In [22]:
```
################################################################################
# Author:      Syed Kazim Ali
# MatNr:       12407686
# File:        assignment1.ipynb
# Description: Implementation of assignment 1 solutions, including functions
#              for decoding messages, filtering trails, and other exercises
#              as required by the course.
# Comments:    This file contains all required functions and test cases.
#              Each function is documented with docstrings and examples.
#              Noise removal, code word replacement, and data filtering
#              functionalities are implemented and tested.
################################################################################
```

Assignment 1 (15 P)

- "Basic Programming Concepts" -

Imports

In [5]:
```
# Import your packages here:
import math
```

1. Functions

1.1 Introduce yourself

Required Function: `say_hi`

In [3]:
```
# Write your code in this cell:
def say_hi():
    print("Hi, I'm Syed Kazim Ali.")
```

In [4]:
```
# Test your code in this cell:
say_hi()
```

```
Hi, I'm Syed Kazim Ali.
```

1.2 Calculating the area of a circle

Required Function: `calculate_area_of_circle`

In [5]:
```
# Write your code in this cell:
import math

def calculate_area_of_circle(radius):
    if radius <= 0:
```

```
        return None
    return math.pi * radius ** 2
```

In [7]:
```
# Test your code in this cell:
print(calculate_area_of_circle(5))
```

```
78.53981633974483
```

2. Conditional Statements

2.1 Calculate a student's Informatics 1 grade

Required Function: `calculate_grade`

In [8]:
```
# Write your code in this cell:
def calculate_grade(points):
    if not isinstance(points, (int, float)):
        return None
    if points < 0 or points > 100:
        return None
    if points >= 90:
        return 1
    elif points >= 80:
        return 2
    elif points >= 65:
        return 3
    elif points >= 50:
        return 4
    else:
        return 5
```

In [9]:
```
# Test your code in this cell:
print(calculate_grade(95))
print(calculate_grade(83))
print(calculate_grade(70))
print(calculate_grade(55))
print(calculate_grade(40))
```

```
1
2
3
4
5
```

3. Sequences and Sets

3.1 Wine barrel analysis

Required Function: `check_wine_barrels`

In [10]:
```
# Write your code in this cell:
def check_wine_barrels(barrels):
```

```
    if not barrels:
        return (0, 0, 0)
    total = len(barrels)
    avg = round(sum(barrels) / total, 1)
    above_400 = sum(1 for b in barrels if b > 400)
    return (total, avg, above_400)
```

In [11]:
```
# Test your code in this cell:
barrels = [356.7, 402.3, 298.4, 450.1, 389.2]
print(check_wine_barrels(barrels))
```

```
(5, 379.3, 2)
```

3.2 Tailoring

Required Function: `dressmaking_orders`

In [12]:
```
# Write your code in this cell:
def dressmaking_orders(items):
    unique_items = set(items)
    print(f"Total unique orders: {len(unique_items)}")
    return unique_items
```

In [13]:
```
# Test your code in this cell:
items_example = ['dress', 'hat', 'shirt', 'socks', 'apron', 'shirt', 'jacket', 'sca
prices_example = [152.51, 34.99, 97.94, 22.79, 27.89, 30.91, 213.79, 24.99, 77.54]
status_example = ['sold', 'reserved', 'sold', 'in stock', 'sold', 'sold', 'in stock
print(dressmaking_orders(items_example))
print(dressmaking_orders(prices_example))
print(dressmaking_orders(status_example))
```

```
Total unique orders: 8
{'shirt', 'dress', 'apron', 'scarf', 'socks', 'pants', 'hat', 'jacket'}
Total unique orders: 9
{97.94, 34.99, 77.54, 213.79, 22.79, 152.51, 24.99, 27.89, 30.91}
Total unique orders: 3
{'sold', 'in stock', 'reserved'}
```

3.3 Ski trips

Required Function: `ski_compare`

In [14]:
```
# Write your code in this cell:
def ski_compare(austria, switzerland):
    return {
        "both": austria & switzerland,
        "only_austria": austria - switzerland,
        "only_switzerland": switzerland - austria
    }
```

In [15]:
```
# Test your code in this cell:
austria = {'Lea', 'Jonas', 'Marlene', 'Sven', 'Tom'}
```

```
switzerland = {'Marlene', 'Sven', 'Urs', 'Valeria', 'Yannick'}
print(ski_compare(austria, switzerland))
```

```
{'both': {'Sven', 'Marlene'}, 'only_austria': {'Tom', 'Lea', 'Jonas'}, 'only_switzer
land': {'Valeria', 'Urs', 'Yannick'}}
```

4. Dictionaries

4.1 Hiking trails

Required Function: `find_trails`

In [16]:
```
# Write your code in this cell:
def find_trails(trails, max_length, max_difficulty):
    """
    Returns a dictionary of trails with length <= max_length and difficulty <= max_
    """
    return {name: info for name, info in trails.items()
            if info['length'] <= max_length and info['difficulty'] <= max_difficult
```

In [17]:
```
# Test your code in this cell:
trails = {
    'Alpine trail': {'length': 12, 'difficulty': 3},
    'Lake circuit': {'length': 8, 'difficulty': 2},
    'Drachenwand via ferrata': {'length': 5, 'difficulty': 5},
    'Forest path': {'length': 3, 'difficulty': 1},
    'Summit route': {'length': 14, 'difficulty': 4}
}
max_length = 10
max_difficulty = 2

result = find_trails(trails, max_length, max_difficulty)
print(result)
```

```
{'Lake circuit': {'length': 8, 'difficulty': 2}, 'Forest path': {'length': 3, 'diffi
culty': 1}}
```

5.1 Strings

5.1 Tutor's Secret Message

Required Function: `decode_tutor_message`

In [18]:
```
# Write your code in this cell:
def decode_tutor_message(encoded_lines, code_words, noise_symbols):
    decoded_lines = []
    for line in encoded_lines:
        # Remove noise symbols
        for symbol in noise_symbols:
            line = line.replace(symbol, "")

        # Replace code words
```

```
        words = line.split()
        decoded_words = [code_words.get(word, word) for word in words]
        decoded_lines.append(" ".join(decoded_words))

    return decoded_lines
```

In [21]:
```
# Test your function
encoded_lines = [
    "h<ey# c0de~r, yoo di+d a g+r8 j0+b",
    "thx 4 st@y!ng m#0t!v@ted ~t!ll t<he e~n+d",
    "t+y f0+r a#~l++<l y0ur w0#rk"
]
code_words = {
    "c0der,": "coder,",
    "yoo": "you",
    "gr8": "great",
    "j0b": "job",
    "thx": "thanks",
    "4": "for",
    "st@y!ng": "staying",
    "m0t!v@ted": "motivated",
    "t!ll": "till",
    "ty": "thanks",
    "f0r": "for",
    "y0ur": "your",
    "w0rk": "work"
}
noise_symbols = ("#", "~", "+", "<")

decoded = decode_tutor_message(encoded_lines, code_words, noise_symbols)
for line in decoded:
    print(line)
```

```
hey coder, you did a great job
thanks for staying motivated till the end
thanks for all your work
```