

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 2
з навчальної дисципліни
“Скриптові мови програмування (Python)”

ФУНКЦІЇ PYTHON

ВИКОНАВ
студент академічної групи КН-24
Мироненко Я.М

ПЕРЕВІРИВ
асистент кафедри кібербезпеки та
програмного забезпечення
Ткаченко О.С

Тема: Функції у Python

Мета: навчитися створювати власні функції у мові Python

Варіант 3

Розробіть наступні функції:

1. Функцію, що приймає 1 аргумент – номер місяця (від 1 до 12), і повертає пору року, якому цей місяць належить (зима, весна, літо чи осінь);
2. Функцію, яка приймає 1 аргумент – номер року, і повертає True, якщо рік високосний, і False інакше. Примітка: Будь-який рік, який ділиться на 4 без залишку, є високосним роком: наприклад, 1988, 1992 та 1996 роки є високосними;
3. Функцію, що приймає 3 аргументи – день, місяць та рік. Повернути True, якщо така дата існує, та False – якщо такої дати не буває (наприклад 15-тий місяць);
4. Функцію, що як аргумент отримує список дат у форматі dd.mm.yyyy (наприклад, ["11.12.1877", "25.01.2022", "01.05.2023"]) і шукає у ньому повтори, якщо повтори є – повертає список значень, що повторюються;
5. Функцію, у якої немає аргументів, а повертає вона випадкову дату у форматі dd.mm.yyyy. Примітка: Для генерації випадкових чисел використати бібліотеку random та, наприклад, її функцію randint;
6. Функцію, що отримує як аргументи дві дати (формат обрати самостійно) та повертає кількість днів між ними. Примітка: Використати бібліотеку datetime та її функцію з такою ж назвою datetime.
7. Функцію, що отримує як аргумент номер місяця, виводить на екран відповідну назву місяця та нічого не повертає.

Помістіть функції в окремий модуль. Реалізуйте програму, яка використовує всі функції зі створеного модуля. Зробіть описи Doc strings для кожної реалізованої функції.

Завдання 1

1. Функцію, що приймає 1 аргумент – номер місяця (від 1 до 12), і повертає пору року, якому цей місяць належить (зима, весна, літо чи осінь)

Лістинг функції

```
def ask_mans(n): # Function 1
    """Функція, яка приймає номер місяця і виводить пору року"""
    if n == 12 or n == 1 or n == 2: # 12, 1, 2 - зима
        print("Пора року: Зима")
    elif 3 <= n <= 5: # 3, 4, 5 - весна
        print("Пора року: Весна")
    elif 6 <= n <= 8: # 6, 7, 8 - літо
        print("Пора року: Літо")
    elif 9 <= n <= 11: # 9, 10, 11 - осінь
        print("Пора року: Осінь")
    else:
        print("Такого місяця не існує") # якщо введено невірне значення
```

Принцип роботи функції

Функція ask_mans(n) визначає пору року за номером місяця n. Вона працює за таким принципом

Якщо n дорівнює 12, 1 або 2, виводить "Зима".

Якщо n знаходиться в діапазоні від 3 до 5, виводить "Весна".

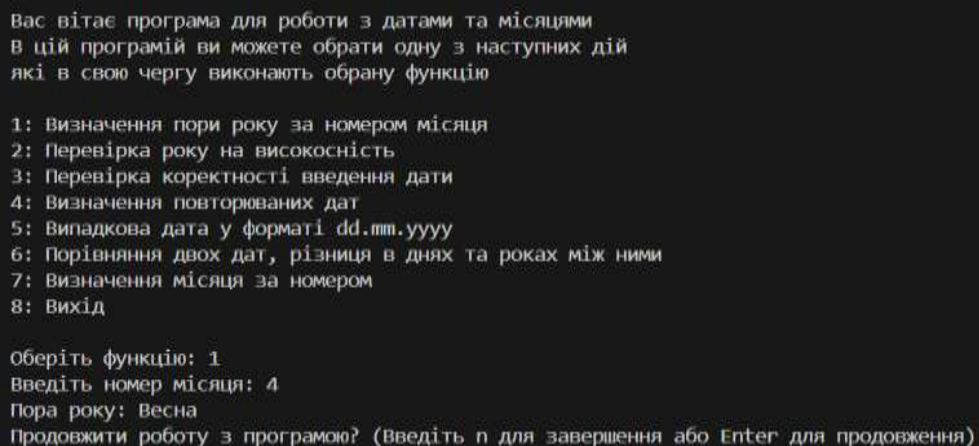
Якщо n знаходиться в діапазоні від 6 до 8, виводить "Літо".

