Імпорт модуля datetime
Спершу потрібно імпортувати модуль **datetime**, щоб мати можливість використовувати його функціональність:

```
import datetime
```

Тепер ми готові приступити до роботи з різними типами об'єктів, які надає цей модуль.
Створення об'єктів date, time, datetime, timedelta
date: Цей об'єкт представляє дату (рік, місяць, день) і не має часової інформації. Для створення об'єкта **date**, використовуйте конструктор **date(year, month, day)**:

```
my_date = datetime.date(2023, 11, 8)
```

my_time = datetime.time(14, 30, 0)

time: Цей об'єкт представляє час (години, хвилини, секунди) і не має інформації про дату. Для створення об'єкта **time**, використовуйте конструктор **time(hour, minute, second, microsecond)**:

my_time = datetime.time(14, 30, 0)

datetime: Цей об'єкт поєднує інформацію про дату та час. Для створення об'єкта **datetime**, використовуйте конструктор **datetime(year, month, day, hour, minute, second, microsecond)**:

my_datetime = datetime.datetime(2023, 11, 8, 14, 30, 0)

timedelta: Цей об'єкт представляє різницю між двома датами або часами. Він дозволяє виконувати операції додавання або віднімання часу від дати чи часу. Для створення об'єкта **timedelta**, використовуйте конструктор **timedelta(days, seconds, microseconds, milliseconds, minutes, hours, weeks)**:

time_difference = datetime.timedelta(days=1, hours=3)

Ці об'єкти дозволяють працювати з датою, часом та їх комбінаціями, що дуже корисно при роботі з датами і часом у програмах. Вони надають широкі можливості для обчислень та форматування дати та часу.
Отримання поточної дати та часу: • Використання datetime.now().

Для отримання поточної дати та часу використовуйте функцію **datetime.now()** з модуля **datetime**. Ось приклад використання:

```
import datetime

current_datetime = datetime.datetime.now()

print("Поточна дата і час:", current_datetime)
```

Цей код створить об'єкт **current_datetime**, який буде містити поточну дату та час на момент виконання програми. Результат буде виглядати приблизно так:

Отримана інформація включає рік, місяць, день, години, хвилини, секунди і мікросекунди. Ви можете подальше використовувати цей об'єкт для виконання операцій з датою та часом або для виводу їх у зручному форматі.

додати час
Для додавання часу до поточної дати та часу використовуйте об'єкт **timedelta** з модуля **datetime**. Ось приклад, як можна додати, наприклад, 1 годину і 30 хвилин:

```
import datetime

# Отримання поточної дати та часу
current_datetime = datetime.datetime.now()

# Визначення часового інтервалу, який потрібно додати
time_difference = datetime.timedelta(hours=1, minutes=30)

# Додавання часового інтервалу до поточної дати та часу
new_datetime = current_datetime + time_difference

# Виведення результату
print("Поточна дата і час:", current_datetime)
print("Після додавання часу:", new_datetime)
```

У цьому прикладі використовується **timedelta(hours=1, minutes=30)**, щоб визначити, що ми хочемо додати 1 годину і 30 хвилин. Результат буде виведено у вигляді:

Ви можете налаштовувати **timedelta** на будь-який інтервал часу, який вам потрібен (години, хвилини, секунди, мікросекунди тощо), і додавати його до поточної дати та часу за допомогою операції додавання (+).

3. Робота з датами: • Вивід дати у різних форматах. • Обробка дати, витягнення окремих компонентів (рік, місяць, день).

Вивід дати у різних форматах
Для виведення дати у різних форматах використовуйте метод **strftime** (string format time) об'єкта **datetime**. Ось приклади:

```
import datetime

# Отримання поточної дати та часу
current_datetime = datetime.datetime.now()

# Виведення дати у різних форматах
print("Повна дата та час:", current_datetime.strftime("%Y-%m-%d %H:%M:%S"))
print("Коротка дата:", current_datetime.strftime("%d.%m.%Y"))
print("Час:", current_datetime.strftime("%H:%M"))
```

