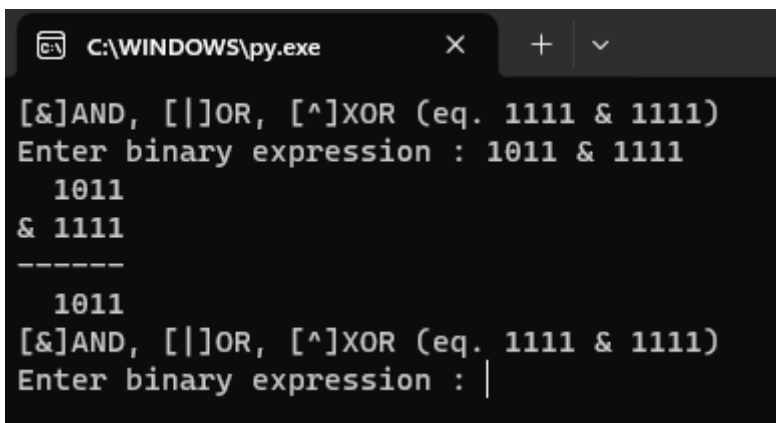


Bitwise OR XOR AND Calculator

Example:



```
C:\WINDOWS\py.exe
[&]AND, [|]OR, [^]XOR (eq. 1111 & 1111)
Enter binary expression : 1011 & 1111
    1011
& 1111
-----
    1011
[&]AND, [|]OR, [^]XOR (eq. 1111 & 1111)
Enter binary expression : |
```

Code:

```
def valid_expression(expression: str) -> bool:
    """ Returns True if string is valid binary expression. False otherwise.

    - Operand is valid binary digits
    - Operator is either '|', '&', or '^'
    - Length of operands is more than 2 digits
    - Both operands are same length
    - Expression is in the format of "operand operator operand", seperated by blanks"""
    if len(expression.split()) == 3:
        first, operator, second = expression.split()
    else:
        print("Format must be "operand operator operand", separated by blanks")
        return False
    for i in range(len(first)):
        if not first[i] in '10' or not second[i] in '10': # If digit is not 1 or 0
            print('Operand does not consist of binary digits')
            return False
```

```

if not operator in '&^|':
    print(f'{operator} is invalid. Must be !, & or ^')
    return False
elif len(first) < 2 or len(second) < 2:
    print('Length of the operands must be at least 2 digits')
    return False
elif len(first) != len(second):
    print('The operands are of different length')
    return False
return True

```

```

def calc_binary_expression(firstBinary: str, operator: str, secondBinary: str) -> str:

```

```

    """ Returns the result (str) of the binary expression. """
    result = ""
    for i in range(0, len(firstBinary)):
        z = 0
        if operator == "&":
            z = 1 if int(firstBinary[i]) and int(secondBinary[i]) else 0
        elif operator == "|":
            z = 1 if int(firstBinary[i]) or int(secondBinary[i]) else 0
        elif operator == "^":
            z = 1 if int(firstBinary[i]) != int(secondBinary[i]) else 0
        result = str(result) + str(z)
    return result

```

```

def display_bin_calculation(firstBinary: str, operator: str, secondBinary: str) -> None:

```

```

    """ Print the result of the binary and operator from parameter """
    print(f"{'':<2}{firstBinary}")
    print(f"{'':<2}{operator} {secondBinary}")
    print(f"{'':<2}{calc_binary_expression(firstBinary, operator, secondBinary)}")

```

```

def main() -> None:

```

```

    """ This program reads in a binary expression as a string and evaluates the result. """
    while True:
        userInput = input("&AND, []OR, []XOR (eq. 1111 & 1111) \nEnter binary expression : ")

        # Without any input, break the loop
        if len(userInput) == 0:
            print("End of program.")
            break

```

```
if valid_expression(userInput):  
    firstBinary, operator, secondBinary = userInput.split()  
    display_bin_calculation(firstBinary, operator, secondBinary)
```

```
if __name__ == "__main__":  
    main()
```

Revision #2

Created 31 December 2023 08:30:08 by aki

Updated 31 December 2023 08:38:24 by aki