Practical 10

Aim: Write a program for validating the user input

Theory:

In Python, we can validate user input by implementing conditional statements and input validation functions. Here are some ways to validate user input in Python:

1. Using conditional statements: We can use if-else statements to check if the input is in the correct format or range. For example, if we are expecting an integer input, we can check if the input is an integer using the isinstance() function, and prompt the user to re-enter the input if it is not.

```
while True:
    user_input = input("Enter a number: ")
    if isinstance(user_input, int):
        break
    else:
        print("Invalid input, please enter a valid integer.")
```

2. Using input validation functions: We can define functions to validate the input and return a boolean value indicating whether the input is valid or not. For example, we can define a function to check if the input is a valid email address using regular expressions.

```
import re

def validate_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]
{2,}'


    return bool(re.match(pattern, email))

while True:
    user_email = input("Enter your email address: ")
    if validate_email(user_email):
        break
    else:
        print("Invalid email address, please enter a valid
email.")
```

3. Using try-except blocks: We can use try-except blocks to catch input errors and prompt the user to re-enter the input. For example, if we are expecting a numerical input, we can catch the ValueError exception if the input is not numerical and prompt the user to re-enter the input.

```
while True:
    try:
        user_input = float(input("Enter a number: "))
        break
    except ValueError:
        print("Invalid input, please enter a valid number.")
```

These are some ways to validate user input in Python. The approach to use depends on the type of input and the specific requirements of the program.