У цьому прикладі **%Y**, **%m**, **%d**, **%H**, **%M**, **%S** є форматами, які представляють різні компоненти дати та часу (рік, місяць, день, години, хвилини, секунди). Більше інформації про формати можна знайти у документації.
Обробка дати, витягнення окремих компонентів
Для отримання окремих компонентів дати, таких як рік, місяць і день, можна використовувати атрибути об'єкта **datetime**. Ось приклад:

```
import datetime

# Отримання поточної дати та часу
current_datetime = datetime.datetime.now()

# Виведення дати у різних форматах
print("Повна дата та час:", current_datetime.strftime("%Y-%m-%d %H:%M:%S"))
print("Коротка дата:", current_datetime.strftime("%d.%m.%Y"))
print("Час:", current_datetime.strftime("%H:%M"))
```

Ви також можете використовувати атрибути **hour**, **minute**, **second** для отримання годин, хвилин і секунд відповідно.

Ці операції дозволяють легко працювати з компонентами дати та часу та виводити їх в зручному форматі.

4. Робота з часом: • Вивід часу у різних форматах. • Обробка часу, витягнення окремих компонентів (години, хвилини, секунди).

Вивід часу у різних форматах
Для виведення часу у різних форматах також використовуємо метод **strftime** об'єкта **datetime**, але зосередимось на форматах, що стосуються лише часу:

```
import datetime

# Отримання поточної дати та часу
current_datetime = datetime.datetime.now()

# Виведення дати у різних форматах
print("Повна дата та час:", current_datetime.strftime("%Y-%m-%d %H:%M:%S"))
print("Коротка дата:", current_datetime.strftime("%d.%m.%Y"))
print("Час:", current_datetime.strftime("%H:%M"))
```

У цьому прикладі **%H**, **%M**, **%S** є форматами, які представляють години, хвилини і секунди відповідно. Більше про формати можна знайти у документації.
Обробка часу, витягнення окремих компонентів
Для отримання окремих компонентів часу, таких як години, хвилини і секунди, можна використовувати атрибути об'єкта **time**. Ось приклад:

```
import datetime

# Отримання поточної дати та часу
current_datetime = datetime.datetime.now()

# Витягнення окремих компонентів
year = current_datetime.year
month = current_datetime.month
day = current_datetime.day

# Виведення окремих компонентів
print("Рік:", year)
print("Місяць:", month)
print("День:", day)
```

Ви можете використовувати атрибути **hour**, **minute** і **second** для отримання годин, хвилин і секунд відповідно.
Ці операції дозволяють вам легко працювати з компонентами часу та виводити їх в зручному форматі.

5. Робота з об'єктами datetime та timedelta: • Вирахування різниці між датами та часом. • Додавання або віднімання інтервалу часу.

Вирахування різниці між датами та часом
Для вирахування різниці між двома об'єктами **datetime** використовуйте операцію віднімання. Ось приклад:

```
import datetime

# Дві дати
date1 = datetime.datetime(2023, 1, 1)
date2 = datetime.datetime(2023, 12, 31)

# Вирахування різниці
date_difference = date2 - date1

# Виведення різниці
print("Різниця між датами:", date_difference)
```

Отримана різниця буде представлена об'єктом **timedelta**, який містить інформацію про кількість днів, годин, хвилин, секунд тощо між двома датами.
Додавання або віднімання інтервалу часу
Для додавання або віднімання інтервалу часу до/від об'єкта **datetime** використовуйте об'єкт **timedelta**. Ось приклади:

```
import datetime

# Поточна дата та час
current_datetime = datetime.datetime.now()

# Додавання 3 годин до поточної дати та часу
new_datetime_after_addition = current_datetime + datetime.timedelta(hours=3)

# Віднімання 2 днів від поточної дати та часу
new_datetime_after_subtraction = current_datetime - datetime.timedelta(days=2)
```

```
# Виведення результатів
print("Поточна дата і час:", current_datetime)
print("Після додавання 3 годин:", new_datetime_after_addition)
print("Після віднімання 2 днів:", new_datetime_after_subtraction)
```

У цих прикладах **timedelta(hours=3)** вказує на додавання 3 годин, а **timedelta(days=2)** - на віднімання 2 днів.
Такі операції забезпечують можливість простого використання інтервалів часу для зміни значень дати та часу.

7. Модуль time: • Введення до модуля time. • Робота з епохою UNIX.

