Python Fundamentals

Editors, Syntax, Data Types, Variables, Operators, & Expressions

Objectives:

- What is a text/code editor?
- What is language syntax and semantics?
- What is a data type?
- What is a variable?
- What are operators & expressions?

What is a text/code editor

- A text/code editor is a tool used to compose (write) code (similar to a word processing program such as Microsoft word)

- There are many on the market. Sublime, Atom, Brackets, Visual Studio Code, and much, much more.

What is a text/code editor

- Some not only assist with writing code, but also take the code you have written and execute the necessary commands to run your code as well.

- Python can install a text editor for you, called IDLE. IDLE stands for **I**ntegrated **D**evelopment and **L**earning **E**nvironment.

- IDLE allows you to easily write and execute Python code.

What is syntax? What are semantics?

- Syntax is the grammatical rules and structural patterns governing the ordered use of appropriate words and symbols for issuing commands, writing code, etc., in a particular software application or programming language.

- Semantics is about the *meaning* of the words. In the context of a programming language, do the words form a coherent set of instructions?

What is syntax? What are semantics?

- When writing code you need to ensure that both the **syntax** is correct, and that its **semantic** meaning makes sense.

- Python will often spot issues related to syntax or semantics while attempting to interpret your source code.

- Basically, it's important to pay very close attention to what you write!

What is a data type?

- Firstly what is Data?
- Data can be:
  - A number e.g. 1,2,3
  - A person's name e.g. "Fred"
  - A single letter/character e.g. 'a', 'b'
  - A value of true or false
- In programming not only do we store data, but we also perform operations on data. For example imagine a program that calculates the average age of students

What is a data type?

For Example:

Name: Bob, **Age**: 23

Name: Sam, **Age**: 18

Name: Sue, **Age**: 21

Name: Andrew, **Age**: 30

```
average = (23 + 18 + 21 + 30) / 4 = 23
```

What is a data type?

This makes sense, but it would not make sense to try and calculate the average name.

For Example:

```
Bob + Sam + Sue + Andrew / 4 = ???????
```

This is because a person's name is a different type of data than their age.

What is a data type?

What does this mean? The operations we can perform with data depend on their type.

Python does not require you to declare the *data type* of a value; it will determine it at run-time (ie: interpreter). However, if you perform an operation that is invalid for that type of data, Python will generate an error.

In this sense, Python is **strongly and dynamically typed**.

What is a data type?

Data types in Python tend to fall into the following three categories:

- Numeric types.
- Text sequence types.
- Boolean values. True or False

While there are more, these are the basic ones.

What is a data type?

Below are listed the most common numeric types

- **int**. An integer is a whole number with no decimal portion. an_interger = 10

- **float**. A floating point number is a number with a decimal portion. a_float = 10.5

Numeric types have no practical size limit in Python 3.

What is a data type?

Below is listed the *only* text sequence type in Python.

- **String**
- **str**. An immutable sequence of Unicode characters. Used for storing a single character, words, sentences, or even pages of text. a_string = “hello”

What’s Unicode? It’s an international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs.

What is a data type?

```
10 # An int value

"The quick brown fox" # A str value

2.50 # A float value

True # A Boolean value
```

Note how we surround our **str** values with double quotes. In addition, these are known as *literal* values. In that we've literally written a value.

What is a variable?

Often we'll have some data we want to store, so we can refer back to it later in our code. The result of a calculation, for example. Or storing a literal value.

We call this a variable. It's also implicit in the name that the stored data may change (vary) throughout the life of the program.

Variables are in effect the same as algebra e.g. `x = 2, y = 5, z = x + y (z = 7)`

What is a variable?

```
number = 10 # An int variable

word = "The quick brown fox" # A str variable

price = 2.50 # A float variable

is_valid = True # A Boolean variable
```

Above we are declaring four different variables. Note the type of each variable does not need to be declared.

Rules for Variable

Rules for variable names

- No spaces
- Start with a letter
- Cannot use reserve words: True, False, class, if …
- Variable name is **case sensitive!**

It is recommended to use **snake_case** for variable name

- Letters in a word are written in lower case
- Each new word is separated by an _ (underscore)

What is an operator?