Якщо n знаходиться в діапазоні від 9 до 11, виводить "Осінь".

Якщо n не відповідає жодному з цих діапазонів, виводить "Такого місяця не існує".

Функція використовує умовні оператори (if, elif, else) для перевірки значення n і виведення відповідної пори року.

Тестові запуски функції



```
Вас вітає програма для роботи з датами та місяцями
В цій програмі ви можете обрати одну з наступних дій
які в свою чергу виконують обрану функцію

1: Визначення пори року за номером місяця
2: Перевірка року на високосність
3: Перевірка коректності введення дати
4: Визначення повторюваних дат
5: Випадкова дата у форматі dd.mm.yyyy
6: Порівняння двох дат, різниця в днях та роках між ними
7: Визначення місяця за номером
8: Вихід

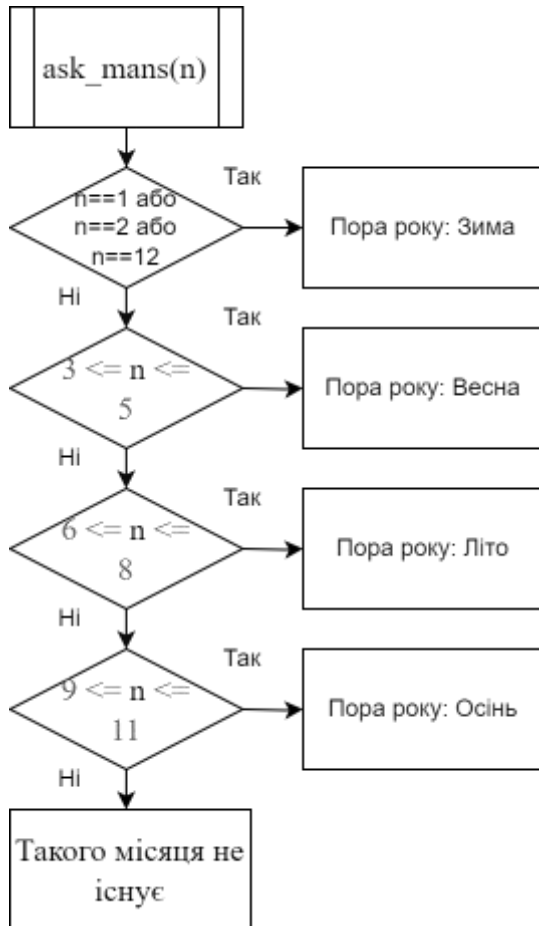
Оберіть функцію: 1
Введіть номер місяця: 4
Пора року: Весна
Продовжити роботу з програмою? (Введіть n для завершення або Enter для продовження)
```

Рисунок 1

```
Оберіть функцію: 1
Введіть номер місяця: 2
Пора року: Зима
Продовжити роботу з програмою? (Введіть n для завершення або Enter для продовження)
```

Рисунок 2

Блок-Схема



Завдання 2

2. Функцію, яка приймає 1 аргумент – номер року, і повертає True, якщо рік високосний, і False інакше. Примітка: Будь-який рік, який ділиться на 4 без залишку, є високосним роком: наприклад, 1988, 1992 та 1996 роки є високосними

Лістинг функції

```
def ask_year(year): # Function 2
    """Функція яка приймає рік і виводить True якщо він високосний та False якщо
    ні"""
    if year % 4 == 0 and year % 100 != 0 or year % 400 == 0: # перевірка чи рік
        високосний
        return True # якщо високосний
    else:
        return False # якщо не високосний
```

Принцип роботи функції

Функція `ask_year(year)` перевіряє, чи є рік `year` високосним, і повертає `True` або `False`.
Рік вважається високосним, якщо: Він ділиться на 4 без остачі ($\text{year} \% 4 == 0$), але не ділиться на 100 без остачі ($\text{year} \% 100 != 0$), або він ділиться на 400 без остачі ($\text{year} \% 400 == 0$).

Якщо умова виконується, функція повертає `True` (високосний рік).

Якщо умова не виконується, функція повертає `False` (не високосний рік).

