Python Class

Computer Programming

Organized and taught by Vanessa.

Introductions

Introduction:

- Brief bio about myself
- Brief bio about students

Course Objectives:

- "By the end of this course, you will be able to:"
  - Understand basic Python syntax and structure.
  - Write and debug simple Python programs.
  - Utilize Python’s built-in data structures.
  - Develop basic applications and scripts.
  - Data analysis, Data Visualization,Artificial Intelligence and Machine Learning.

Basics of Python

What is Python?

- Brief history and evolution.
- Popularity and use cases (web development, data analysis, machine learning, etc.).

Installing Python

- Guide on how to install Python on different operating systems (Windows, macOS, Linux).
- Introduction to IDEs and text editors (e.g., IDLE, PyCharm, VS Code).

Slide 2.1: Introduction to Python

Title: What is Python?
Content:

- **History and Evolution:**
  - Created by Guido van Rossum and released in 1991.
  - Named after Monty Python's Flying Circus.
  - Python 2 vs. Python 3: Differences and why we use Python 3.

- **Popularity and Use Cases:**
  - Widely used in web development, data science, artificial intelligence, automation, and more.
  - Companies using Python: Google, Facebook, Instagram, Spotify, etc.

Setting Up Python

- **Windows:**
  - Download from the official website: python.org.
  - Run the installer and ensure you check "Add Python to PATH".
- **macOS:**
  - Use Homebrew: `brew install python`.
  - Alternatively, download the installer from python.org.
- **Linux:**
  - Use the package manager: `sudo apt-get install python3`.
- **Using IDEs and Text Editors:**
  - IDEs: PyCharm, VS Code.
  - Text Editors: Sublime Text, Atom.
  - Using IDLE: Python’s built-in IDE.

Running Python Code

- **Interactive Shell:**
  - Open the Python shell by typing `python3` in the terminal.
  - Demonstrate basic commands (e.g., `print("Hello, World!")`)

.

- **Running Scripts:**
  - Create a Python file with `.py` extension.
  - Run the script using `python3 filename.py` in the terminal.

Writing Your First Python Program

Hello, World! Program:

- Write the code: `print("Hello, World!")`.
- Explaining the `print()` function.
- Run the program in the interactive shell and as a script.

Comments

Single-Line Comments:

- Use the # symbol to write comments.
- Example: `# This is a single-line comment`.

Multi-Line Comments:

- Use triple quotes `'''` or `"""` for multi-line comments.

Example:
python
Copy code

```
"""

This is a multi-line comment.

It spans multiple lines.

    """
```

Variables and Data Types

- **Variables:**
  - Naming conventions and rules.
  - Assigning values to variables.
  - Example: `x = 5`, `name = "Alice"`.

Do’s and Don’t about creating variables

1. Variable Naming Rules:

- A variable name must start with a letter or the underscore character.
- It cannot be any of the Python keywords.
- Variable names are case-sensitive.

Do’s

2. Naming Styles:

Note: Python supports various naming conventions, but it's best to adopt a single convention and apply it consistently throughout the codebase.

Common naming styles include:

- snake_case: All lowercase letters with words separated by underscores.
- Example: my_variable_name

- CamelCase: The first letter of each word is capitalized, except for the initial word. Example: myVariableName
-
- PascalCase: Similar to CamelCase, but the first letter of the first word is also capitalized. Example: MyVariableName

Do’s

- Choose variable names that are clear, concise, and reflect the purpose of the variable in the code.
- Descriptive names enhance code readability and understanding. For example, use user_profile instead of info_map.
- It is recommended to start a variable name with small letters.

Operators

- An operator tells a computer what to do with an operand.
- An operand is what an operator acts upon.(value)

These operators are categorized in;

Arithmetic operators(+, - ,/ ,// , *, %, **)

Examples;

```
num1 = 10

num2 = 20

print(num1 + num2)

print(num1 - num2)

print(num1 * num2)

print(num1 ** num2)

print(num1 % num2)

print(num1 / num2)

print(num1 // num2)
```

Assignment operators (= {assignment},+={adding two values},-=, *=, /=, %=, **=)

```
num3 = 50

num4 = 100

num3 += num4  #(it’s the same as num3 = num3 + num4)

print(num3)

num3 -= num4

print(num3)

num4 += num3

print(num4)
```

comparison operators(==, !=, < , >, >=, <=)

```
print(num1 == num2)

print(num1 != num2)

print(num1 < num2)

print(num1 <= num2)
```

logical operators(and, or, not,)

```
print(True and True)
print( True and False)
print(True or False)
print(not True)
```

Assignment

Research about ;

1. membership operators
2. Python Bitwise operators

Data Types

Definition:

These define the type of data a variable or object can hold and how the computer system should interpret its value.

These help in :

Categorization of values that are going to be stored in variable names to prevent computer memory wastage.

Examples;

- numeric data type
- String
- Sequence
- Mapping
- Booleans
- Set

etc.

In numerics we have;

- int [whole number]
- float[decimal point]
- complex

Examples of numerics;

```
num1 = 1000
num2 = 1000.0
num3 = 1 + 2j
num4 ="1000"

print(type(num1))
print(type(num2))
print(type(num3))
print(type(num4))
```

String(str)

- Any value in single or double quotes

```
name = "Vanessa"

print(type(name))
```

Sequence or List

refers to a ordered list of elements or terms.

These elements can be numbers, characters, or any other type of data, and they follow a specific order.

```
my_list = [0,2,4,6]

print(type(my_list))


my_list2 = [0,2,4,6,"Vanessa",10.5]

print(type(my_list2))
```

Tuple

It serves as an ordered collection of elements, often used for grouping related data.

NOTE:

Tuples are immutable in Python, meaning their elements cannot be changed after creation.

```
my_tuple = (0,2,4,6)

print(type(my_tuple))
```

Mapping(dict)

A dictionary is a built-in mapping type that consists of a collection of key-value pairs.

```
my_dict = {"uganda" : "kampala","Italy" :"Rome","France" :"Paris","Tanzania" : "Dodoma"}

print(type(my_dict))
```

Booleans

Boolean data type, often denoted as true and false, is designed to represent the two truth values of logic and Boolean algebra.

```
name1 = True

name2 = False

print(type(name1))
```

Set

A set gives you unordered list of items.

It eliminates duplicate items.

```
my_set = {0,5,10,15,20}

my_set2 = {1,1,2,2,10,10,3,3}

print(my_set)

print(my_set2)

print(set(my_dict))
```