Введення до модуля time
Модуль **time** в Python надає функції для роботи з часом. Однією з основних функцій цього модуля є робота з епохою UNIX.
Робота з епохою UNIX
Епоха UNIX - це визначений момент у часі, який визначає початок обліку часу для багатьох операційних систем. В Python, епоха UNIX визначається як 1 січня 1970 року 00:00:00 UTC.
Отримання поточного часу в секундах з епохи UNIX

```
import time

current_time_seconds = time.time()
print("Поточний час в секундах з епохи UNIX:", current_time_seconds)
```

Метод **time.time()** повертає поточний час в секундах з епохи UNIX.
Перетворення секунд в об'єкт struct_time

```
import time

current_time_seconds = time.time()
struct_time = time.gmtime(current_time_seconds)

print("Об'єкт struct_time:", struct_time)
```

Метод **time.gmtime()** приймає час у секундах з епохи UNIX і повертає об'єкт **struct_time**, який містить різні компоненти часу (рік, місяць, день, година, хвилина, секунда та ін.).

Перетворення об'єкта struct_time **в рядок**

```
import time

current_time_seconds = time.time()
struct_time = time.gmtime(current_time_seconds)

time_string = time.strftime("%Y-%m-%d %H:%M:%S", struct_time)
print("Поточний час у форматі рядка:", time_string)
```

Метод **time.strftime()** дозволяє перетворити об'єкт **struct_time** у рядок за заданим форматом.
Ці операції надають можливість працювати з часом у секундах та в інших представленнях, таких як **struct_time**.

8. Вимірювання часу виконання коду: • Використання time.time().

Вимірювання часу виконання коду за допомогою time.time()
Для вимірювання часу виконання коду можна використовувати метод **time.time()**. Ось приклад:

```
import time

# Запам'ятовуємо поточний час перед виконанням коду
start_time = time.time()

# Ваш код, який ви хочете виміряти

# Запам'ятовуємо час після виконання коду
end_time = time.time()

# Обчислюємо різницю між початковим і кінцевим часом
execution_time = end_time - start_time

# Виводимо час виконання коду
print("Час виконання коду: {:.5f} секунд".format(execution_time))
```

У цьому прикладі, код, який вам потрібно виміряти, поміщається між отриманням часу перед та після виконання коду. Різниця між цими двома часами дає час виконання коду в секундах.
Цей метод дозволяє вам ефективно вимірювати час виконання конкретного коду та оптимізувати його продуктивність.

9. Введення до інструментів локалізації: • Імпорт модуля locale. 10. Зміна локалізації: • Встановлення різних локалізацій для форматування дати та часу. 11. Форматування дати та часу: • Використання locale.strftime().

Введення до інструментів локалізації
Модуль **locale** в Python використовується для роботи з локалізацією, тобто зміни представлення даних відповідно до мовних та регіональних встановлень користувача.
Зміна локалізації
Перш ніж ви використовуватимете інструменти локалізації, важливо встановити локаль за замовчуванням. Для цього використовуйте метод **locale.setlocale()**.

```
import locale

# Встановлюємо локаль за замовчуванням
locale.setlocale(locale.LC_TIME, "uk_UA.utf8")  # Приклад для української локалізації

# Решта вашого коду, де ви будете використовувати локалізацію
```

Форматування дати та часу
Після встановлення локалізації ви можете використовувати метод **locale.strftime()** для форматування дати та часу з урахуванням локальних налаштувань.

```
import locale
import time

# Встановлюємо локаль за замовчуванням
locale.setlocale(locale.LC_TIME, "uk_UA.utf8")  # Приклад для української локалізації

# Отримуємо поточний час
current_time = time.localtime()

# Форматуємо дату та час відповідно до локалізації
formatted_time = locale.strftime("%A, %d %B %Y %H:%M:%S", current_time)

# Виводимо отформатований час
print("Отформатований час:", formatted_time)
```

У цьому прикладі **%A**, **%d**, **%B**, **%Y**, **%H**, **%M**, **%S** - це формати для виведення дня тижня, дня місяця, повного назви місяця, року, годин, хвилин і секунд відповідно.
Використання інструментів локалізації дозволяє легко адаптувати формати дати та часу до мовних та регіональних вимог користувача.