Тестові запуски функції

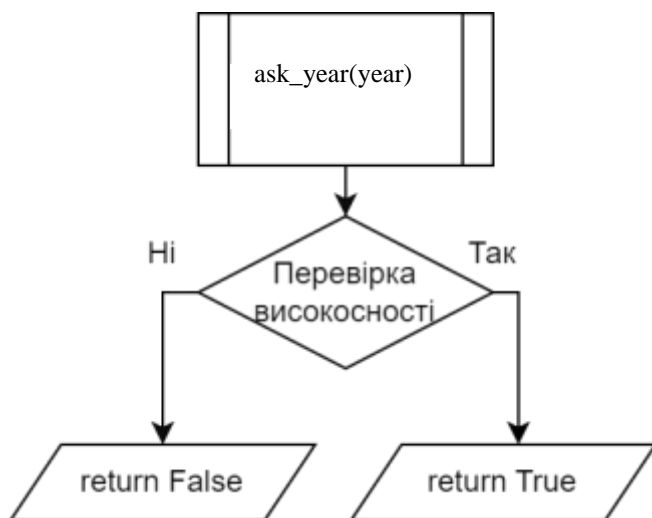
```
Оберіть функцію: 2  
Введіть рік: 2020  
Рік високосний
```

Рисунок 3

```
Оберіть функцію: 2  
Введіть рік: 1234  
Рік не високосний
```

Рисунок 4

Блок-Схема



Завдання 3

3. Функцію, що приймає 3 аргументи - день, місяць та рік. Повернути True, якщо така дата існує, та False - якщо такої дати не буває (наприклад 15-тий місяць)

Лістинг функції

```
def date_correct(day, month, year): # Function 3
    """Функція яка перевіряє правильність введення дати"""

    if year <= 0: # перевірка року
        return False

    if month < 1 or month > 12: # перевірка місяця
        return False

    days_in_month = { # кількість днів у місяці
        1: 31, # Січень
        2: 29 if (year % 4 == 0 and year % 100 != 0 or year % 400 == 0) else 28, #
        3: 31, # Березень
        4: 30, # Квітень
        5: 31, # Травень
        6: 30, # Червень
        7: 31, # Липень
        8: 31, # Серпень
        9: 30, # Вересень
        10: 31, # Жовтень
        11: 30, # Листопад
        12: 31 # Грудень
    }

    if day < 1 or day > days_in_month[month]: # перевірка дня
        return False

    return True
```

Принцип роботи функції

Функція date_correct(day, month, year) перевіряє, чи є введена дата коректною.

Перевірка року: Якщо year менше або дорівнює 0, дата некоректна, повертає False.

Перевірка місяця: Якщо month менше 1 або більше 12, дата некоректна, повертає False.

Перевірка дня: Використовується словник days_in_month, який містить кількість днів у кожному місяці.

Для лютого (month == 2) враховується, чи рік високосний (29 днів) чи ні (28 днів).

Якщо day менше 1 або більше за кількість днів у вказаному місяці, дата некоректна, повертає False.

Якщо всі перевірки пройдені успішно, повертає True (дата коректна).

Функція забезпечує коректність дати з урахуванням року, місяця та кількості днів у місяці.

Тестові запуски функції

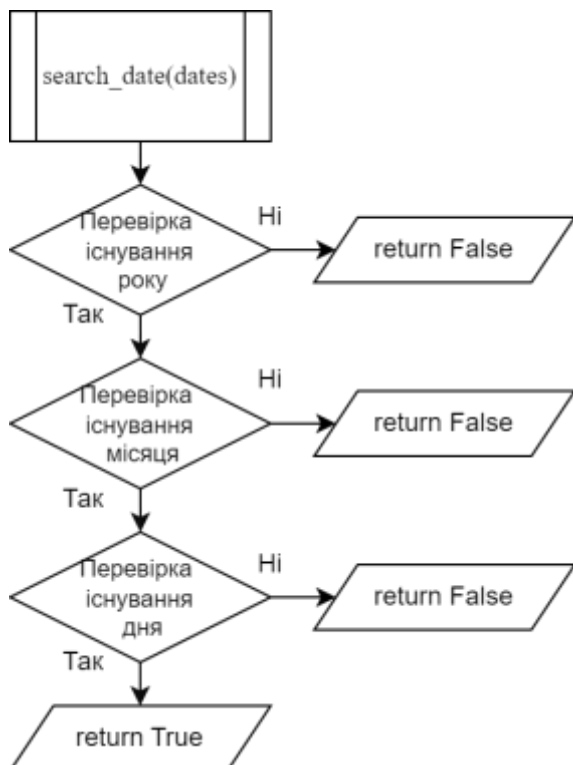
```
Оберіть функцію: 3
Введіть день: 4
Введіть місяць: 3
Введіть рік: 2008
04.03.2008 - коректна дата
```

Рисунок 5

```
Оберіть функцію: 3
Введіть день: 123
Введіть місяць: 13
Введіть рік: 14411484123
123.13.14411484123 - некоректна дата
```

