I. Docstrings.

1. Що таке **docstrings?** Навіщо вони потрібні**?**

Докстрінг **-** це опис модуля/функції/класу**,** який розміщується в перших рядках модуля/функції/класу**.** Докстрінг допомагає іншим розробникам **(**або і власне розробнику**)** зрозуміти**,** що робить модуль/функція/клас**,** які параметри приймає**,** які значення повертає**,** які винятки можуть виникнути**,** інші нюанси використання в або поза певного контексту**.**

1. Створіть **docstrings** для функції**,** що розраховує суму двох цілих чисел та повертає їхню суму**.**

In [ ]:

```
"""
This function adds two numbers and returns their sum.

Parameters:
a (int): The first number
b (int): The second number

Returns:
int: The sum of the first and second numbers
"""
```

1. Визначте функцію**,** що розраховує суму двох цілих чисел та повертає їхню суму**,** використовуючи документацію з завдання **2.**

In [1]:

```
def sum(a, b):
    """
    This function adds two numbers and returns their sum.

    Parameters:
    a (int): The first number
    b (int): The second number

    Returns:
    int: The sum of the first and second numbers
    """
    return a + b
```

1. Створіть **docstring** для функції**,** що перевіряє**,** чи є число простим**.** Увага**:** під час виклику функції ми очікуємо**,** що користувач**(**ка**)** може подати на вхід будь**-**який тип об'єкту**.** У випадку**,** якщо на вході не підходящий тип даних**,** маємо кинути помилку**.**

In [ ]:

```
"""
Checks if a number is prime.

Parameters:
n (int): The number to check.

Returns:
bool: True if the number is prime, otherwise False.

Raises:
TypeError: If the input is not an integer.
"""
```

1. Визначте функцію, яка перевіряє, чи є число простим, використовуючи документацію з завдання **4.**

In [4]:

```
def is_prime(n):
    """
    Checks if a number is prime.

    Parameters:
    n (int): The number to check.

    Returns:
    bool: True if the number is prime, otherwise False.

    Raises:
    TypeError: If the input is not an integer.
    """
    if not isinstance(n, int):
        raise TypeError("Input must be an integer")
    if n < 2:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

print(is_prime(10))

print(is_prime(5))
print(is_prime(7))
print(is_prime(29))
```

```
False
True
True
True
```

1. Створіть **docstring** для функції, що повертає найдовший спільний префікс двох рядків не залежно від регістру. Обов'язково включіть два приклади згідно формату.

In [ ]:

```
"""
This function returns the longest common prefix of two strings regardless of case.

Parameters:
str1 (str): The first string
str2 (str): The second string

Returns:
str: The longest common prefix of str1 and str2

Example:
>>> longest_common_prefix('Python', 'pylanse')
'py'

>>> longest_common_prefix('DistEdu', 'distance')
'dist'
"""
```

1. Реалізуйте функцію згідно документації із завдання **6,** використовуючи її.

In [5]:

```
def longest_common_prefix(str1, str2):
    """
```

```
    This function returns the longest common prefix of two strings regardless of case.

    Parameters:
    str1 (str): The first string
    str2 (str): The second string

    Returns:
    str: The longest common prefix of str1 and str2

    Example:
    >>> longest_common_prefix('Python', 'pylanse')
    'py'

    >>> longest_common_prefix('DistEdu', 'distance')
    'dist'
    """
    str1 = str1.lower()
    str2 = str2.lower()
    prefix = ''
    for i in range(min(len(str1), len(str2))):
        if str1[i] == str2[i]:
            prefix += str1[i]
        else:
            break
    return prefix

print(longest_common_prefix('Python', 'python'))
print(longest_common_prefix('DistEdu', 'distance'))
```

```
python
dist
```

1. Створіть **docstring** для функції, що розраховує суму щомісячного платежу по єОселі.

In [ ]:

```
"""
This function returns the monthly due payment for EOselya estate loan.

Parameters:
price (int): The total price of the property
duration (int): The duration of the loan in months
interest (float): The annual real interest rate

Returns:
float: The monthly due payment

Note:
Does not account for down payment or other fees
"""
```

1. Використовуючи документацію із завдання **8,** реалізуйте відповідну функцію.

