



Python Class Notes

Clarusway



Introduction, Data Types

Nice to have VSCode Extentions:

- Jupyter

Needs

- Python (for Windows OS: add to the path while installing!)

Summary

- Backend Roadmap
- Setting up the environment
- Introduction to Python
- Built-in functions
- Modules - Standard Modules (Optional)
- Python syntax
- PEP8 Conventions
- Variables
 - Naming variables
- Data Types
 - String
 - Indexes
 - String Methods
 - Numeric Types
 - Integer

- Using Python as a calculator
 - Float
- Boolean
 - Comparison operators
- Type Conversion
- Concatnation
- Logical Operators











Setting up the environment

We will use `.ipynb` file extention in VSCode. This is beter to see the code. To set up the environment you can visit [Jupyter Notebooks in VS Code](#) this link.

Introduction to Python

Python is a programming language that lets you work more quickly and integrate your systems more effectively. Here is [Python Website](#).

Python is the most widely used and popular programming language in the world.

TIOBE		About us News Coding Standards TIOBE Index Contact				
		Products Quality Models Markets		Schedule a demo		
Apr 2023	Apr 2022	Change	Programming Language		Ratings	Change
1	1			Python	14.51%	+0.59%
2	2			C	14.41%	+1.71%
3	3			Java	13.23%	+2.41%
4	4			C++	12.96%	+4.68%
5	5			C#	8.21%	+1.39%
6	6			Visual Basic	4.40%	-1.00%
7	7			JavaScript	2.10%	-0.31%
8	9	▲		SQL	1.68%	-0.61%
9	10	▲		PHP	1.36%	-0.28%
10	13	▲		Go	1.28%	+0.20%

Built-in functions

Built-in functions in Python are pre-defined functions that come with the Python language and can be used without requiring the user to define or create them. These functions are available in Python without the need for any additional modules or libraries. Some examples of built-in functions in Python include:

```
print() # prints output to the console
len() # returns the length of an object
range() # generates a sequence of numbers
type() # returns the type of an object
sum() # returns the sum of a sequence of numbers
input() # reads input from the user
str() # converts an object to a string
int() # converts a string or number to an integer
```

Built-in functions make it easy to perform common tasks in Python without having to write a lot of code from scratch. Here is the list of [Built-in functions in Python](#).

Modules - Standard Modules

Standard modules in Python are pre-built libraries that come with the Python programming language. These modules provide a wide range of functionalities that can be used in various types of applications. Some examples of standard modules in Python include:

- **math** provides mathematical functions like trigonometric, logarithmic, and exponential functions,
- **datetime** provides classes for working with dates and times,
- **random** provides functions for generating random numbers and making random choices.

```
# Examples:
import random

# Generate a random integer between 1 and 10 (inclusive)
random_int = random.randint(1, 10)
print(random_int)

# or;
from datetime import datetime

# Create a datetime object for the current date and time
now = datetime.now()
print(now)

# or;
import math

# using math module functions
x = math.sqrt(25)
print(x) # 5.0
```

Standard modules are included with every installation of Python, and they can be imported and used in any Python program. By using standard modules, developers can save time and effort by reusing existing code and functionality rather than building everything from scratch.

Python syntax

Some syntax:

- Comments in Python start with the hash `#` character.
- The equal sign `=` is used to assign a value to a variable.
- `\` can be used to escape.
- Other escape sequences are:
 - `\n` for new line,
 - `\t` for a tab,
 - `\b` for a backspace.
- Indentation can be either a tab or 4 spaces, but don't mix them.

PEP8 Conventions

Overall code syntax rules are stated at PEP8. There are online tools.

[PEP8 online check](#)

Variables

In Python, a variable is a named location in memory that is used to store data. Variables are used to store values that can be manipulated or referenced later in a program. Variables in Python are dynamically typed, which means that you don't need to specify the data type of a variable when you create it. The type of a variable is determined automatically based on the value that is assigned to it.

Here is an example of creating and using variables in Python:

```
x = 5
y = "hello"
z = True

print(x) # Output: 5
print(y) # Output: "hello"
print(z) # Output: True

x = x + 1
print(x) # Output: 6
```

In this code, we create three variables: "x", "y", and "z". "x" is assigned the integer value 5, "y" is assigned the string value "hello", and "z" is assigned the boolean value True. We can then print the values of these variables using the "print" statement.

We can also manipulate the value of a variable by assigning a new value to it. In this code, we increment the value of "x" by 1 and print the new value.

Variables in Python can store different types of data, including integers, floats, strings, booleans, and more complex data structures like lists and dictionaries. They are an important part of programming in Python and are used extensively in writing Python programs.