Рисунок 6

Блок-схема



Завдання 4

4. Функцію, що як аргумент отримує список дат у форматі dd.mm.yyyy (наприклад, ["11.12.1877", "25.01.2022", "01.05.2023"]) і шукає у ньому повтори, якщо повтори є – повертає список значень, що повторюються

Лістинг функції

```
def search_date(dates): # Function 4
    """Функція яка отримує список дат у форматі dd.mm.yyyy та
    та шукає у ньому повтори дат та виводить їх список
    Parameters:
    dates (list): Список дат у форматі dd.mm.yyyy

    Returns:
    list: Список повторюваних дат або повідомлення про їх відсутність"""

    date_counts = {} # створюємо словник для підрахунку дат

    for date in dates: # перебір дат
        # Перевіряємо формат дати
        parts = date.split('.') # розділяємо дату на частини
        if len(parts) != 3: # якщо дата не має трьох частин
            print(f"Неправильний формат дати: {date}") # виводимо повідомлення
            continue

        # Розбиваємо дату на день, місяць та рік
        day, month, year = map(int, parts)

        if date_correct(day, month, year):
            # Якщо дата існує, додаємо її до словника
            if date in date_counts:
                date_counts[date] += 1
            else:
                date_counts[date] = 1
        else:
            print(f"Дата {date} не існує") # виводимо повідомлення

    duplicates = {date: count for date, count in date_counts.items() if count > 1} #
    знаходимо дати з лічильником більше 1
    return duplicates if duplicates else {"Немає однакових дат"} # повертаємо словник
    дублікатів або повідомлення про їх відсутність
```

Принцип роботи функції

Функція search_date(dates) шукає повторювані дати у списку та повертає їх.

Ініціалізація словника: Створюється словник date_counts для підрахунку кількості входжень кожної дати.

Перебір дат: Кожна дата зі списку `dates` розбивається на частини (день, місяць, рік) за допомогою `split('.')`.

Перевіряється, чи має дата правильний формат (три частини: день, місяць, рік). Якщо ні, виводиться повідомлення про помилку.

Перевірка коректності дати: Використовується функція `date_correct(day, month, year)` для перевірки, чи існує така дата.

Якщо дата коректна, вона додається до словника `date_counts` з лічильником входжень.

Пошук дублікатів: Створюється словник `duplicates`, який містить тільки ті дати, які зустрічаються більше одного разу.

Повернення результату: Якщо є дублікати, повертається словник `duplicates`.

Якщо дублікатів немає, повертається повідомлення `{"Немає однакових дат"}`.

Тестові запуски функції

```
Меню функції:
1. Введення дат з клавіатури
2. Випадкові дати

Виберіть пункт меню: 1
Введіть дати у форматі dd.mm.yyyy (натисніть Enter з пустим полем для завершення): 22.02.2022
Введіть дати у форматі dd.mm.yyyy (натисніть Enter з пустим полем для завершення): 22.02.2022
Введіть дати у форматі dd.mm.yyyy (натисніть Enter з пустим полем для завершення): 31.12.2077
Введіть дати у форматі dd.mm.yyyy (натисніть Enter з пустим полем для завершення):

Список дат:
22.02.2022
22.02.2022
31.12.2077
Знайдені дублікати:
Дата 22.02.2022 зустрічається 2 разів
```

Рисунок 7

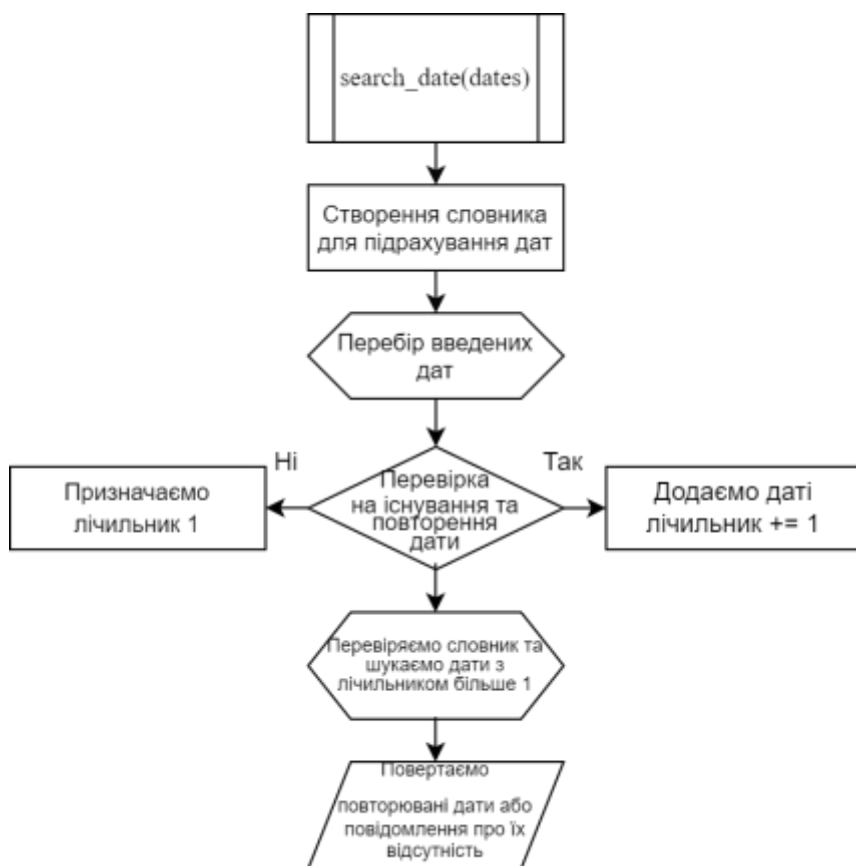
Меню функції:
1. Введення дат з клавіатури
2. Випадкові дати

