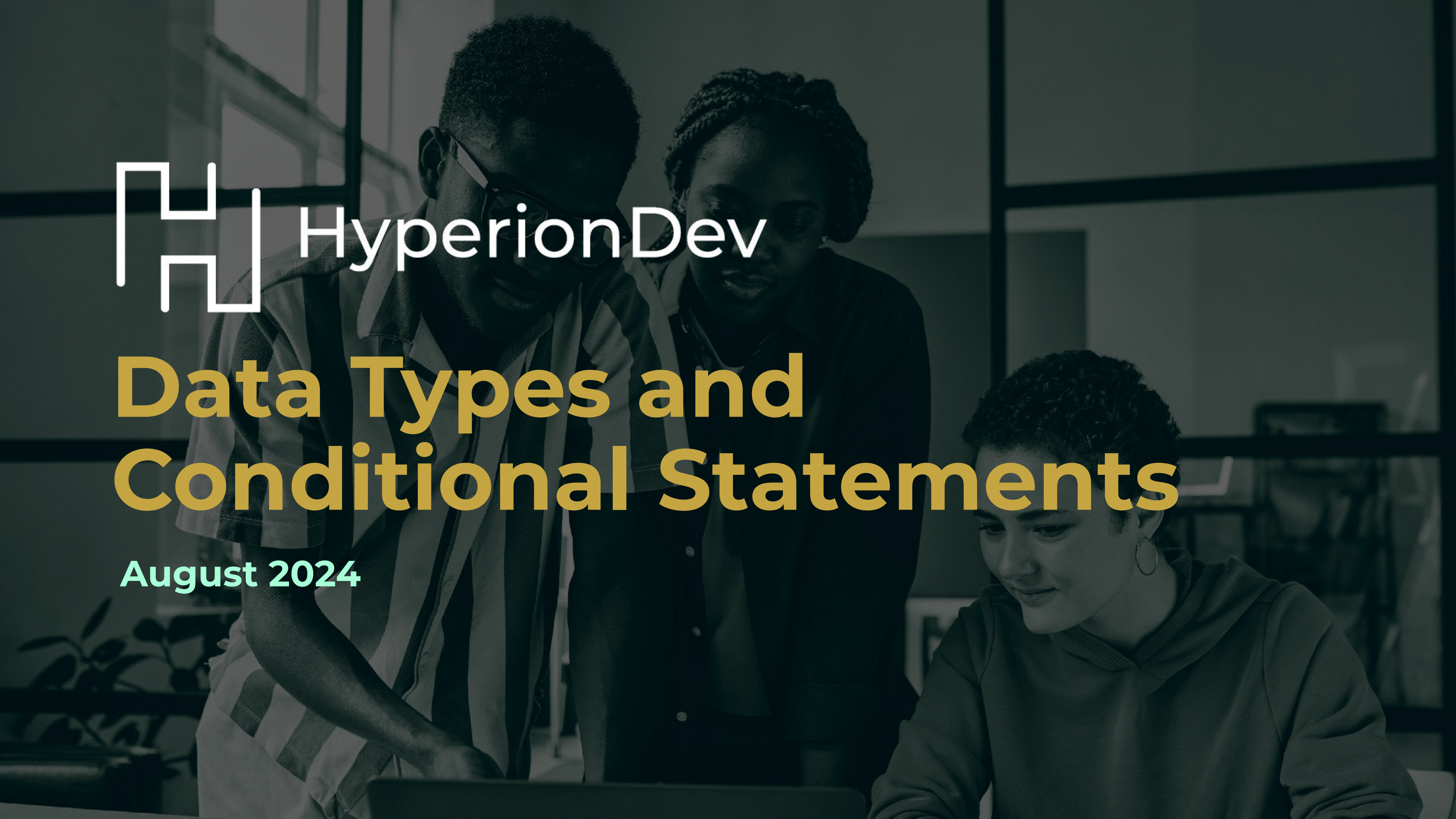


HyperionDev

Data Types and Conditional Statements

August 2024

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Learning Objectives

- ❖ **Define** and **use** variables and various data types in Python scripts.
- ❖ **Explain** and **implement** conditional statements to control the flow of a Python program.
- ❖ **Design** and **create** simple Python programs that solve specific problems using variables, data types, and conditional statements.