In [12]:

```
def monthly_due(price, duration, interest):
    """
    This function returns the monthly due payment for EOselya estate loan.

    Parameters:
    price (int): The total price of the property
    duration (int): The duration of the loan in months
    interest (float): The annual real interest rate

    Returns:
    float: The monthly due payment

    Note:
    Does not account for down payment or other fees
```

```
    """
    down_payment = 0.2 * price
    loan = price - down_payment
    monthly_interest = interest / 12
    return round(loan * monthly_interest / (1 - (1 + monthly_interest) ** -duration), 2)

print(monthly_due(1_620_255, 12, 0.0394))
```

```
110336.12
```

II. Errors and exceptions handling.

1. Реалізуйте функцію, яка отримує значення зі словника за ключем та повертає його, а за умови відсутності ключа відловлює **KeyError.**

In [15]:

```
def get_value_from_dict(dictionary, key):
    try:
        return dictionary[key]
    except KeyError:
        print("The key is not present in the dictionary.")
        return None

print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'b'))
print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'd'))
```

```
2
The key is not present in the dictionary.
None
```

1. Реалізуйте функцію із завдання **10** без відловлювання помилки, однак за допомогою конструкції **if-else.**

In [14]:

```
def get_value_from_dict(dictionary, key):
    if key in dictionary:
        return dictionary[key]
    else:
        print("The key is not present in the dictionary.")
        return None

print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'b'))
print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'd'))
```

```
2
The key is not present in the dictionary.
None
```

1. Реалізуйте функцію з завдання **10,** використовуючи метод словника, який дозволяє зробити операцію навіть без **if-else.**

In [16]:

```
def get_value_from_dict(dictionary, key):
    return dictionary.get(key, None)

print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'b'))
print(get_value_from_dict({'a': 1, 'b': 2, 'c': 3}, 'd'))
```

```
2
None
```

1. Визначте функцію, яка обробляє значення, введене користувачем з терміналу, приводячи його до цілого числа. Використайте відловлення відповідної помилки. Наприклад, людина має ввести свій вік.

In [19]:

```
def get_user_age():
    try:
        user_input = int(input("Please enter your age: "))
        return user_input
    except ValueError:
        print("ValueError: Invalid input. Please enter a number.")
        return None

print(get_user_age())
```

```
55645
```

1. Визначте функцію з завдання **13,** де використайте конструкцію **if-else** замість відловдення помилки**.**

In [22]:

```
def get_user_age():
    user_input = input("Please enter your age: ")
    if user_input.isdigit():
        return int(user_input)
    else:
        print("ValueError: Invalid input. Please enter a number.")
        return None

print(get_user_age())
```

```
ValueError: Invalid input. Please enter a number.
None
```

1. Визначте функцію з завдання **13,** враховуючи додаткове обмеження**,** що число має бути у діапазоні від **18** до **120 (**повнолітня людина**).** Відловіть відповідні типи помилок**.**

In [23]:

```
def get_user_age():
    try:
        user_input = int(input("Please enter your age: "))
        if user_input < 18 or user_input > 120:
            print("ValueError: Invalid input. Please enter a number between 18 and 120."
)
            return None
        return user_input
    except ValueError:
        print("ValueError: Invalid input. Please enter a number.")
        return None

print(get_user_age())
```

```
ValueError: Invalid input. Please enter a number between 18 and 120.
None
```

1. Напишіть код**,** який би генерував максимальну кількість помилок різного типу**.**

In [28]:

```
age_plus7 = int(input("Please enter your age: ")) + "7"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[28], line 1
----> 1 age_plus7 = int(input("Please enter your age: ")) + "7"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [29]:

```
ages = [12, 18, 24, 30, 36, 42, 48, 54, 60, 66]
print(ages[23])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[29], line 2
      1 ages = [12, 18, 24, 30, 36, 42, 48, 54, 60, 66]
----> 2 print(ages[23])

IndexError: list index out of range
```

In [30]:

```
print(invalid_name_ages)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 print(invalid_name_ages)

NameError: name 'invalid_name_ages' is not defined
```

1. Наскільки це можливо, виправте код із завдання **16.**

In [31]:

```
age_plus7 = int(input("Please enter your age: ")) + 7
ages = [12, 18, 24, 30, 36, 42, 48, 54, 60, 66]
ages.append(age_plus7)
print(ages[2:3])
print(ages)
```

```
[24]
[12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 73]
```

Вітаю! Ви велика(ий) молодець, що впоралась(вся). Похваліть себе та побалуйте чимось приємним. Я Вами пишаюся.