Виберіть пункт меню: 2
Введіть кількість дат для генерації: 10

Список дат:
01.02.1958
16.01.1971
02.03.2056
20.05.1993
26.04.1993
17.11.2009
26.11.1950
28.01.2073
13.08.1964
03.02.2038
Дублікатів не знайдено

Рисунок 8

Блок-схема



Завдання 5

5. Функцію, у якої немає аргументів, а повертає вона випадкову дату у форматі dd.mm.yyyy. Примітка: Для генерації випадкових чисел використати бібліотеку random та, наприклад, її функцію randint

Лістинг функції

```
import random as r

def random_data(): # Function 5
    """Функція у якої немає аргументів та яка повертає випадкову дату у форматі
    "dd.mm.yyyy" """
    while True: # ЦИКЛ для введення дат
        day = r.randint(1, 31) # випадковий день
        month = r.randint(1, 12) # випадковий місяць
        year = r.randint(1939, 2077) # випадковий рік

        if month == 2:
            if ask_year(year):
                if day <= 29:
                    break
            else:
                if day <= 28:
                    break
        elif month in [4, 6, 9, 11]:
            if day <= 30:
                break
        else:
            break
    return f"{day:02d}.{month:02d}.{year}" # виведення випадкової дати
```

Принцип роботи функції

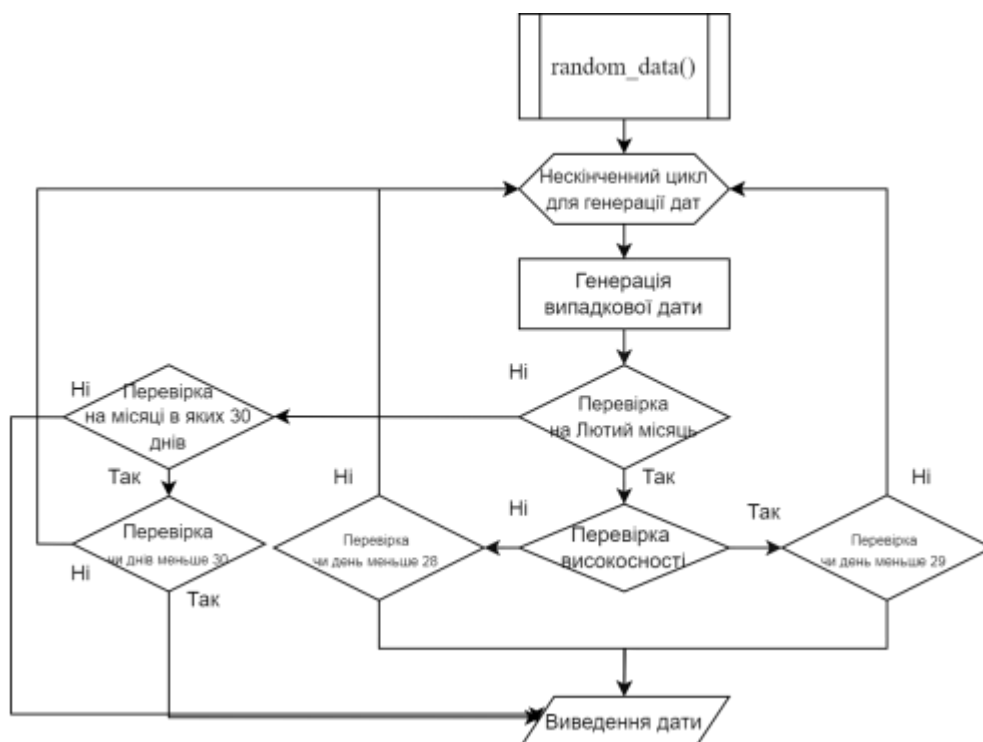
Функція random_data() генерує випадкову коректну дату у форматі “dd.mm.yyyy”

Генерація випадкових значень:

День (day) генерується випадково від 1 до 31.