HyperionDev

Lecture Overview

→ Setting Up Dev Environments

→ Variables

→ Data Types

→ Conditional Statements

HyperionDev

Setting up your Dev Environments

HyperionDev

Installation Cheat Sheet

Installation:
Visit the official website and download the installer for your operating system: VsCode Download Link Website

Install VSCode:
Run the installer and follow the on-screen instructions to install VSCode on your system.

Settings:
Customize VSCode settings by going to File > Preferences > Settings (or by pressing Ctrl+,). You can configure editor settings, themes, and extensions preferences here.

Variables

HyperionDev

Variables

Symbols used to represent values stored in the computer's memory

❖ It is followed by the **name** of the variable, **“=” operator** and a **value/expression**.

```
num1 = 5
name = "Zahra"
```

❖ After a variable has been defined, its **name** can be used in expressions.

```
num2 = num1 + 5
```

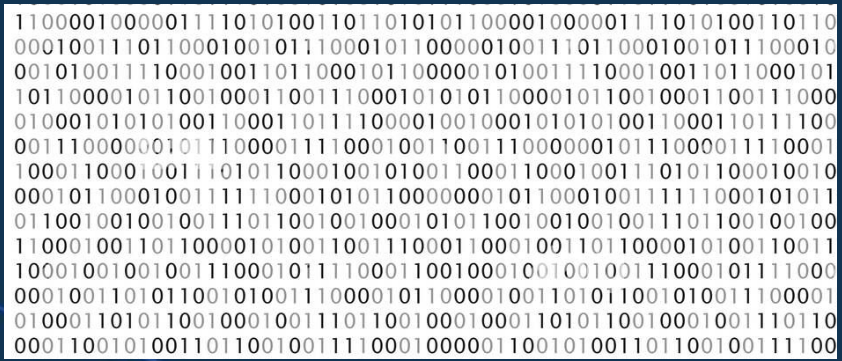
❖ The **“=” operator** can be used at any time on **existing variables** to **reassign** a new value to that variable.

Data Types

HyperionDev

Data Types

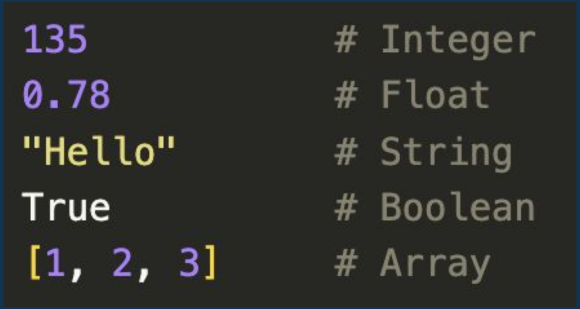
- A typical modern computer has more than **100 billion bits** in its volatile data storage (**working memory**).
- To be able to work with such quantities of **bits** without getting lost, we separate them into **chunks** that represent **pieces of information**.



HyperionDev

Data Types

- In Python, those chunks are called **values**.
- Every **value** has a **type** that determines its **role**.
- Understanding data types is fundamental because it allows you to work with different kinds of information effectively.

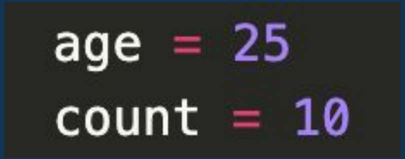


```
135             # Integer
0.78            # Float
"Hello"         # String
True            # Boolean
[1, 2, 3]       # Array
```

HyperionDev

Integers (int)

- Whole numbers **without decimals**.
- It can be **positive, negative, or zero**.
- Integers are used to represent quantities that can be counted or measured in whole units.

