Назначение лекции

Новые слова

**Инструкция if -** позволяет вашей программе принимать решения и изменять тот путь, по которому происходит передача управления в вашей программе.

# Работа с датой

Для работы с датой может использоваться несколько модулей. В лекции рассмотрен модуль datetime.

import datetime  
yesterday = datetime.date(2017,5, 2)  
print(yesterday) *# 2017-05-02*

Создание и печать даты в «американском» формате.

from datetime import **date**

today = date.today()

print(today) # текущая дата в американском формате

print("{}.{}.{}".format(today.day, today.month, today.year)) #дата

Используя класс date и метод today можно получить текущую дату в «американском» и привычном форматах.

from datetime import **time**

current\_time = time()

print(current\_time) # 00:00:00

current\_time = time(16, 25)

print(current\_time) # 16:25:00

current\_time = time(16, 25, 45)

print(current\_time) # 16:25:45

Метод time() последовательно принимает часы, минуты, секунды и микросекунды. Все параметры необязательные, и если мы какой-то параметр не передадим, то соответствующее значение будет инициализироваться нулем.

from datetime import **datetime**

deadline = datetime(2017, 5, 10)

print(deadline) # 2017-05-10 00:00:00

deadline = datetime(2017, 5, 10, 4, 30)

print(deadline) # 2017-05-10 04:30:00

Класс datetime из одноименного модуля объединяет возможности работы с датой и временем.

from datetime import datetime

now = datetime.now()

print(now) # 2023-07-10 08:45:28.451328

print("{}.{}.{} {}:{}".format(now.day, now.month, now.year, now.hour, now.minute)) # 10.7.2023 8:45

print(now.date()) # 2023-07-10

print(now.time()) # 08:45:28.451328

Для получения текущих даты и времени можно вызвать метод now().

from datetime import datetime  
deadline = datetime.strptime("10/07/2023", "%d/%m/%Y")  
print(deadline) *# 2023-07-10 00:00:00*deadline = datetime.strptime("10/07/2023 09:01", "%d/%m/%Y %H:%M")  
print(deadline) *# 2023-07-10 09:01:00*deadline = datetime.strptime("07-10-2023 09:01", "%m-%d-%Y %H:%M")  
print(deadline) *# 2023-10-07 09:01:00*

Метод strptime(), который позволяет «распарсить» строку и преобразовать ее в дату в «американском» формате. Этот метод принимает два параметра: strptime(str, format). Первый параметр str представляет строковое определение даты и времени, а второй параметр - формат, который определяет, как различные части даты и времени расположены в этой строке:

%d: день месяца в виде числа

%m: порядковый номер месяца

%y: год в виде 2-х чисел

%Y: год в виде 4-х чисел

%H: час в 24-х часовом формате

%M: минута

%S: секунда

from datetime import datetime  
now = datetime.now()  
print(now.strftime("%Y-%m-%d")) *# 2023-07-10*print(now.strftime("%d/%m/%Y")) *# 10/07/2023*print(now.strftime("%d/%m/%y")) *# 10/07/23*print(now.strftime("%d %B %Y (%A)")) *# 10 July 2023 (Monday)*print(now.strftime("%d/%m/%y %I:%M")) *# 10/07/23 09:20*

Для форматирования объектов date и time в обоих этих классах предусмотрен метод strftime(format). Для определения формата можно использовать один из следующих кодов форматирования:

* %a: аббревиатура дня недели. Например, Wed - от слова Wednesday (по умолчанию используются английские наименования)
* %A: день недели полностью, например, Wednesday
* %b: аббревиатура названия месяца. Например, Oct (сокращение от October)
* %B: название месяца полностью, например, October
* %d: день месяца, дополненный нулем, например, 01
* %m: номер месяца, дополненный нулем, например, 05
* %y: год в виде 2-х чисел
* %Y: год в виде 4-х чисел
* %H: час в 24-х часовом формате, например, 13
* %I: час в 12-ти часовом формате, например, 01
* %M: минута
* %S: секунда
* %f: микросекунда
* %p: указатель AM/PM
* %c: дата и время, отформатированные под текущую локаль
* %x: дата, отформатированная под текущую локаль
* %X: время, форматированное под текущую локаль

from datetime import datetime  
import locale  
locale.setlocale(locale.LC\_ALL, 'Russian\_Russia.1251')  
now = datetime.now()  
print(now.strftime("%d %B %Y (%A)")) *# 10 Июль 2023 (понедельник)*

При выводе названий месяцев и дней недели по умолчанию используются английские наименования. Если мы хотим использовать текущую локаль, но то мы можем ее предварительно установить с помощью модуля locale.

Для иллюстрации работы с датой покажем код для ввода даты рождения и сравнения этой даты с текущей датой:

from datetime import datetime  
*# Получение текущей даты*current\_date = datetime.now().date()  
*# Ввод даты о дне пождения*user\_input = input("Введите дату рождения в формате ДД-ММ-ГГГГ: ")  
*# Преобразование введенной строки в объект datetime*try:  
 input\_date = datetime.strptime(user\_input, "%d-%m-%Y").date()  
 if input\_date > current\_date:  
 print("У вас день рождения в этом году будет.")  
 elif input\_date < current\_date:  
 print("У вас день рождения в этом году уже был.")  
 else:  
 print("У вас день рождения сегодня.")  
except ValueError:  
 print("Некорректный формат даты.")

Текущая дата вводится в виде строки и сразу преобразуется в формат даты в переменную current\_date, затем вводится дата рождения в требуемом виде, как строка, в переменную user\_input, и затем преобразуется в формат даты, методом date(). В результате две даты в одинаковом формате и их можно сравнивать.

Данный код содержит «исключение» try. Это важный механизм, который применяется для решения разных аварийных ситуаций и не только при работе с датой. Рассмотрим этот механизм подробно.

# Исключения

Вот общий вид исключения:

try:

    #код, который может вызвать аварию

except причина\_аварии1:

    #код обработки причины1

except причина\_аварии2:

    #код обработки причины2

. . .

except причина\_аварииN:

    #код обработки причиныN

finally:

    #код обработки другой причины

После оператора try следует код, где может произойти аварийная ситуация. Например, при работе с файлом, преобразовании даты и т.д. Оператор except обозначает «обработчик» аварийной ситуации.

Причин аварии может быть много. В Python их тоже много, например, **ValueError** – возникает, когда функция получают аргумент с неправильным значением. Список исключений есть в сети и в любой литературе.

Можно вообще не указывать причину, тогда при любой аварии произойдет исключение.

Оператор finally применяется в том случае, если ни одна из причин аварии не сработала.

Очень часто механизм исключений применяется при работе с файлами или базами данных. Например, не найден требуемый файл и об этом надо сообщить пользователю, не завершая программы.

Программист должен помнить, что аварийное завершение программы - это очень нежелательное событие. Программа должна всегда сообщать о наличии аварии (случай отключения электроэнергии не рассматривается). Именно для таких случаев требуется механизм исключений.

Пока все