Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

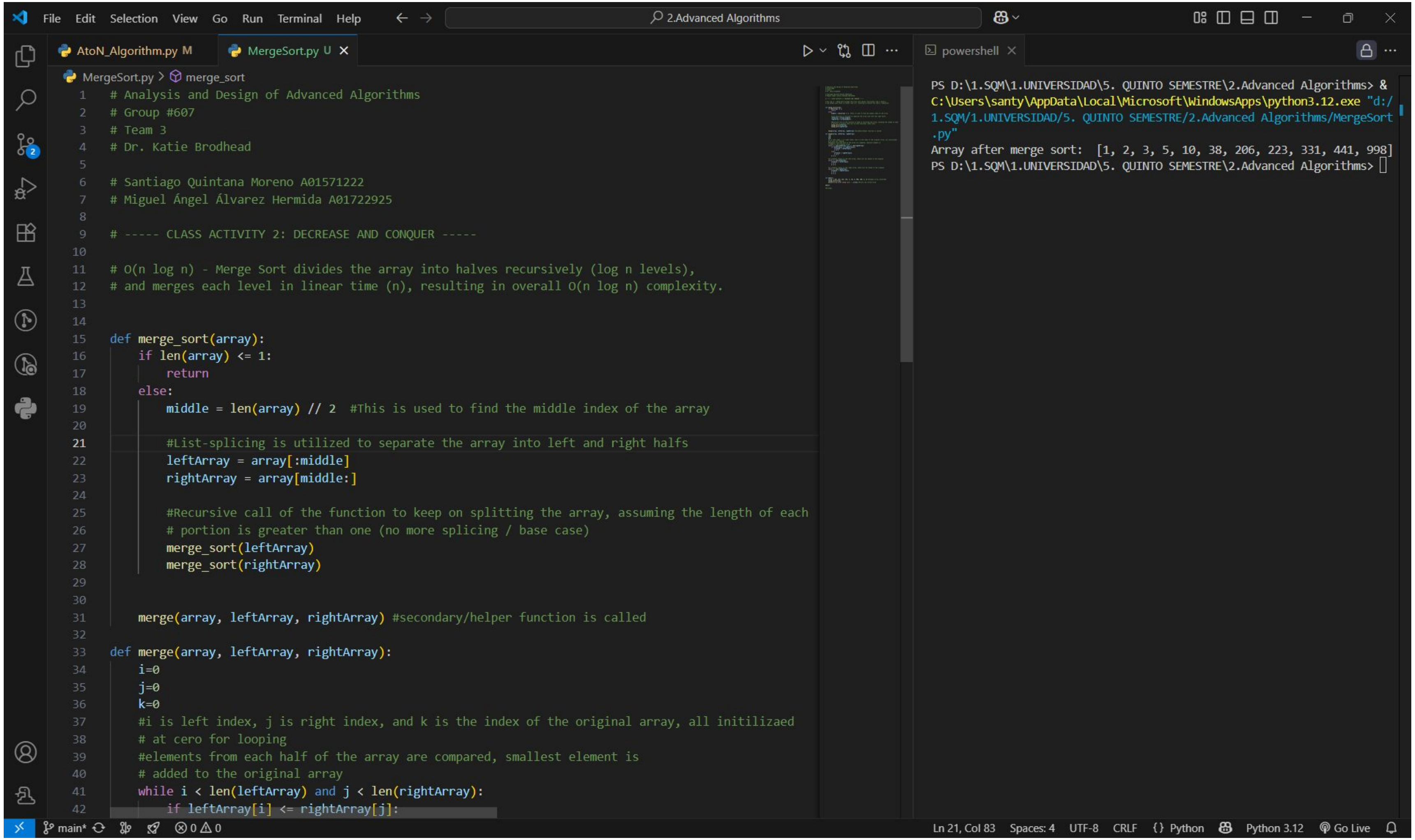
Class Activity 2:
Decrease and Conquer & Divide and Conquer