Місяць (month) генерується випадково від 1 до 12.

Рік (year) генерується випадково від 1939 до 2077.



Завдання 6

6. Функцію, що отримує як аргументи дві дати (формат обрати самостійно) та повертає кількість днів між ними. Примітка: Використати бібліотеку `datetime` та її функцію з такою ж назвою `datetime`.

Лістинг функції

```
def dates(date1, date2): # Function 6
    """Функція яка отримує дві дати в форматі dd.mm.yyyy та
    виводить різницю в днях між ними. З використанням бібліотеки datetime та її
    функцією datetime"""

    date1 = dt.datetime.strptime(date1, "%d.%m.%Y") # перетворюємо рядок в об'єкт
datetime
    date2 = dt.datetime.strptime(date2, "%d.%m.%Y") # перетворюємо рядок в об'єкт
datetime

    difference = date1 - date2 # знаходимо різницю між датами
    days = difference.days # знаходимо кількість днів
    years = days // 365 # знаходимо кількість років

    return days, years # повертаємо кількість днів та років
```

Принцип роботи функції

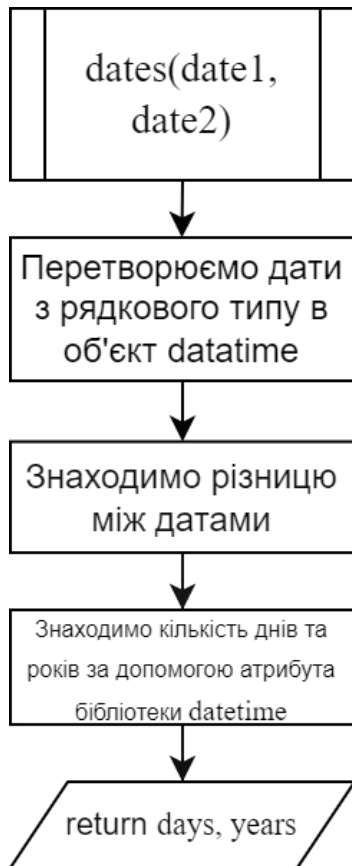
Функція `dates` отримує дві дати у форматі `dd.mm.yyyy`, перетворює їх на об'єкти типу `datetime` за допомогою бібліотеки `datetime`, обчислює різницю між цими датами у днях та роках, і повертає ці значення.

Тестові запуски функції

```
Оберіть функцію: 6
Введіть першу(більшу) дату у форматі dd.mm.yyyy: 12.03.2025
Введіть другу(меншу) дату у форматі dd.mm.yyyy: 27.03.2007
Різниця між датами: 12.03.2025 та 27.03.2007 : 6560 днів та 17 років
```

Рисунок 10

Блок-схема



Завдання 7

7. Функцію, що отримує як аргумент номер місяця, виводить на екран відповідну назву місяця та нічого не повертає.

Лістинг функції

```
def ask_month (nam_month): # Function 7

    """Функція яка приймає як аргумент номер місяця і виводить його нзву\n
    на екран та нічого не повертає.

    Parameters:
    month_number (int): Номер місяця (1-12)
    """

    month = { # словник з назвами місяців
        1: "Січень",
        2: "Лютий",
        3: "Березень",
        4: "Квітень",
        5: "Травень",
        6: "Червень",
        7: "Липень",
        8: "Серпень",
        9: "Вересень",
```

```
10: "Жовтень",  
11: "Листопад",  
12: "Грудень"  
}
```

```
if not isinstance(nam_month, int) or nam_month < 1 or nam_month > 12: # перевірка  
чи введено число та чи воно в межах від 1 до 12  
    print("Такого місяця не існує")  
else:  
    print(month[nam_month]) # виведення назви місяця
```

Принцип роботи функції

Функція `ask_month` приймає номер місяця (ціле число від 1 до 12) і виводить його назву на екран.

Створюється словник, де ключі — це номери місяців (1-12), а значення — відповідні назви місяців.

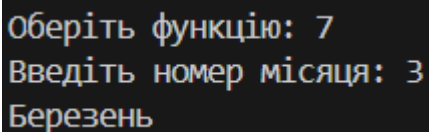
Перевіряється, чи є введений аргумент цілим числом (`int`).

