpreter is the “interlocutor” that we will be talking to the computer through.

The Python Interpreter

- The Python interpreter can be used in two ways:
    - Interactive mode: Python runs as an interactive program in a command line shell
    - Using it to execute a Python file:
        - Python file types usually have a “.py” extension
        - Python files are just text files populated with a sequence of valid Python statements

Python Syntax

- In order to provide Python with instructions, we must conform to the rules Python defines when forming commands.
  - These are quite strict in comparison to human writing.
  - Any “grammatical” errors in our commands will result in errors in our programs.
- The rules when writing Python commands are called the **syntax**
  - Syntax is defined as the “pattern used for of formation of sentences or phrases in a language.” – Some Dictionary somewhere

Python Syntax

1. Python commands are separated by newline characters. In other words, each new line in Python is a new command.
2. It is sometimes required to group commands together as a single executable unit. This is called a “block” of code, and blocks of code are defined using indentation.
   - You can use any number of tabs or spaces to indent code, but you must be consistent throughout your program. **4 spaces is standard though.**

Python Syntax - Variables

- Python programs solve problems by manipulating data.
- The most basic and important instruction in Python is defining data.
- Data in Python is represented by something called a **variable**
- A variable is created when it is assigned data, and the syntax looks like this:

<Variable Name> = <Data To Save>

Example: X = 5

Python Syntax - Variables

- A variable name is and sequence of characters that do not violate any of the naming rules in Python
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Data In Python

- Python handles different “types” of data, and there are several types of data that are built in.
- Different types of data can “do” different things. Numbers can be used to perform math, text can be printed, etc.
- It’s also possible to create new types, but we will cover that later.
- Any unit of data of any type can be saved in any variable.

Python Data Types

- Basic built in data types that Python supports are the following:
  1. Numerical: Integers or decimal point numbers
  2. Text: Called strings, represented by characters enclosed in quotation marks.
  3. Sequence Types: Multiple units of data can be stored in a single variable. Sequences are when the values are found based on their position.
  4. Mapping Types: Similar to a sequence data type, but units of data are found based on a known “key”. Keys are usually strings, but can be most data types.
  5. Boolean Types: “True” or “False” values

Python Syntax - Variables

- Data saved to a variable can come from different sources
- The most simple data source is what we call a “literal”
  - A “literal” is a direct representation of data in Python.
  - Example: x = “hello world”
- A more complex example would be the result of an expression.
  - An expression is a sequence of operands and operators
  - Example: 5 + 4
  - Different data types support different operators.

Python Syntax - Math

- Python can be used to perform mathematical operations
- The operators +, -, * and / can be used to perform arithmetic; parentheses (()) can be used for grouping. The ** operator can be used to calculate powers.

Examples:

```
(5 + 4) / 2
3 ** 4
5 * (4 + 3) / 2 ** 8
```

Python Syntax - Math

- Results of expressions can be saved to variables
- Expressions that return numbers can be nested in other expressions
  - A variable is actually just a really simple expression

Example:

```
width = 400
height = 800
area = width * height
```

Python Interpreter - Exercise

- Now that we understand how variables are defined, and how mathematical expressions are formed, we can use Python as a basic calculator
- To test this out, do the following:
  1. Open the Python interpreter through the command line terminal.
  2. Enter a valid mathematical expression.
- How did you tell Python to execute your command? What was the output? How did the interpreter behave? Did you manage to produce any errors? If so, how?

Python Syntax - Text

- You can also use Python to manipulate text.
- Text is represented in Python by a data type known as a “string”.
- A string is any sequence of characters wrapped in quotation marks
- Double or single quotation marks are valid
- You can use the “+” to concatenate strings together
- Example:

```
Hello_world = “Hello”+ “World!”
Single_quotes = ‘This is also valid!’
```

Python Syntax - Text

- Python has the ability to print text to the console as well as reading in user input.
- To do this we use special statements known as “functions”
  - We will provide a more detailed explanation to what functions are soon
  - For now, we will talk about functions as Python statements that perform an operation. Some functions require that you supply data, and some also return data that can be saved in a variable.

Python Syntax – Text Output

- To print text to the screen we use this statement:

```
print(“Hello World!”)
```

- In this example, “Hello World” is a string literal that is provided to the function as the data we would like printed.
- The function can accept almost any data type.
- We can pass in variables, or even other function executions that return data.

Python Syntax – Text Input

- Text input is similar to text output. We use a function called “input” to receive text from the user.
- The “input” function accepts a string as it’s argument, and it returns a string.
- Example:

```
text_from_user = input(“Enter some text:  ”)
```

- Try it out in the Python interpreter!

Python Syntax – Composite Data Types

- When writing Python code, it’s often necessary to store more than one unit of data in a single variable.
- These are “composite data types”, and for now we will focus on the syntax for two of the most common built in data types: Lists and Dictionaries

Python Syntax - Lists

- A list is created by using square brackets like this:

example_list = [ ]

- The above example is an empty list. A list can be created with data by writing literals, variables, or expressions in the square brackets and by separating them with commas:

populated_list = [“Hello”, “world!”]

- The above list contains two elements; the string literals “Hello” and “world!”

Python Syntax - Lists

- Individual elements in lists can be accessed by using square brackets in combination with the index of the item we want to access.
- Each element in a list is given an index that matches it’s location in the list
- The first index is always 0, and the index is incremented by one for each element in the list.
- To access the second element in the list, use this syntax:

  second_item = populated_list**[1]**

- Using this logic, how would you save an item to a spot in a list?

Python Syntax - Dictionaries

- A dictionary is created by using curly brackets like this:

```
example_dict = { }
```

- The above example is an empty dictionary. A dictionary can be created with data by writing literals, variables, or expressions in the curly brackets.
- Each entry in a dictionary needs to be paired with a “key” at will be used to access that item.
- A populated dictionary creation statement would look like this:

```
numbers = { “four”: 4, “three”: 3, “two”: 2 }
```

Python Syntax - Dictionaries

- Individual elements in dictionaries can be accessed by using square brackets in combination with the key of the item we want to access.
- Each element in a list is given an index that matches it’s location in the list
- To access a specific element in the list, use this syntax:

four = numbers**[“four”]**

- Using this logic, how would you save an item to a spot in a dictionary?

Group Exercise

Exercise

- Create a Python program that prints the cost of an item.
- When the program runs, it waits for a user to enter the name of an item. If the price of the item is known, print the price of the item to the screen.
- Example of execution output:

```
python menu.py
Enter the name of the item you would like to know the price of: hamburger
The price of a hamburger is $9.99
```

Questions?