Group: 607
Team #3
Dr. Katie Brodhead

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

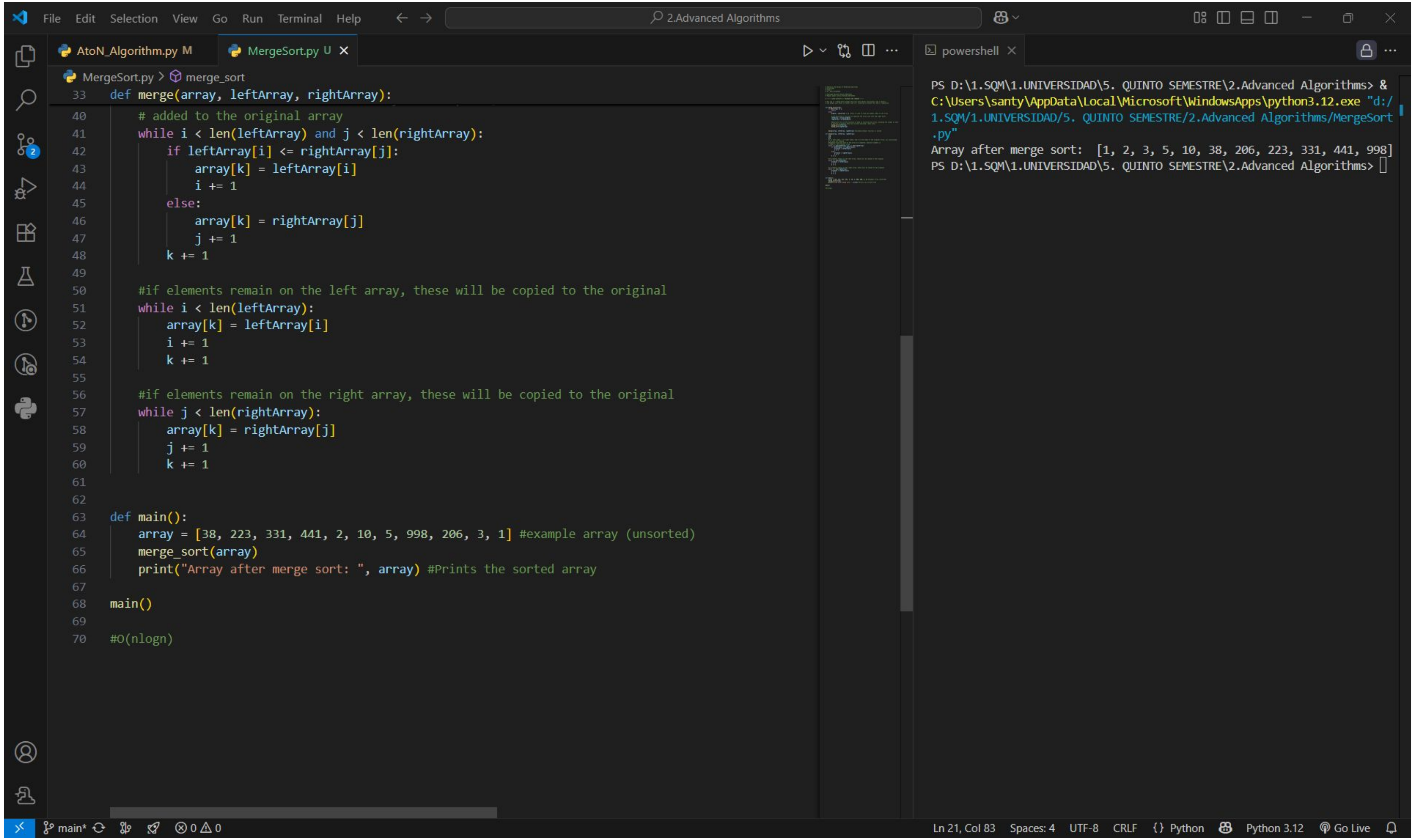
Merge Sort

https://colab.research.google.com/drive/1jdUcKlCduHmBRZuC65mKSWzPKujHKY7B?usp=sharing


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
AtoN_Algorithm.py M
MergeSort.py U
MergeSort.py > merge_sort
powershell
main*
0 0
Ln 21, Col 83 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Dr. Katie Brodhead

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ----- CLASS ACTIVITY 2: DECREASE AND CONQUER -----

# O(n log n) - Merge Sort divides the array into halves recursively (log n levels),
# and merges each level in linear time (n), resulting in overall O(n log n) complexity.


