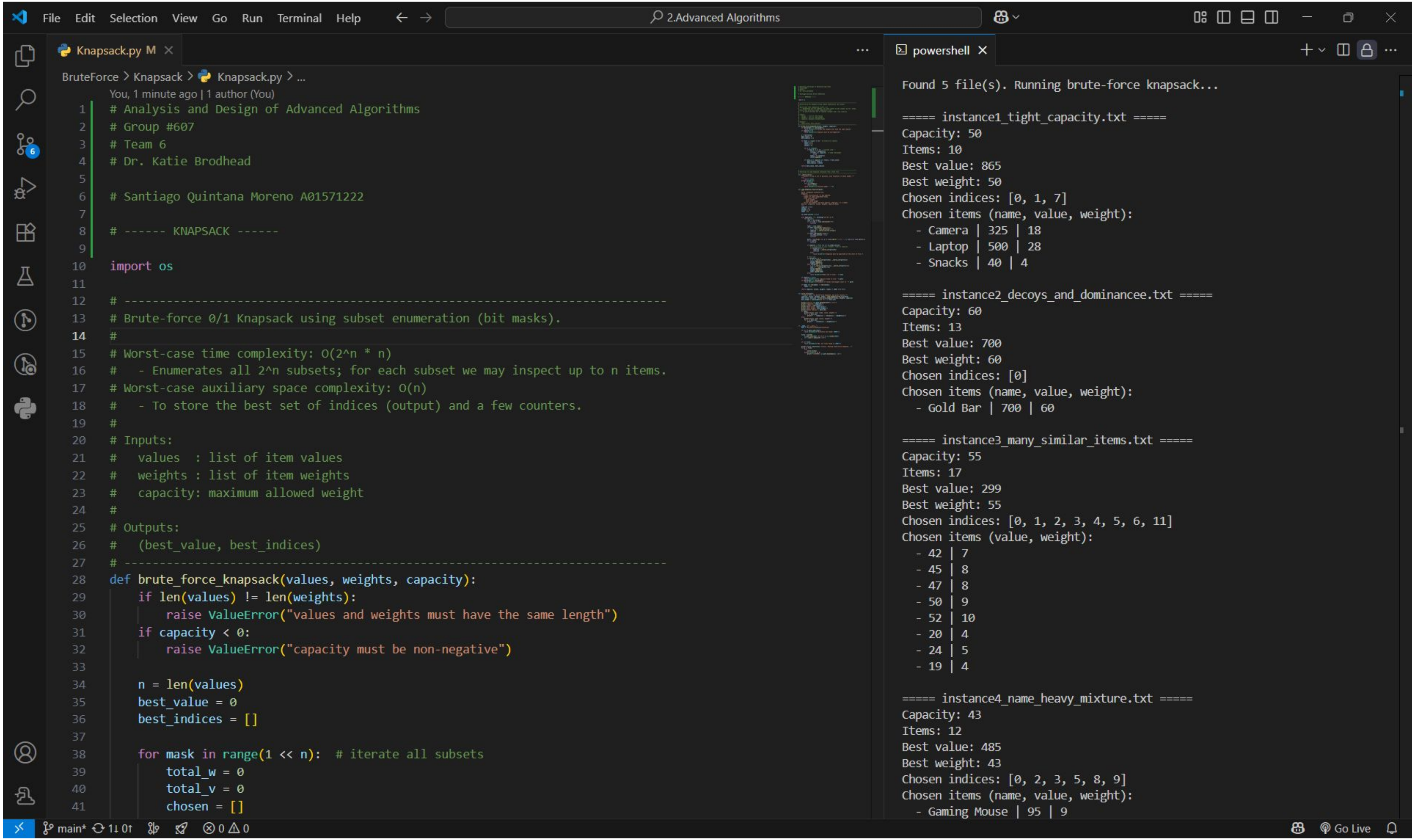
Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Homework 1:
Knapsack, by Brute Force - Individual work

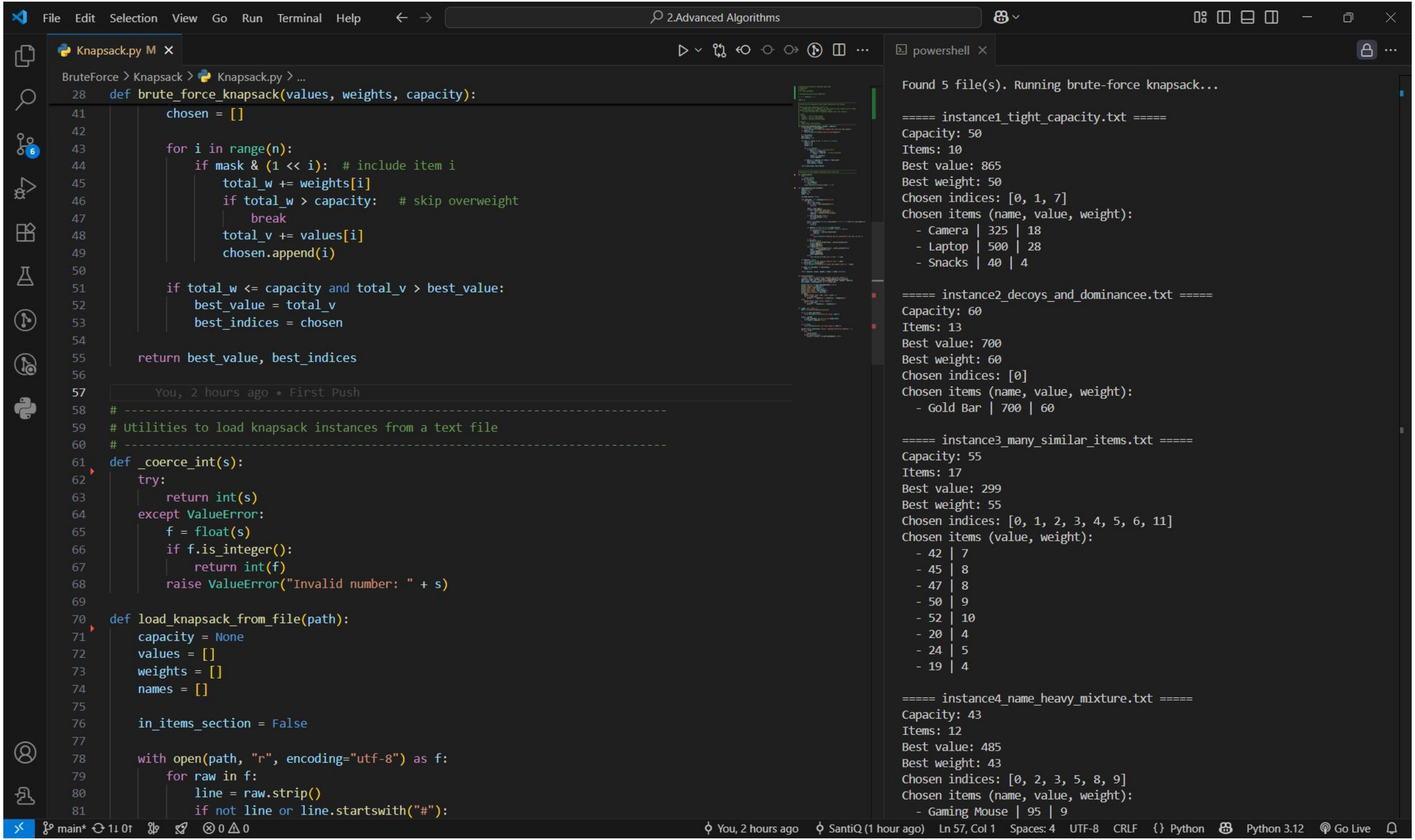
Group: 607
Team #6
Dr. Katie Brodhead

Santiago Quintana Moreno A01571222



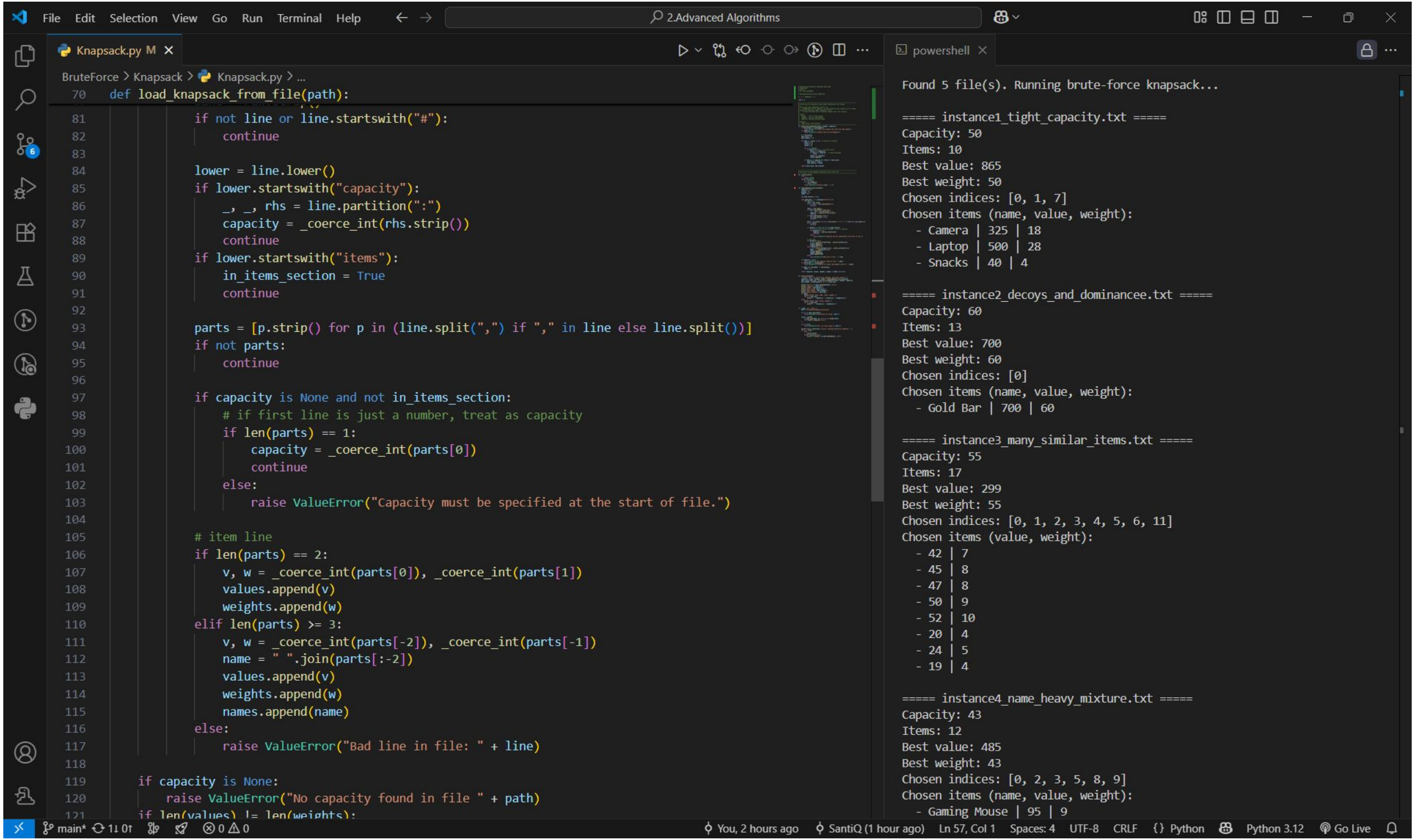
```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 6
# Dr. Katie Brodhead

# Santiago Quintana Moreno A01571222

# ------ KNAPSACK ------

import os

# -----------------------------------------------------------------------------
# Brute-force 0/1 Knapsack using subset enumeration (bit masks).
#
# Worst-case time complexity: O(2^n * n)
#   - Enumerates all 2^n subsets; for each subset we may inspect up to n items.
# Worst-case auxiliary space complexity: O(n)
#   - To store the best set of indices (output) and a few counters.