Перевіряється, чи знаходиться число в межах від 1 до 12.

Якщо введене значення некоректне, виводиться повідомлення: "Такого місяця не існує".

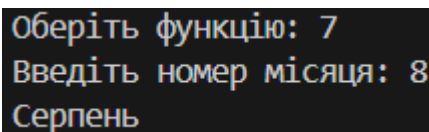
Якщо значення коректне, виводиться назва місяця зі словника.

Тестові запуски функції



```
Оберіть функцію: 7  
Введіть номер місяця: 3  
Березень
```

Рисунок 11



```
Оберіть функцію: 7  
Введіть номер місяця: 8  
Серпень
```

Рисунок 12

Блок-Схема



Лістинг main файлу програми

```
from func_modul import * # Імпорт функції з іншого файлу

def main():
    """Основна функція яка викликає інші функції з модуля\n
    func_modul та виводить результат їх роботи"""

    main_menu = { # Список головного меню
        1: "Визначення пори року за номером місяця",
        2: "Перевірка року на високосність",
        3: "Перевірка коректності введення дати",
        4: "Визначення повторюваних дат",
        5: "Випадкова дата у форматі dd.mm.yyyy",
        6: "Порівняння двох дат, різниця в днях та роках між ними",
        7: "Визначення місяця за номером",
        8: "Вихід"
    }

    while True:
        # Привітання та виведення меню
        print("\nВас вітає програма для роботи з датами та місяцями\n"
              "В цій програмі ви можете обрати одну з наступних дій\n"
              "які в свою чергу виконують обрану функцію\n")
        for key, value in main_menu.items(): # Виведення меню
            print(f"{key}: {value}")

        # Вибір функції
        menu = (input("\nОберіть функцію: ")) # Вибір дії

        # Валідація введення
        if not menu.isdigit(): # перевірка чи введене число
            print("Номер повинен бути числом від 1 до 8"
                  "Продовжити роботу з програмою? (Введіть n для завершення або Enter для\n"
                  "продовження)")
            if input() == "n":
                print("До побачення!")
                break
            else:
```



```

        continue

menu = int(menu) # перетворення введеного значення в число

# Валідація введення
if menu not in main_menu:
    print("Не вірний номер функції виберіть від 1 до 8"
          " Продовжити роботу з програмою? (Введіть n для завершення або Enter для"
          " продовження)")
    if input() == "n":
        print("До побачення!")
        break
    else:
        continue

# Виклики функцій

if menu == 1:
    n = int(input("Введіть номер місяця: ")) # введення номера місяця
    ask_mans(n) # виклик функції

    # Запит на продовження роботи з програмою
    print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для"
          " продовження)")
    if input() == "n":
        print("До побачення!")
        break
    else:
        continue

elif menu == 2:
    year = int(input("Введіть рік: ")) # введення року
    ask_year(year) # виклик функції

    if ask_year(year): # перевірка чи рік високосний
        print("Рік високосний") # виведення результату
    else:
        print("Рік не високосний") # виведення результату

    # Запит на продовження роботи з програмою
    print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для"
          " продовження)")
    if input() == "n":
        print("До побачення!")
        break
    else:
        continue

elif menu == 3:
    day = int(input("Введіть день: ")) # введення дня
    month = int(input("Введіть місяць: ")) # введення місяця
    year = int(input("Введіть рік: ")) # введення року

    if date_correct(day, month, year): # виклик функції
        print(f"{day:02d}.{month:02d}.{year} - коректна дата") # виведення результату
    else:
        print(f"{day:02d}.{month:02d}.{year} - некоректна дата") # виведення
результату

    # Запит на продовження роботи з програмою

```

```

print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для
продовження)")
if input() == "n":
    print("До побачення!")
    break
else:
    continue

elif menu == 4:
    list_date = [] # створюємо пустий список для дат

    # Запит на вибір методу введення дат
    func_metod = {
        1: "Введення дат з клавіатури",
        2: "Випадкові дати"
    }

    print("Меню функції:")
    for key, value in func_metod.items(): # виведення меню
        print(f"{key}. {value}")

    input_method = (input("\nВиберіть пункт меню: "))

    if not input_method.isdigit(): # перевірка чи введене число
        print("Номер повинен бути числом від 1 до 2")
        return

    if input_method == "1":
        while True: # цикл для введення дат
            date_inupt = input("Введіть дати у форматі dd.mm.yyyy (натисніть Enter\nабо пустим полем для завершення): ") # введення дат

            if date_inupt == "": # якщо натиснуто Enter
                break # вихід з циклу

            list_date.append(date_inupt) # додавання дати в список

        elif input_method == "2":
            num_dates = int(input("Введіть кількість дат для генерації: "))
            for _ in range(num_dates):
                list_date.append(random_data())

    print("\nСписок дат:")
    for date in list_date: # виведення списку дат
        print(date)

    duplicate_dates = search_date(list_date) # отримуємо словник повторюваних дат

    if duplicate_dates: # якщо є повторювані дати
        if "Немає однакових дат" in duplicate_dates:
            print("Дублікатів не знайдено\n")
        else:
            print("Знайдені дублікати:")
            for date, count in duplicate_dates.items():
                print(f"Дата {date} зустрічається {count} разів")
    else:
        print("Дублікатів не знайдено\n")
    print()

    # Запит на продовження роботи з програмою