def merge_sort(array):
    if len(array) <= 1:
        return
    else:
        middle = len(array) // 2  #This is used to find the middle index of the array

        #List-splicing is utilized to separate the array into left and right halfs
        leftArray = array[:middle]
        rightArray = array[middle:]

        #Recursive call of the function to keep on splitting the array, assuming the length of each
        # portion is greater than one (no more splicing / base case)
        merge_sort(leftArray)
        merge_sort(rightArray)


    merge(array, leftArray, rightArray) #secondary/helper function is called

def merge(array, leftArray, rightArray):
    i=0
    j=0
    k=0
    #i is left index, j is right index, and k is the index of the original array, all initilizaed
    # at cero for looping
    #elements from each half of the array are compared, smallest element is
    # added to the original array
    while i < len(leftArray) and j < len(rightArray):
        if leftArray[i] <= rightArray[j]:
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/MergeSort.py"
Array after merge sort:  [1, 2, 3, 5, 10, 38, 206, 223, 331, 441, 998]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> 
```

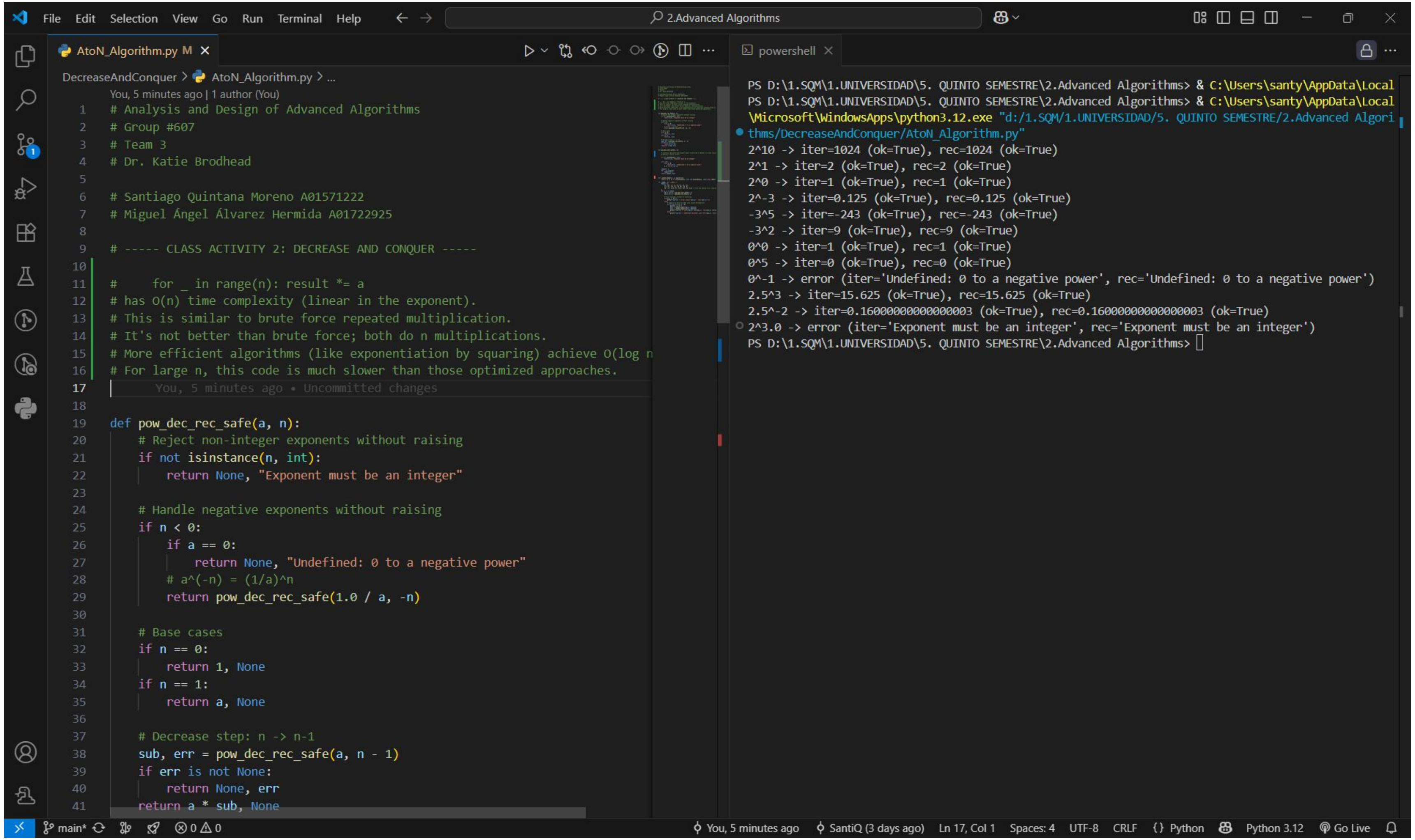

File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
AtoN_Algorithm.py M
MergeSort.py U
MergeSort.py > merge_sort
powershell
main*
Ln 21, Col 83 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 Go Live