#
# Inputs:
#   values  : list of item values
#   weights : list of item weights
#   capacity: maximum allowed weight
#
# Outputs:
#   (best_value, best_indices)
# -----------------------------------------------------------------------------
def brute_force_knapsack(values, weights, capacity):
    if len(values) != len(weights):
        raise ValueError("values and weights must have the same length")
    if capacity < 0:
        raise ValueError("capacity must be non-negative")

    n = len(values)
    best_value = 0
    best_indices = []

    for mask in range(1 << n):  # iterate all subsets
        total_w = 0
        total_v = 0
        chosen = []
```

```
Found 5 file(s). Running brute-force knapsack...

===== instance1_tight_capacity.txt =====
Capacity: 50
Items: 10
Best value: 865
Best weight: 50
Chosen indices: [0, 1, 7]
Chosen items (name, value, weight):
  - Camera | 325 | 18
  - Laptop | 500 | 28
  - Snacks | 40 | 4

===== instance2_decoys_and_dominancee.txt =====
Capacity: 60
Items: 13
Best value: 700
Best weight: 60
Chosen indices: [0]
Chosen items (name, value, weight):
  - Gold Bar | 700 | 60

===== instance3_many_similar_items.txt =====
Capacity: 55
Items: 17
Best value: 299
Best weight: 55
Chosen indices: [0, 1, 2, 3, 4, 5, 6, 11]
Chosen items (value, weight):
  - 42 | 7
  - 45 | 8
  - 47 | 8
  - 50 | 9
  - 52 | 10
  - 20 | 4
  - 24 | 5
  - 19 | 4

===== instance4_name_heavy_mixture.txt =====
Capacity: 43
Items: 12
Best value: 485
Best weight: 43
Chosen indices: [0, 2, 3, 5, 8, 9]
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
```



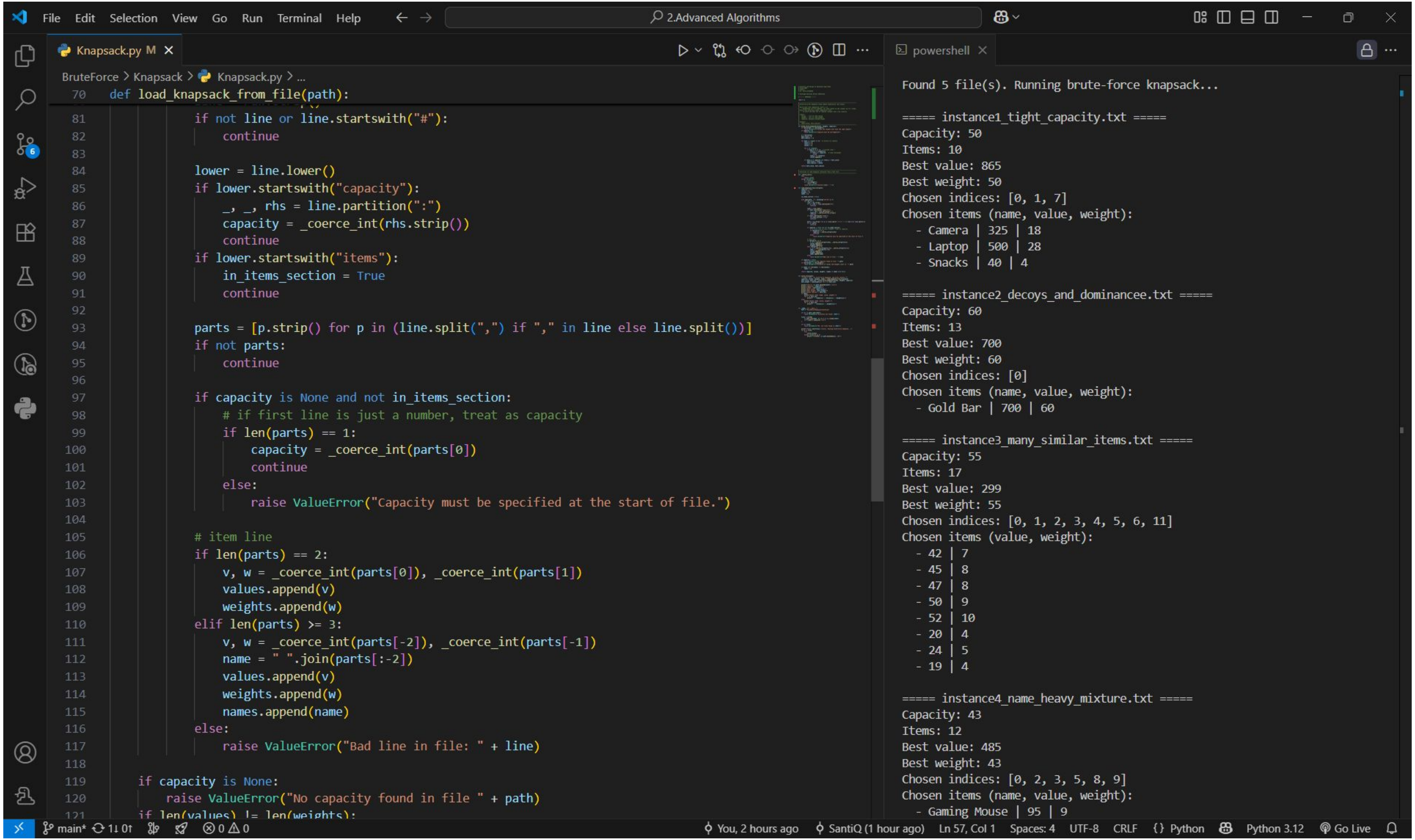
```python
def brute_force_knapsack(values, weights, capacity):
        chosen = []

        for i in range(n):
            if mask & (1 << i):  # include item i
                total_w += weights[i]
                if total_w > capacity:   # skip overweight
                    break
                total_v += values[i]
                chosen.append(i)

        if total_w <= capacity and total_v > best_value:
            best_value = total_v
            best_indices = chosen

    return best_value, best_indices

      You, 2 hours ago • First Push
# ---------------------------------------------------------------------------
# Utilities to load knapsack instances from a text file
# ---------------------------------------------------------------------------
def _coerce_int(s):
    try:
        return int(s)
    except ValueError:
        f = float(s)
        if f.is_integer():
            return int(f)
        raise ValueError("Invalid number: " + s)

def load_knapsack_from_file(path):
    capacity = None
    values = []
    weights = []
    names = []

    in_items_section = False

    with open(path, "r", encoding="utf-8") as f:
        for raw in f:
            line = raw.strip()
            if not line or line.startswith("#"):
```

```
Found 5 file(s). Running brute-force knapsack...

