Evaluate Infix Expression

```
def evaluateInfix(string):
    operator = []
    operands = []

    for char in string:
        if char == '(':
            operator.append(char)
        elif char in {'1','2','3','4','5','6','7','8','9'}:
            operands.append(int(char))
        elif char == ')':
            while operator[-1] != '(':
                v2 = operands.pop()
                v1 = operands.pop()
                temp = operator.pop()
                res = evaluateOperand(v1, v2, temp)
                operands.append(res)
            operator.pop()
        elif char in {'+', '-', '*', '/'}:
            while len(operator) > 0 and operator[-1] != '(' and
getPrecendence(char) <= getPrecendence(operator[-1]):
                v2 = operands.pop()
                v1 = operands.pop()
                temp = operator.pop()
                res = evaluateOperand(v1, v2, temp)
                operands.append(res)
            operator.append(char)

    while len(operator)>0:
        v2 = operands.pop()
        v1 = operands.pop()
        temp = operator.pop()
        res = evaluateOperand(v1, v2, temp)
        operands.append(res)
    return operands.pop()

    # return


def getPrecendence(char):
```

```
    if char == '+':
        return 1
    elif char == '-':
        return 1
    elif char == '*':
        return 2
    else:
        return 2


def evaluateOperand(v1, v2, char):
    if char == '+':
        return v1 + v2
    elif char == '-':
        return v1 - v2
    elif char == '*':
        return v1 * v2
    else:
        return v1 / v2


string = '2+3/4-(5*6)'
print(evaluateInfix(string))
```