```python
def merge(array, leftArray, rightArray):
    # added to the original array
    while i < len(leftArray) and j < len(rightArray):
        if leftArray[i] <= rightArray[j]:
            array[k] = leftArray[i]
            i += 1
        else:
            array[k] = rightArray[j]
            j += 1
        k += 1

    #if elements remain on the left array, these will be copied to the original
    while i < len(leftArray):
        array[k] = leftArray[i]
        i += 1
        k += 1

    #if elements remain on the right array, these will be copied to the original
    while j < len(rightArray):
        array[k] = rightArray[j]
        j += 1
        k += 1


def main():
    array = [38, 223, 331, 441, 2, 10, 5, 998, 206, 3, 1] #example array (unsorted)
    merge_sort(array)
    print("Array after merge sort: ", array) #Prints the sorted array

main()

#O(nlogn)
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &
C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/
1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/MergeSort
.py"
Array after merge sort:  [1, 2, 3, 5, 10, 38, 206, 223, 331, 441, 998]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

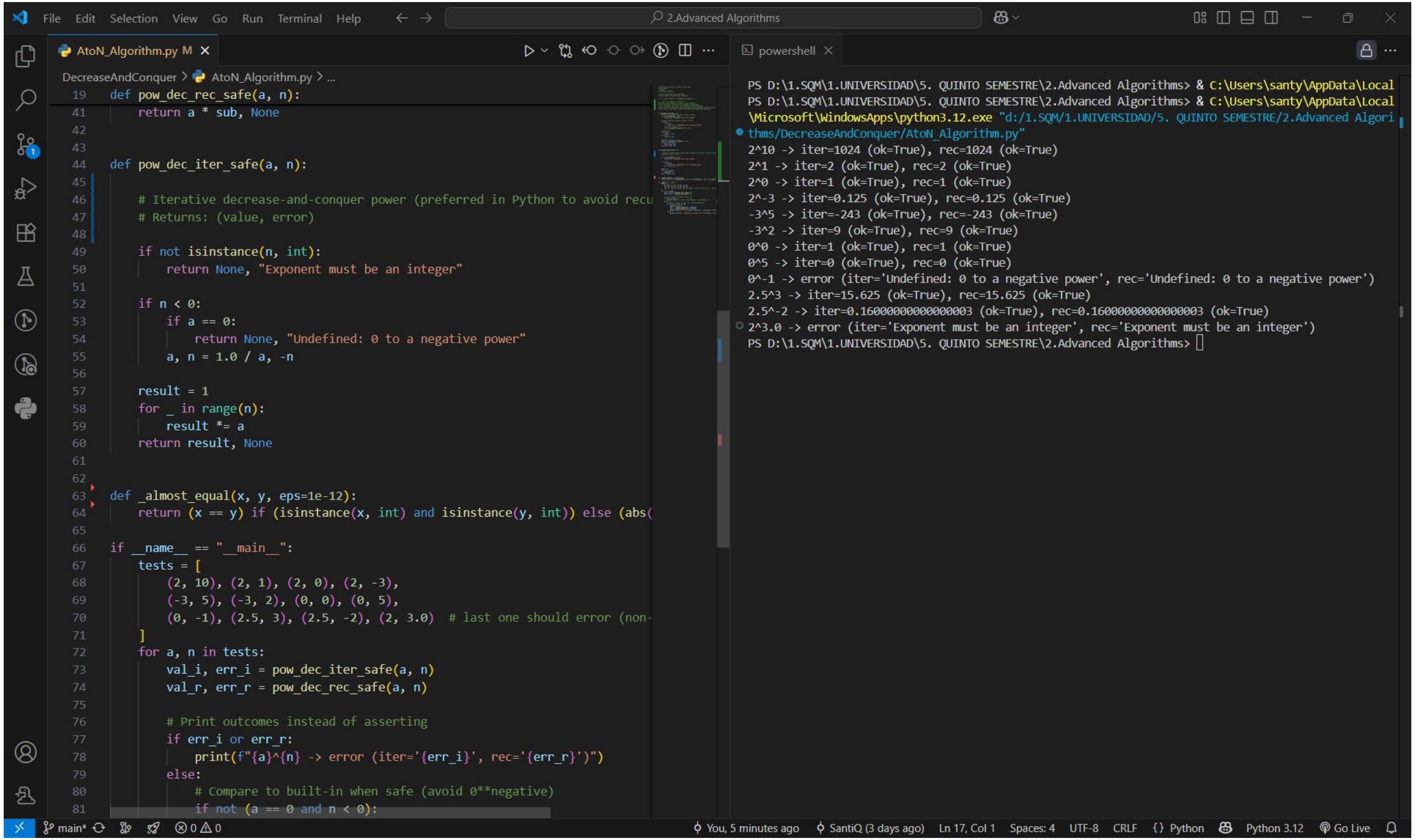
a^n using a conquer method

https://colab.research.google.com/drive/1QKHwbSd00QR6IRwO67kAfImr2VI4Y86H?usp=sharing