===== instance1_tight_capacity.txt =====
Capacity: 50
Items: 10
Best value: 865
Best weight: 50
Chosen indices: [0, 1, 7]
Chosen items (name, value, weight):
  - Camera | 325 | 18
  - Laptop | 500 | 28
  - Snacks | 40 | 4

===== instance2_decoys_and_dominancee.txt =====
Capacity: 60
Items: 13
Best value: 700
Best weight: 60
Chosen indices: [0]
Chosen items (name, value, weight):
  - Gold Bar | 700 | 60

===== instance3_many_similar_items.txt =====
Capacity: 55
Items: 17
Best value: 299
Best weight: 55
Chosen indices: [0, 1, 2, 3, 4, 5, 6, 11]
Chosen items (value, weight):
  - 42 | 7
  - 45 | 8
  - 47 | 8
  - 50 | 9
  - 52 | 10
  - 20 | 4
  - 24 | 5
  - 19 | 4

===== instance4_name_heavy_mixture.txt =====
Capacity: 43
Items: 12
Best value: 485
Best weight: 43
Chosen indices: [0, 2, 3, 5, 8, 9]
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
```



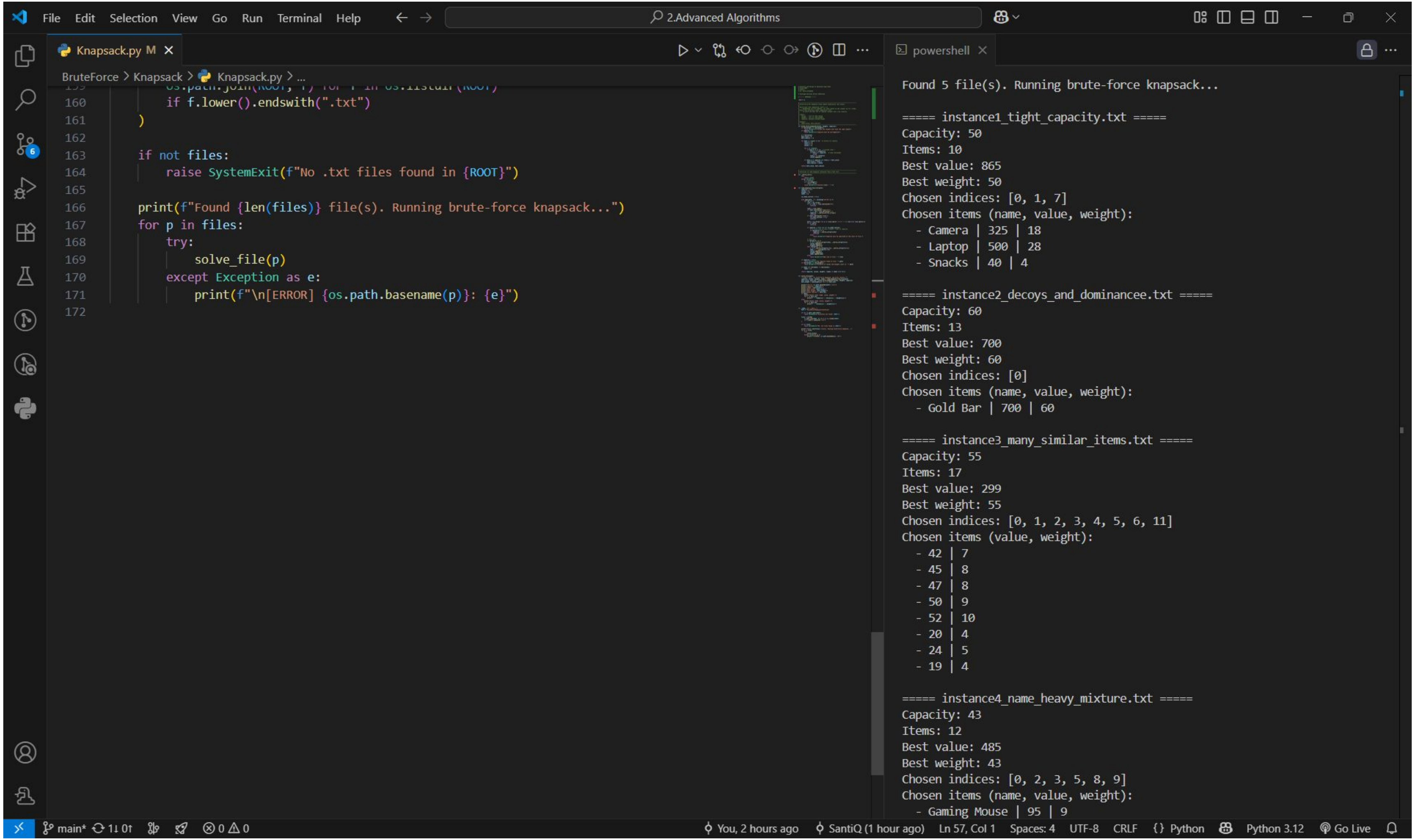
```
def load_knapsack_from_file(path):
            if not line or line.startswith("#"):
                continue

            lower = line.lower()
            if lower.startswith("capacity"):
                _, _, rhs = line.partition(":")
                capacity = _coerce_int(rhs.strip())
                continue
            if lower.startswith("items"):
                in_items_section = True
                continue

            parts = [p.strip() for p in (line.split(",") if "," in line else line.split())]
            if not parts:
                continue

            if capacity is None and not in_items_section:
                # if first line is just a number, treat as capacity
                if len(parts) == 1:
                    capacity = _coerce_int(parts[0])
                    continue
                else:
                    raise ValueError("Capacity must be specified at the start of file.")

            # item line
            if len(parts) == 2:
                v, w = _coerce_int(parts[0]), _coerce_int(parts[1])
                values.append(v)
                weights.append(w)
            elif len(parts) >= 3:
                v, w = _coerce_int(parts[-2]), _coerce_int(parts[-1])
                name = " ".join(parts[:-2])
                values.append(v)
                weights.append(w)
                names.append(name)
            else:
                raise ValueError("Bad line in file: " + line)

    if capacity is None:
        raise ValueError("No capacity found in file " + path)
    if len(values) != len(weights):
```

```
Found 5 file(s). Running brute-force knapsack...