Naming variables

When naming variables in Python, it is important to choose names that are clear and descriptive, while also following some basic naming conventions. Here are some guidelines to keep in mind:

Use descriptive names: Variable names should clearly indicate what the variable represents. For example, instead of using "x" as a variable name, use something like "num_of_students" or "total_sales".

Use lowercase letters: In Python, variable names should be written in lowercase letters, with underscores (_) separating multiple words. This makes it easier to read and understand the variable names.

Avoid using reserved words: Certain words, such as "if", "while", "for", and "in", are reserved words in Python and should not be used as variable names.

Follow naming conventions: Python follows two main naming conventions: snake_case and CamelCase. Snake case is when you use all lowercase letters and underscores to separate words, while CamelCase is when you capitalize the first letter of each word (except for the first word). In Python, it is common to use snake_case for variable names and CamelCase for class names.

Be consistent: Once you choose a naming convention, be consistent and stick to it throughout your code. This makes your code easier to read and understand, especially when working with larger programs.

Here are some examples of variable names that follow these guidelines:

```
num_of_students
total_sales
average_grade
is_valid_input
first_name
last_name
user_input
```

By following these naming conventions, you can write code that is easier to read and understand, making it more efficient to work with and less prone to errors.

Data Types

String

In Python, a string is a sequence of characters enclosed in single quotes, double quotes, or triple quotes. Strings are one of the built-in data types in Python, and they are used to represent textual data.

Here's an example of a string in Python:

```
my_string = "Hello, world!"
```

Expressing string with quotes, " ", ' ', """ """, """ """

Multiline Strings with ''' ''' or """ """:

```
a = """Python  
class"""
```

Last one is called docstring. A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a docstring becomes the **doc** special attribute of that object.

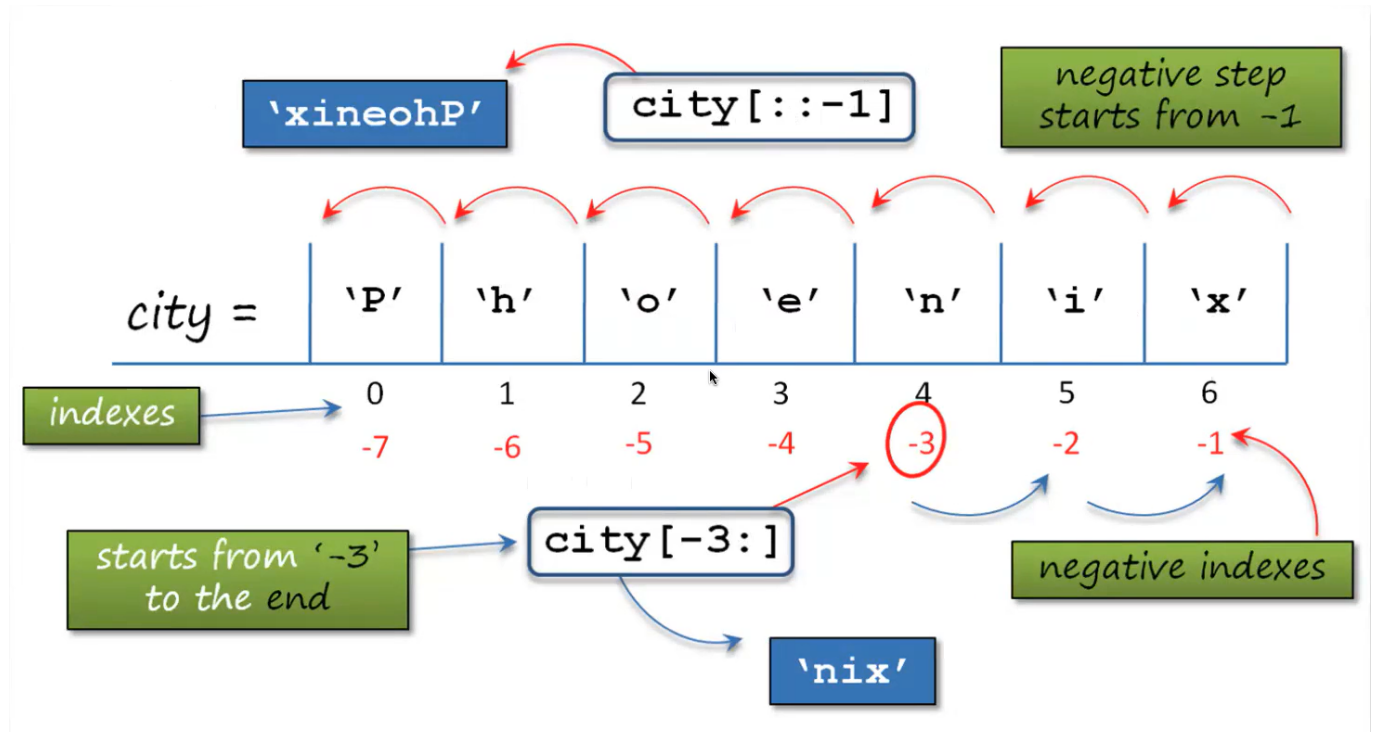
Strings are immutable, which means a string value cannot be updated.