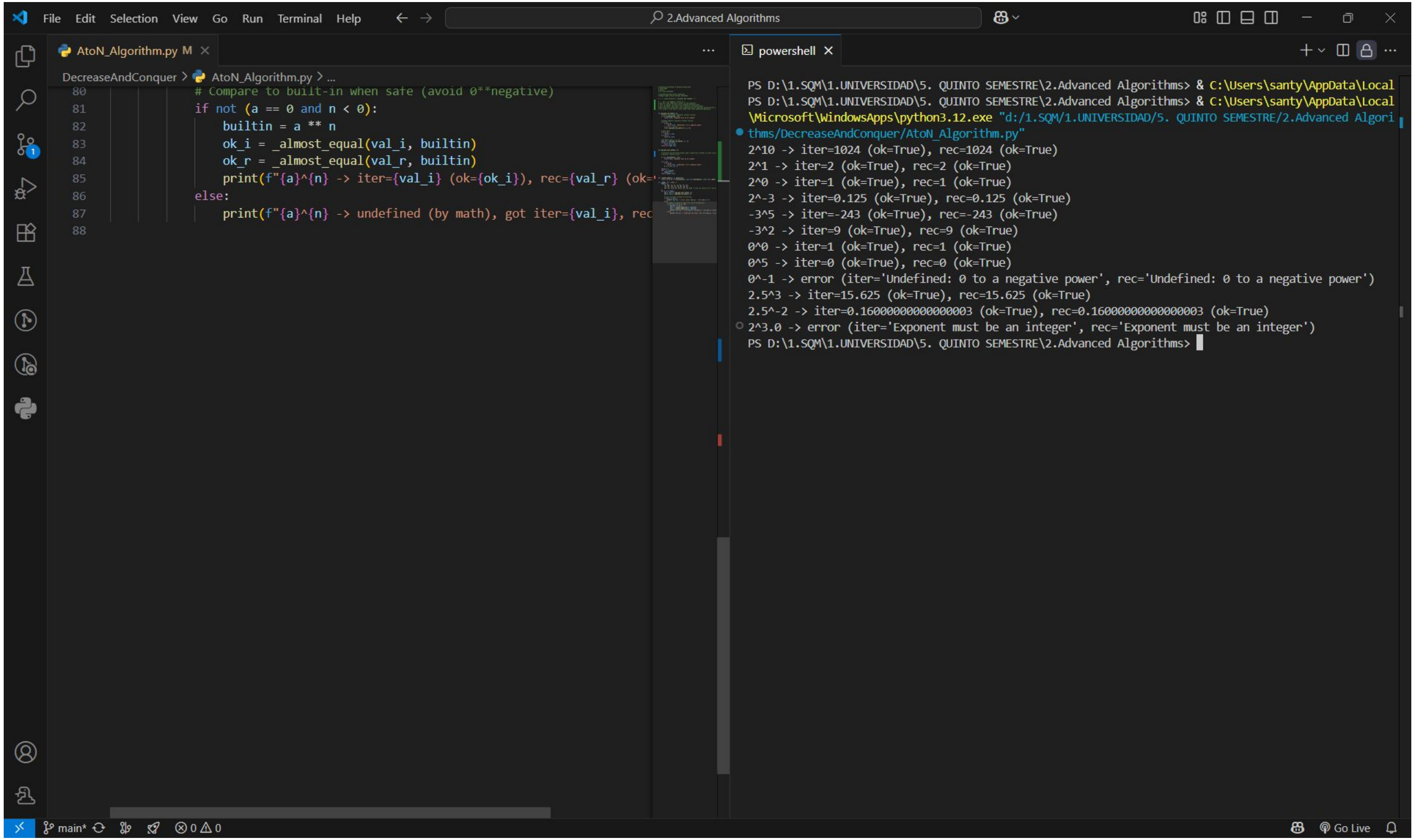
```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Dr. Katie Brodhead

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ----- CLASS ACTIVITY 2: DECREASE AND CONQUER -----

#     for _ in range(n): result *= a
# has O(n) time complexity (linear in the exponent).
# This is similar to brute force repeated multiplication.
# It's not better than brute force; both do n multiplications.
# More efficient algorithms (like exponentiation by squaring) achieve O(log n
# For large n, this code is much slower than those optimized approaches.


def pow_dec_rec_safe(a, n):
    # Reject non-integer exponents without raising
    if not isinstance(n, int):
        return None, "Exponent must be an integer"

    # Handle negative exponents without raising
    if n < 0:
        if a == 0:
            return None, "Undefined: 0 to a negative power"
        # a^(-n) = (1/a)^n
        return pow_dec_rec_safe(1.0 / a, -n)

    # Base cases
    if n == 0:
        return 1, None
    if n == 1:
        return a, None

    # Decrease step: n -> n-1
    sub, err = pow_dec_rec_safe(a, n - 1)
    if err is not None:
        return None, err
    return a * sub, None
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algori
thms/DecreaseAndConquer/AtoN_Algorithm.py"
2^10 -> iter=1024 (ok=True), rec=1024 (ok=True)
2^1 -> iter=2 (ok=True), rec=2 (ok=True)
2^0 -> iter=1 (ok=True), rec=1 (ok=True)
2^-3 -> iter=0.125 (ok=True), rec=0.125 (ok=True)