===== instance1_tight_capacity.txt =====
Capacity: 50
Items: 10
Best value: 865
Best weight: 50
Chosen indices: [0, 1, 7]
Chosen items (name, value, weight):
  - Camera | 325 | 18
  - Laptop | 500 | 28
  - Snacks | 40 | 4

===== instance2_decoys_and_dominancee.txt =====
Capacity: 60
Items: 13
Best value: 700
Best weight: 60
Chosen indices: [0]
Chosen items (name, value, weight):
  - Gold Bar | 700 | 60

===== instance3_many_similar_items.txt =====
Capacity: 55
Items: 17
Best value: 299
Best weight: 55
Chosen indices: [0, 1, 2, 3, 4, 5, 6, 11]
Chosen items (value, weight):
  - 42 | 7
  - 45 | 8
  - 47 | 8
  - 50 | 9
  - 52 | 10
  - 20 | 4
  - 24 | 5
  - 19 | 4

===== instance4_name_heavy_mixture.txt =====
Capacity: 43
Items: 12
Best value: 485
Best weight: 43
Chosen indices: [0, 2, 3, 5, 8, 9]
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
```



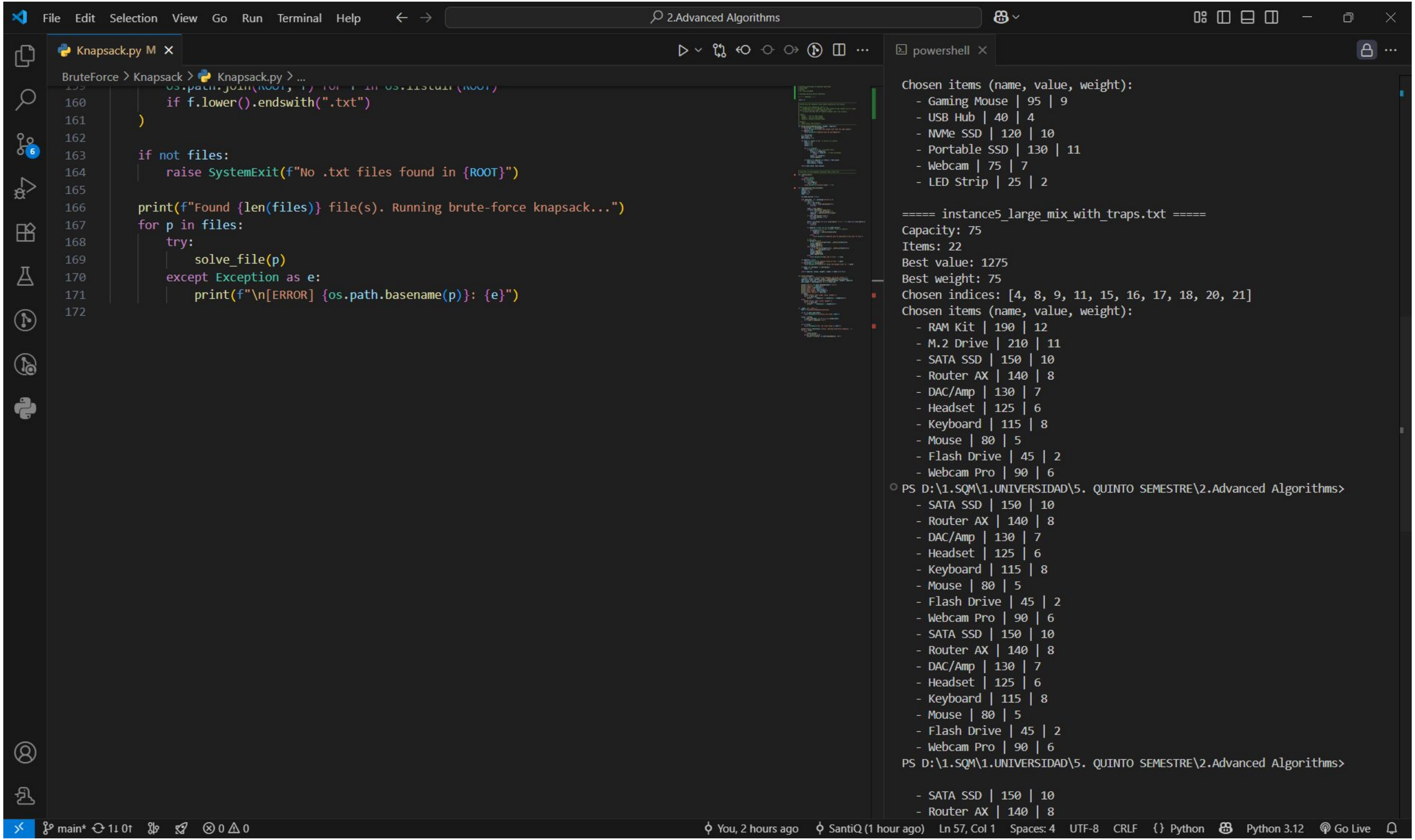
```python
def load_knapsack_from_file(path):
            if not line or line.startswith("#"):
                continue

            lower = line.lower()
            if lower.startswith("capacity"):
                _, _, rhs = line.partition(":")
                capacity = _coerce_int(rhs.strip())
                continue
            if lower.startswith("items"):
                in_items_section = True
                continue

            parts = [p.strip() for p in (line.split(",") if "," in line else line.split())]
            if not parts:
                continue

            if capacity is None and not in_items_section:
                # if first line is just a number, treat as capacity
                if len(parts) == 1:
                    capacity = _coerce_int(parts[0])
                    continue
                else:
                    raise ValueError("Capacity must be specified at the start of file.")