```

```

        print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для
продовження)")
        if input() == "n":
            print("До побачення!")
            break
        else:
            continue

    elif menu == 5:

        print(f"\nВипадкова дата --> {random_data()}") # виклик функції

        # Запит на продовження роботи з функцією
        print("Продовжити роботу з програмою? (Введіть n для завершення або Enter
для продовження)")
        if input() == "n":
            print("До побачення!")
            break
        else:
            continue

    elif menu == 6:
        date1 = str(input("Введіть першу(більшу) дату у форматі dd.mm.yyyy: ")) # введення
першої дати
        date2 = str(input("Введіть другу(меншу) дату у форматі dd.mm.yyyy: ")) # введення
другої дати

        days, years = dates(date1, date2) # виклик функції
        if days >= 0: # перевірка чи друга дата менша за першу
            print(f"Різниця між датами: {date1} та {date2} : {days} днів та {years}
років") # виведення результату
        else:
            print("Перша дата повинна бути меншою за другу") # виведення результату

        # Запит на продовження роботи з програмою
        print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для
продовження)")
        if input() == "n":
            print("До побачення!")
            break
        else:
            continue

    elif menu == 7:
        nam_month = input("Введіть номер місяця: ") # введення номера місяця

        if not nam_month.isdigit():
            print("Номер місяця повинен бути числом")
        else:
            nam_month = int(nam_month)
            ask_month(nam_month) # виклик функції

        # Запит на продовження роботи з програмою
        print("Продовжити роботу з програмою? (Введіть n для завершення або Enter для
продовження)")
        if input() == "n":
            print("До побачення!")
            break
        else:

```

```

        continue

    elif menu == 8:
        print("Вихід з програми")
        break

    break

if __name__ == "__main__": # Перевірка чи файл був запущений напряму
    main() # Виклик основної функції

```

Контрольні запитання

1. Як створити функцію у мові Python?

Щоб створити функцію у Python, потрібно використати ключове слово `def`, за яким йде назва функції, круглі дужки для параметрів та двокрапка. Тіло функції пишеться з відступом. Наприклад:

```

def назва_функції(параметри):
    # Тіло функції
    return результат # Необов'язково

```

2. Що таке модулі та пакети?

Модуль — це файл з розширенням `.py`, який містить код на Python (функції, класи, змінні тощо). Він дозволяє організувати та повторно використовувати код.

Пакет — це директорія, яка містить один або кілька модулів та файл `__init__.py`. Він дозволяє групувати модулі в ієрархічну структуру.

3. Які бувають способи підключення модулів та пакетів?

Існує кілька способів підключення модулів та пакетів у Python:

```
import модуль
```

Підключення з псевдонімом:

```
import модуль as псевдонім
```

Підключення конкретних об'єктів з модуля:

```
from модуль import об'єкт
```

Підключення всього вмісту модуля (не рекомендується через можливість конфліктів імен): `from модуль import *`

Підключення пакету або підмодуля:

```
from пакет import модуль
```

4. Що таке анонімні функції та інструкція `lambda`?

Анонімні функції (або лямбда-функції) — це функції, які створюються без використання ключового слова `def`. Вони є одностроковими та не мають імені.

Використовуються для коротких і простих операцій.

lambda аргументи: вираз

5. Чи можна використати модуль як самостійну програму?

Так, модуль у Python можна використати як самостійну програму. Для цього використовується спеціальна конструкція `if __name__ == "__main__":`. Це дозволяє визначити, чи запущений модуль напряму, чи імпортований у інший скрипт.

Як це працює: Коли модуль запускається напряму, змінна `__name__` має значення `"__main__"`. Якщо модуль імпортований, `__name__` містить ім'я модуля.

Приклад:

```
# my_module.py
def my_function():
    print("Це функція з модуля")

if __name__ == "__main__":
    print("Модуль запущено напряму")
    my_function()
else:
    print("Модуль імпортовано")
```