- Now that we have a variable storing some data, we probably want to do something with it. This is where operators come in.

- An example might be some addition, like 5 + 10. Or multiplying some values, then subtracting an amount from the result of our multiplication.

What is an operator?

- In fact, 5 and 10 are what we call **operands**. Components used in conjunction with an **operator**. An **operator** that uses one operand is called a **unary operator**. An operator that uses two is called a **binary operator**.

- For now we're going to focus on a few simple types of operators. Some **binary operators** to do basic arithmetic, and some **unary operators** to increment/decrement values.

What is an operator?

Arithmetic operators are the most common type in Python. They fall within the category of binary operators.

+ (addition) e.g. `5 + 5 # result: 10`

- (subtraction) e.g. `3 – 6 # result: -3`

* (multiplication) e.g. `3 * 6 # result: 18`

/ (division) e.g. `4 / 6 # result: 0.6666666666666666`

// (floor division) e.g. `4 / 6 # result: 0`

```
-11 // 3 = -4 # rounded away from 0 for negative number
```

% (modulo) e.g. `5 % 2 # result: 1`

What is an operator?

- Arithmetic operators should be used with numeric types. This makes sense, given we wouldn't want to multiply two words. But what happens when we add two strings together?

e.g. `result = "hello " + "world"`

What is an operator?

- This is called **string concatenation**. The result of this expression would be a string with the value `"hello world"`

- In essence, we combine the contents of two strings and generate a new string with the combined (concatenated) contents.

What is an operator?

- The most common **unary operators** used in Python are the minus and plus operators. Below are some examples.

- - (minus)

e.g. `x = 3`, `x = -x` , x now equals -3

- + (plus)

e.g. `x = -3`, `x = +x` , x still equals -3 (????)

What is an operator?

- You might be wondering what's happening in the last slide. Why does –x change the value to a negative, but +x doesn't change a negative to a positive?

- The plus operator is essentially a **no-op**. A **no-op** is something in your code that effectively does nothing.

What is an expression?

- An expression is a sequence of one or more operands and zero or more operators that can be evaluated to a single value

- What does this mean? That any operand, and combination of operators, can be combined to build an expression. An expression will always return a single value when all of it has been evaluated.

What is an expression?

- For example, 1 + 1 is an expression because it uses an **arithmetic operator** (addition), **two operands** (1 and 1), and will return a single value of **2**

- Another example could be the literal value 5. Why? It has **one operand** (5) and will return a single value of **5**

- While these are simple examples, expressions are one of the fundamental parts of coding.

What is an expression?

```
x = 5
y = 7
w = 5

z = x + y - w
# the variable z now has a value of 7

z = x - y * w
# the variable z now has a value of -30
```

Note that programming languages adhere to the same order of operations as mathematics. It would be a bit worrying if they didn't!

List of Operator

Operator	Description	Example
+, -	Addition, subtraction	10 + 5 = 15 10 - 5 = 5
*, /	Multiplication, division	10 * 5 = 50 10 / 5 = 2.0
//	Floor Division – Divides and remove digit after decimal point	11 // 5 = 2 3 // 5 = 0 -11 // 3 = -4 # rounded away from 0
%	Modulus - Divides and return remainder	11 % 5 = 1 13 % 5 = 3
**	Exponent	2 ** 6 = 64
==, !=	Equal, not equal	(1 == 2) is NOT true (1 != 2) is true
<, <=	Less than, less than or equal to	(1 <= 2) is true
>, >=	Greater than, greater than or equal to	(1 >= 2) is NOT true

List of Operator Cont.

Operator	Description	Example
+=	Add right operand to the left operand	c += a is equivalent to c = c + a
and	Logical AND	Only (True and Ture) is true
or	Logical OR	Only (False or False) is false
not	Logical NOT	(not False) is true
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise	x in y is true, if x is a member of sequence y

References

Introduction to Python tutorial

https://docs.python.org/tutorial/introduction.html

Demonstration:

- What is a text/code editor?
- What is language syntax and semantics?
- What is a data type?
- What is a variable?
- What are operators & expressions?