            # item line
            if len(parts) == 2:
                v, w = _coerce_int(parts[0]), _coerce_int(parts[1])
                values.append(v)
                weights.append(w)
            elif len(parts) >= 3:
                v, w = _coerce_int(parts[-2]), _coerce_int(parts[-1])
                name = " ".join(parts[:-2])
                values.append(v)
                weights.append(w)
                names.append(name)
            else:
                raise ValueError("Bad line in file: " + line)

    if capacity is None:
        raise ValueError("No capacity found in file " + path)
    if len(values) != len(weights):
```

```
Found 5 file(s). Running brute-force knapsack...

===== instance1_tight_capacity.txt =====
Capacity: 50
Items: 10
Best value: 865
Best weight: 50
Chosen indices: [0, 1, 7]
Chosen items (name, value, weight):
  - Camera | 325 | 18
  - Laptop | 500 | 28
  - Snacks | 40 | 4

===== instance2_decoys_and_dominancee.txt =====
Capacity: 60
Items: 13
Best value: 700
Best weight: 60
Chosen indices: [0]
Chosen items (name, value, weight):
  - Gold Bar | 700 | 60

===== instance3_many_similar_items.txt =====
Capacity: 55
Items: 17
Best value: 299
Best weight: 55
Chosen indices: [0, 1, 2, 3, 4, 5, 6, 11]
Chosen items (value, weight):
  - 42 | 7
  - 45 | 8
  - 47 | 8
  - 50 | 9
  - 52 | 10
  - 20 | 4
  - 24 | 5
  - 19 | 4

===== instance4_name_heavy_mixture.txt =====
Capacity: 43
Items: 12
Best value: 485
Best weight: 43
Chosen indices: [0, 2, 3, 5, 8, 9]
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
```



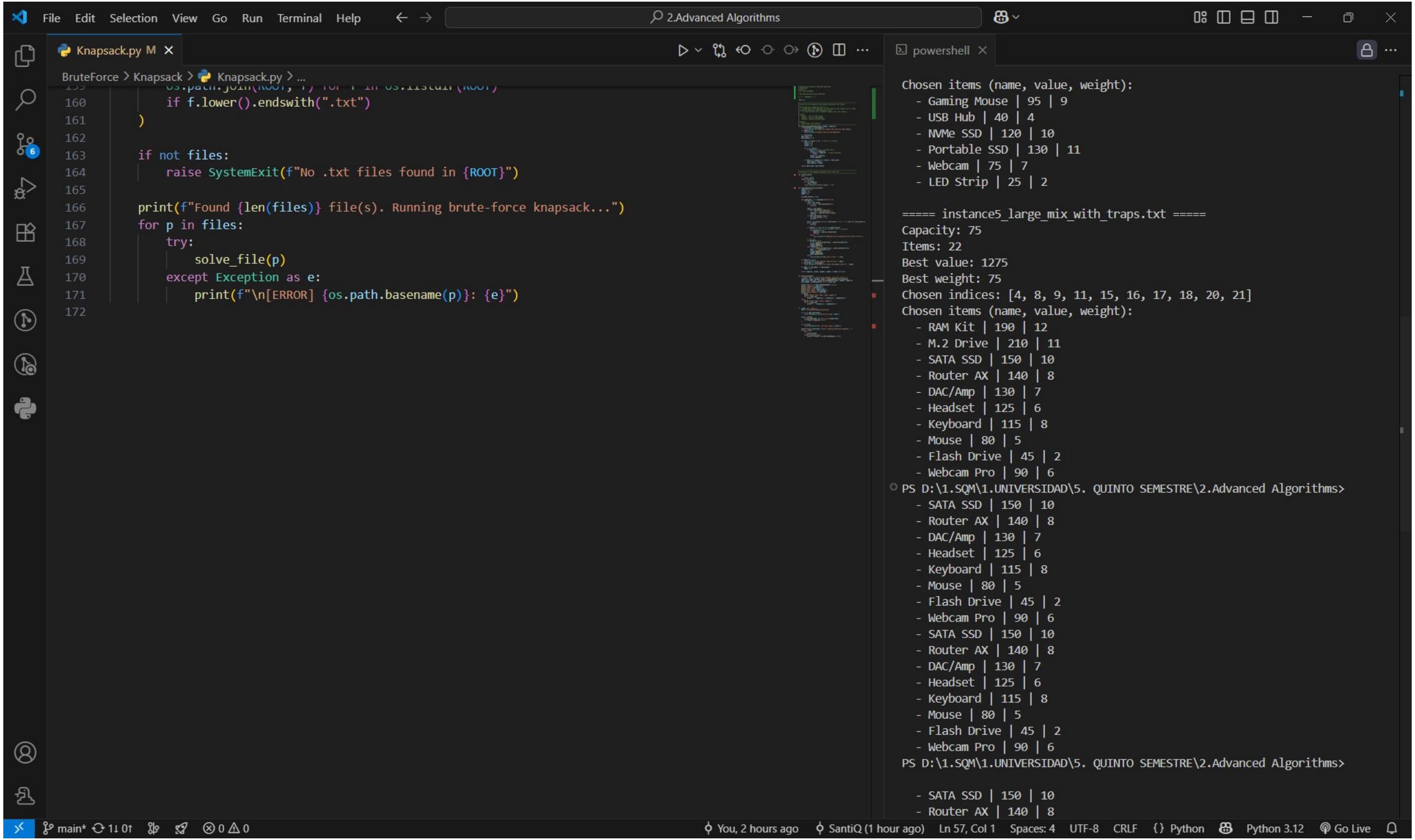
```python
        os.path.join(ROOT, f) for f in os.listdir(ROOT)
        if f.lower().endswith(".txt")
    )

    if not files:
        raise SystemExit(f"No .txt files found in {ROOT}")

    print(f"Found {len(files)} file(s). Running brute-force knapsack...")