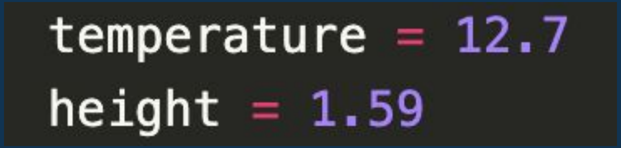


```
age = 25
count = 10
```

HyperionDev

Floats (float)

- Numbers **with decimal points**.
- Floats are used to represent quantities that can have **fractional parts**, such as measurements, percentages, or values resulting from mathematical calculations.

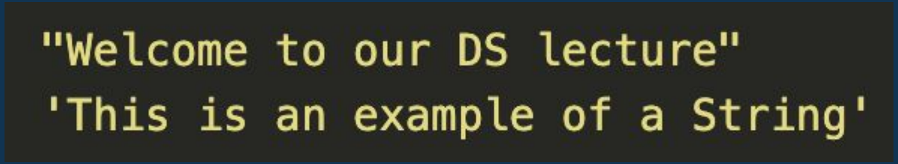


```
temperature = 12.7
height = 1.59
```

HyperionDev

Strings (str)

- A **sequence of characters**, such as letters, numbers, or symbols.
- Enclosed within **single quotes** ('') or **double quotes** ("").
- Strings are used to represent text data in Python.

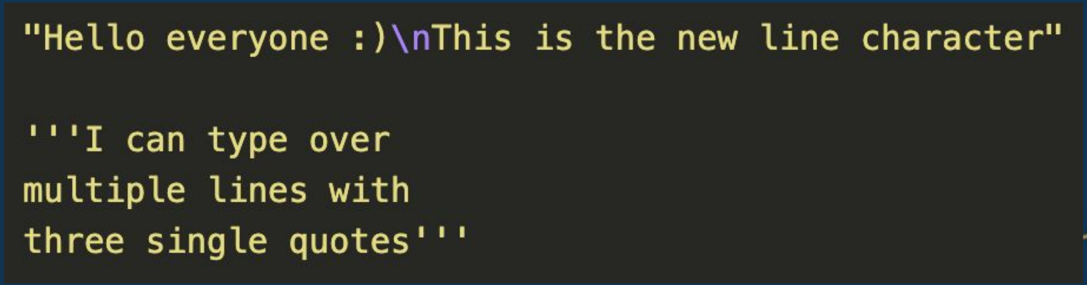


```
"Welcome to our DS lecture"
'This is an example of a String'
```

HyperionDev

Strings (str)

- A **backslash (\)** inside quoted text indicates that the character after it has a special meaning. This is called **escaping** the character.
- Newlines can be included only when the string is quoted with three quotation marks.

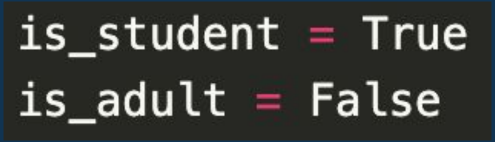


```
"Hello everyone :)\nThis is the new line character"

'''I can type over
multiple lines with
three single quotes'''
```

Booleans (bool)

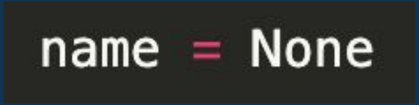
- It is often useful to have a value that distinguishes between only two possibilities, like “yes” and “no” or “on” and “off”.
- For this purpose, Python has a **Boolean** type, which has just two values, **true** and **false**, written as those words.
- Booleans are used in logical operations and conditional statements to make decisions based on whether a condition is true or false



```
is_student = True
is_adult = False
```

NoneType (None)

- Represents the absence of a value or a **null value**.
- It is used to indicate that a variable does not have a value assigned to it.



```python
name = None
```

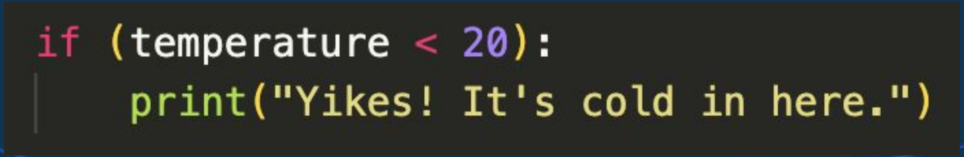
Conditional Statements

HyperionDev

Conditional Statements

Statements that perform different actions depending on whether a condition evaluates to true or false.