We can verify this by trying to update a part of the string which will lead us to an error.

- String Length:

```
a = "Hello, World!"  
print(len(a))
```

Indexes



Indexes (positive and negative)

- Individual characters in a string may be chosen. If the string has length L, then the indices start from 0 for the initial character and go to L-1 for the rightmost character.

- Negative indices may also be used to count from the right end, -1 for the rightmost character through -L for the leftmost character.
- Call char by index:

```
a = "Hello World!"
print(a[1])
```

- Slicing
 - stringReference [start : pastEnd]
 - stringReference [: pastEnd]
 - stringReference [start :]
 - stringReference [:]
 - Change the order: s[::-1]

String Methods

Look at string operations slides.

- Methods

```
upper() # Returns an uppercase version of the string
lower() # Returns a lowercase version of the string
title()
capitalize()
count(item) # Returns the number of repetitions of the substring sub inside s.
join()
split() # Splits at any sequence of whitespace (blanks, newlines, tabs) and
returns the remaining parts of s as a list
split(sep) # Separator that gets removed from between the parts of the list.
startswith() # True/False
endswith() # True/False
strip([chars]) # Returns a copy of the string with both leading and trailing
characters removed
swapcase()
replace()
```

```
string3 = "ol eyyyyy "
```

prints the string by removing leading and trailing whitespaces

```
print(string3.strip())
```

prints the string by removing geeks

```
string3 = "ol eyyyyy "
string3.strip("y")
```

Searching a string is possible:

```
str.find()
str.index() # returns an error if can't find the character.
```

The find(query) method is built-in to standard python. Just call the method on the string object to search for a string, like so: obj.find('search').

The find() method searches for a query string and returns the character position if found. If the string is not found, it returns -1.

Numeric Types

- Integer
- Float
- Complexes (We will not touch that!)

Integer

- Literal integer values may not contain a decimal point.
- Integers may be arbitrarily large and are stored exactly.
- Integers have normal operations, with usual precedence (highest listed first):
 - **: exponentiation (5**3 means 5*5*5)
 - *, multiplication
 - /, division with float result
 - //, floor divisions result int, integer division (ignoring any remainder)
 - %, modulus just the remainder from division
 - +, -: addition, subtraction

Using Python as a calculator

The operators +, -, * and / work just like in most other languages:

```
2 + 2
# 4
```

Float

In Python, a float is a numeric data type that represents real numbers with decimal points. Float values are represented with the keyword float and can be written with a decimal point or using scientific notation.

Here is an example of float value in Python:

```
my_short_pi = 3.14
money_spent = 12.50
```


Boolean

The Boolean data type can be one of two values, either True or False. The values True and False will also always be with a capital T and F.

Comparison operators

Comparison operators are used to compare values and evaluate down to a single Boolean value of either True or False.

```
== Equal to
!= Not equal to
< Less than
> Greater than
<= Less than or equal to
>= Greater than or equal to
```

```
x = 5
y = 8

print("x == y:", x == y)
print("x != y:", x != y)
print("x < y:", x < y)
print("x > y:", x > y)
print("x <= y:", x <= y)
print("x >= y:", x >= y)
```

Type Conversion

You can convert everything to str but reverse is not always possible.

int() str() float()

Concatnation

It's not possible to concatnate different data types. But multiplying the string is ok:

```
some_int = 5
some_str = 'Repeat'
print(some_int * som_str)
```

Logical Operators

```
and True if both are true    x and y
or  True if at least one is true    x or y
```

```
not True only if false not x
```

- For example, they can be used to determine if the grade is passing and that the student is registered in the course. If both of these cases are true, then the student will be assigned a grade in the system.

```
print((9 > 7) and (2 < 4)) # Both original expressions are True
print((8 == 8) or (6 != 6)) # One original expression is True
print(not(3 <= 1))        # The original expression is False
```

- Falsy expressions:

```
# [] {} 0 None # False

if []:
    print("This is true")

# Will not print anything since [] is falsy.
```

- Operational order: not and or
- any & all (I think this topic is not necessary.)

☺ **Happy Coding!** 