    for p in files:
        try:
            solve_file(p)
        except Exception as e:
            print(f"\n[ERROR] {os.path.basename(p)}: {e}")

```

```
Found 5 file(s). Running brute-force knapsack...

===== instance1_tight_capacity.txt =====
Capacity: 50
Items: 10
Best value: 865
Best weight: 50
Chosen indices: [0, 1, 7]
Chosen items (name, value, weight):
  - Camera | 325 | 18
  - Laptop | 500 | 28
  - Snacks | 40 | 4

===== instance2_decoys_and_dominancee.txt =====
Capacity: 60
Items: 13
Best value: 700
Best weight: 60
Chosen indices: [0]
Chosen items (name, value, weight):
  - Gold Bar | 700 | 60

===== instance3_many_similar_items.txt =====
Capacity: 55
Items: 17
Best value: 299
Best weight: 55
Chosen indices: [0, 1, 2, 3, 4, 5, 6, 11]
Chosen items (value, weight):
  - 42 | 7
  - 45 | 8
  - 47 | 8
  - 50 | 9
  - 52 | 10
  - 20 | 4
  - 24 | 5
  - 19 | 4

===== instance4_name_heavy_mixture.txt =====
Capacity: 43
Items: 12
Best value: 485
Best weight: 43
Chosen indices: [0, 2, 3, 5, 8, 9]
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
```



```python
        os.path.join(ROOT, f) for f in os.listdir(ROOT)
        if f.lower().endswith(".txt")
    )

    if not files:
        raise SystemExit(f"No .txt files found in {ROOT}")

    print(f"Found {len(files)} file(s). Running brute-force knapsack...")
    for p in files:
        try:
            solve_file(p)
        except Exception as e:
            print(f"\n[ERROR] {os.path.basename(p)}: {e}")
```

```
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
  - USB Hub | 40 | 4
  - NVMe SSD | 120 | 10
  - Portable SSD | 130 | 11
  - Webcam | 75 | 7
  - LED Strip | 25 | 2

===== instance5_large_mix_with_traps.txt =====
Capacity: 75
Items: 22
Best value: 1275
Best weight: 75
Chosen indices: [4, 8, 9, 11, 15, 16, 17, 18, 20, 21]
Chosen items (name, value, weight):
  - RAM Kit | 190 | 12
  - M.2 Drive | 210 | 11
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
```



```python
            os.path.join(ROOT, f) for f in os.listdir(ROOT)
            if f.lower().endswith(".txt")
        )

    if not files:
        raise SystemExit(f"No .txt files found in {ROOT}")

    print(f"Found {len(files)} file(s). Running brute-force knapsack...")
    for p in files:
        try:
            solve_file(p)
        except Exception as e:
            print(f"\n[ERROR] {os.path.basename(p)}: {e}")
```

```
Chosen items (name, value, weight):
  - Gaming Mouse | 95 | 9
  - USB Hub | 40 | 4
  - NVMe SSD | 120 | 10
  - Portable SSD | 130 | 11
  - Webcam | 75 | 7
  - LED Strip | 25 | 2

===== instance5_large_mix_with_traps.txt =====
Capacity: 75
Items: 22
Best value: 1275
Best weight: 75
Chosen indices: [4, 8, 9, 11, 15, 16, 17, 18, 20, 21]
Chosen items (name, value, weight):
  - RAM Kit | 190 | 12
  - M.2 Drive | 210 | 11
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
  - DAC/Amp | 130 | 7
  - Headset | 125 | 6
  - Keyboard | 115 | 8
  - Mouse | 80 | 5
  - Flash Drive | 45 | 2
  - Webcam Pro | 90 | 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

  - SATA SSD | 150 | 10
  - Router AX | 140 | 8
```