- Conditional statements are like decision-making tools in programming.
- Depending on whether a condition is true or false, you can choose to run different parts of your code.
- It's a way to make your program smarter and more flexible, allowing it to adapt to different scenarios as needed.



```
if (temperature < 20):
    print("Yikes! It's cold in here.")
```

HyperionDev

Conditional Statements

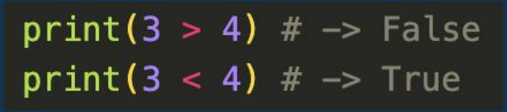
- **Conditional execution** is created with the **if** keyword in Python.
- We want some code to be executed **if**, and only **if**, a certain **condition** holds.
- A **condition** is written after the **if** keyword, between parentheses, followed by a **semicolon** (**:**), then the statement to execute.
- The condition is a **boolean expression** which we form using values, **comparison** and **logical operators**.

HyperionDev

Comparison Operations

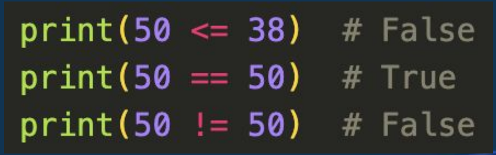
❖ The **>** and **<** signs are the traditional symbols for **“is greater than”** and **“is less than”**, respectively.

❖ Applying them results in a Boolean value that indicates whether they hold true in this case.



```
print(3 > 4) # -> False
print(3 < 4) # -> True
```

❖ Other similar operators are **>= (greater than or equal to)**, **<= (less than or equal to)**, **== (equal to)**, and **!= (not equal to)**.

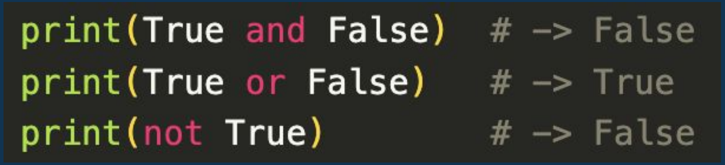


```
print(50 <= 38)  # False
print(50 == 50)  # True
print(50 != 50)  # False
```

HyperionDev

Logical Operators

- ❖ Python supports three logical operators: **and**, **or**, and **not**.
- ❖ The **and** operator represents logical **AND**
  - ➢ Its result is **true** only if **both** the values given to it are **true**.
- ❖ The **or** operator denotes logical **OR**.
  - ➢ Its result is **true** if **either** the values given to it are **true**.
- ❖ **Not** flips the value given to it.
  - ➢ **not True** produces **False** and **not False** gives **True**.

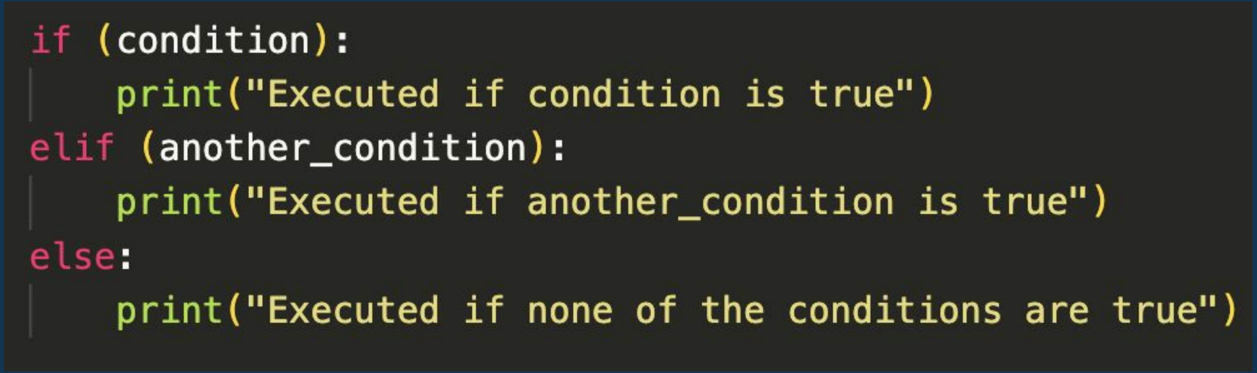


```
print(True and False)  # -> False
print(True or False)   # -> True
print(not True)        # -> False
```