-3^5 -> iter=-243 (ok=True), rec=-243 (ok=True)
-3^2 -> iter=9 (ok=True), rec=9 (ok=True)
0^0 -> iter=1 (ok=True), rec=1 (ok=True)
0^5 -> iter=0 (ok=True), rec=0 (ok=True)
0^-1 -> error (iter='Undefined: 0 to a negative power', rec='Undefined: 0 to a negative power')
2.5^3 -> iter=15.625 (ok=True), rec=15.625 (ok=True)
2.5^-2 -> iter=0.16000000000000003 (ok=True), rec=0.16000000000000003 (ok=True)
2^3.0 -> error (iter='Exponent must be an integer', rec='Exponent must be an integer')
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```



```python
def pow_dec_rec_safe(a, n):
    return a * sub, None


def pow_dec_iter_safe(a, n):

    # Iterative decrease-and-conquer power (preferred in Python to avoid recu
    # Returns: (value, error)

    if not isinstance(n, int):
        return None, "Exponent must be an integer"

    if n < 0:
        if a == 0:
            return None, "Undefined: 0 to a negative power"
        a, n = 1.0 / a, -n

    result = 1
    for _ in range(n):
        result *= a
    return result, None


def _almost_equal(x, y, eps=1e-12):
    return (x == y) if (isinstance(x, int) and isinstance(y, int)) else (abs(

if __name__ == "__main__":
    tests = [
        (2, 10), (2, 1), (2, 0), (2, -3),
        (-3, 5), (-3, 2), (0, 0), (0, 5),
        (0, -1), (2.5, 3), (2.5, -2), (2, 3.0)  # last one should error (non-
    ]
    for a, n in tests:
        val_i, err_i = pow_dec_iter_safe(a, n)
        val_r, err_r = pow_dec_rec_safe(a, n)

