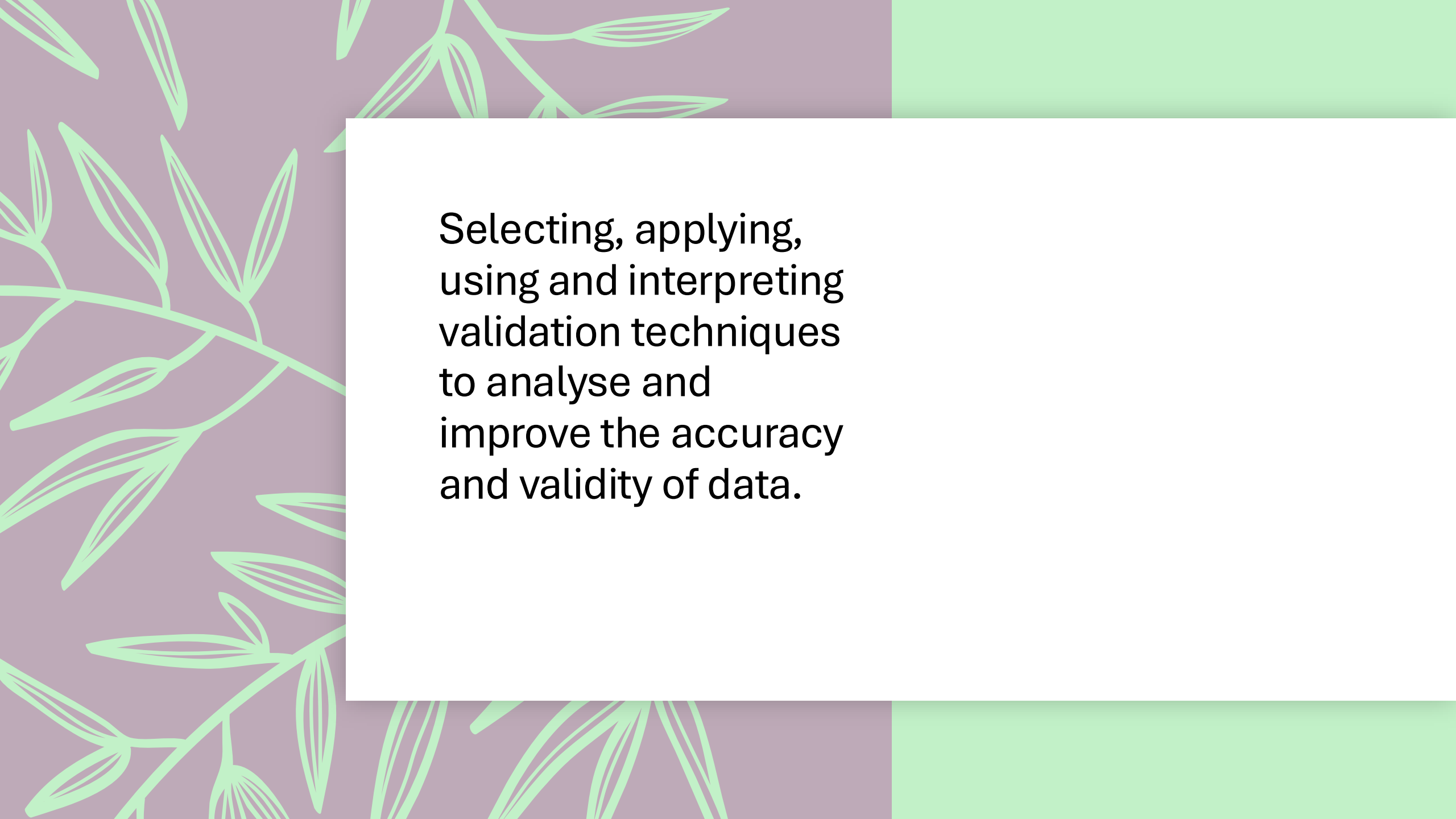


BTEC Level 3 Computing

Unit 1 - Principles of Computer Science

Validating Data



Selecting, applying,
using and interpreting
validation techniques
to analyse and
improve the accuracy
and validity of data.

Validating Data

What is Data Validation?

- Data validation is the process of ensuring that data entered into a system (like a form, database, or program) is correct, relevant, and consistent.
- It's like a quality control check for your data.
- The goal is to prevent bad data from being stored or used, which can lead to errors, inconsistencies, and unreliable results.

Why validate Data?

Data Integrity: Ensures that the data is accurate and reliable. This is crucial for making good decisions based on the data.

Error Prevention: Catches errors early, before they cause problems in other parts of the system. It's much easier to fix bad data at the source than to deal with the consequences later.

Improved User Experience: Provides helpful feedback to users when they enter incorrect data, guiding them to correct it. This prevents frustration and makes the system easier to use.

System Stability: Prevents bad data from crashing the system or corrupting databases.

Security: Can help prevent malicious data entry, such as SQL injection attacks or cross-site scripting (XSS).

Why validate Data?

System Stability: Prevents bad data from crashing the system or corrupting databases.

Security: Can help prevent malicious data entry, such as SQL injection attacks or cross-site scripting (XSS).

Types Of Data Validation

- Data Type.
- Range.
- Constraints.
- Boolean.

Data Type Data Validation

Data type validation is a specific type of data validation that focuses on ensuring that the data entered is of the expected data type.

For example, if you expect an integer (whole number), you would validate that the input is indeed an integer and not a string, a float, or some other data type.

Data Type Data Validation Example

- `age = "51"`

```
if type(age) is int:  
    print("Age accepted")  
else:  
    print("Age NOT  
accepted")
```

- `age = 51`

```
if type(age) is int:  
    print("Age accepted")  
else:  
    print("Age NOT  
accepted")
```


Range Data Validation

- Used to ensure the value is between the required.
- For example a person cannot be less than 0 years old "AND" they cannot be over 150 years old.

Range Data Validation Example

- `age = -1`

```
if age < 0 or age > 150:  
    print("Age is not valid, try again")  
else:  
    print("Age is valid")
```


Constraint Data Validation



This is not a single thing.



You decide the constraint.



For example, the length of a name cannot be longer than 30 characters.

Constraint Data Validation Example

```
• name = input("Enter your name please...")  
  
if len(name) == 0 or len(name) > 30:  
    print("Please try again")  
else:  
    print("Thank you !")
```


Boolean Data Validation

- This will check if the evaluation is True or False.
- For example, if your name is longer than 30 characters, please contact the customer service department to create an account.

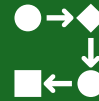
Boolean Data Validation Example

```
• name = input("Enter your name please...")  
  
  if len(name) > 30:  
      print("Please contact Customer Service")  
  else:  
      print("Thank you !")
```


Post-check actions



Post-check actions are actions that are performed *after* a validation check has been completed.



They're the steps you take based on whether the validation was successful or not.



For example, if the age is more than 150 ask them to try again.



For example, if the data is correct add it to the database.

Post-check actions Example

- # Post Check Data Validation

```
while True: # Loop until a valid age is entered
    try:
        age = int(input("Enter your age: ")) # Get age input and try to convert
        to int

        if age > 150:
            print("That age is too high. Please try again.") # Ask to try again
        elif age < 0:
            print("Age cannot be negative. Please try again.")
        else:
            print("Age is valid:", age) # Age is valid, exit the loop
            break # Exit the loop if the age is valid

    except ValueError: # Handle cases where the input is not a number
        print("Invalid input. Please enter a number.")
```




Next Time

Control Structures