CoGrammar

Conditional Statements

❖ There are three primary types of conditional statements in programming:

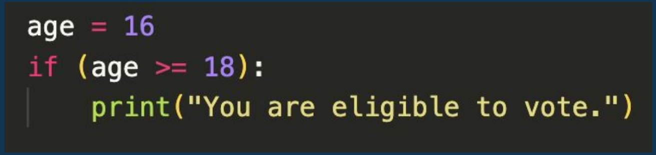


```python
if (condition):
    print("Executed if condition is true")
elif (another_condition):
    print("Executed if another_condition is true")
else:
    print("Executed if none of the conditions are true")
```

HyperionDev

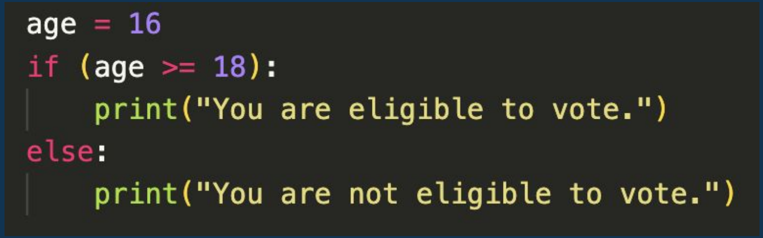
Conditional Statements

❖ **If statement:** executes a block of code if a specified condition is true.



```python
age = 16
if (age >= 18):
    print("You are eligible to vote.")
```

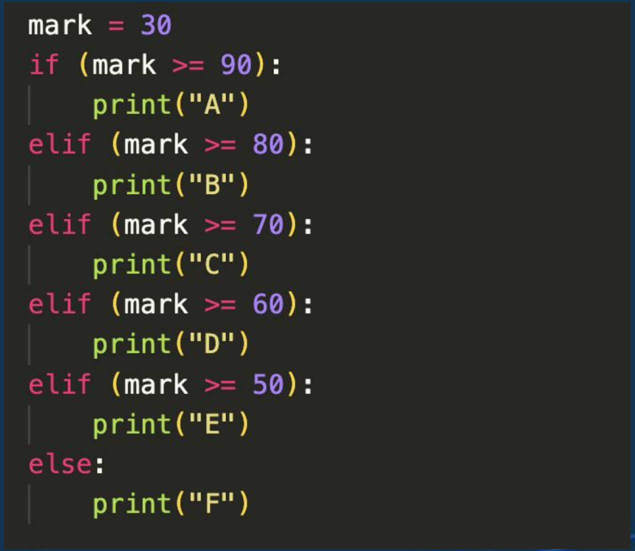
❖ **If-Else statement:** executes one block of code if the condition is true and another block if the condition is false.



```python
age = 16
if (age >= 18):
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

Conditional Statements

❖ **If-Elif-Else statement:** It allows you to check multiple conditions and execute different blocks of code depending on which condition is true.



```python
mark = 30
if (mark >= 90):
    print("A")
elif (mark >= 80):
    print("B")
elif (mark >= 70):
    print("C")
elif (mark >= 60):
    print("D")
elif (mark >= 50):
    print("E")
else:
    print("F")
```

HyperionDev

Questions and Answers

HyperionDev

Thank you for attending

HyperionDev