        # Print outcomes instead of asserting
        if err_i or err_r:
            print(f"{a}^{n} -> error (iter='{err_i}', rec='{err_r}')")
        else:
            # Compare to built-in when safe (avoid 0**negative)
            if not (a == 0 and n < 0):
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algori
thms/DecreaseAndConquer/AtoN_Algorithm.py"
2^10 -> iter=1024 (ok=True), rec=1024 (ok=True)
2^1 -> iter=2 (ok=True), rec=2 (ok=True)
2^0 -> iter=1 (ok=True), rec=1 (ok=True)
2^-3 -> iter=0.125 (ok=True), rec=0.125 (ok=True)
-3^5 -> iter=-243 (ok=True), rec=-243 (ok=True)
-3^2 -> iter=9 (ok=True), rec=9 (ok=True)
0^0 -> iter=1 (ok=True), rec=1 (ok=True)
0^5 -> iter=0 (ok=True), rec=0 (ok=True)
0^-1 -> error (iter='Undefined: 0 to a negative power', rec='Undefined: 0 to a negative power')
2.5^3 -> iter=15.625 (ok=True), rec=15.625 (ok=True)
2.5^-2 -> iter=0.16000000000000003 (ok=True), rec=0.16000000000000003 (ok=True)
2^3.0 -> error (iter='Exponent must be an integer', rec='Exponent must be an integer')
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
AtoN_Algorithm.py M
DecreaseAndConquer > AtoN_Algorithm.py > ...
powershell
main*
Go Live

```python
            # Compare to built-in when safe (avoid 0**negative)
            if not (a == 0 and n < 0):
                builtin = a ** n
                ok_i = _almost_equal(val_i, builtin)
                ok_r = _almost_equal(val_r, builtin)
                print(f"{a}^{n} -> iter={val_i} (ok={ok_i}), rec={val_r} (ok=
            else:
                print(f"{a}^{n} -> undefined (by math), got iter={val_i}, rec
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local
\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algori
thms/DecreaseAndConquer/AtoN_Algorithm.py"
2^10 -> iter=1024 (ok=True), rec=1024 (ok=True)
2^1 -> iter=2 (ok=True), rec=2 (ok=True)
2^0 -> iter=1 (ok=True), rec=1 (ok=True)
2^-3 -> iter=0.125 (ok=True), rec=0.125 (ok=True)
-3^5 -> iter=-243 (ok=True), rec=-243 (ok=True)
-3^2 -> iter=9 (ok=True), rec=9 (ok=True)
0^0 -> iter=1 (ok=True), rec=1 (ok=True)
0^5 -> iter=0 (ok=True), rec=0 (ok=True)
0^-1 -> error (iter='Undefined: 0 to a negative power', rec='Undefined: 0 to a negative power')
2.5^3 -> iter=15.625 (ok=True), rec=15.625 (ok=True)
2.5^-2 -> iter=0.16000000000000003 (ok=True), rec=0.16000000000000003 (ok=True)
2^3.0 -> error (iter='Exponent must be an integer', rec='Exponent must be an integer')
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

REFERENCES

GeeksforGeeks. (2025, July 23). *Merge Sort in Python*. GeeksforGeeks.

https://www.geeksforgeeks.org/python/python-program-for-merge-sort/

Tiwari, A. (2023, February 25). *Divide and conquer algorithms in Python*. DEV Community.

https://dev.to/thegeekyb0y/divide-and-conquer-algorithms-in-python-5gm8

Learn Data Structures and Algorithms with Python: Divide and Conquer Cheatsheet | *Codecademy*. (n.d.). Codecademy.

https://www.codecademy.com/learn/learn-data-structures-and-algorithms-with-python/modules/divide-and-conquer/cheatsheet