Como trabalhar com data e hora no Python

Às vezes pode ser uma dor de cabeça trabalhar com data e hora em determinadas linguagens de programação, mas, não para o Python.
O módulo `datetime` chegou a partir da versão 2.3 do Python para facilitar a manipulação de informações de data e hora.
No decorrer deste artigo, veremos como trabalhar com data e hora no Python, exemplos de como utilizar o módulo `datetime`, e porque ele pode ser tão útil.

O que é o módulo datetime?

O módulo `datetime` possui classes para manipular data e tempo. Além disso, ele suporta cálculos aritméticos de datas e seu foco está na retirada eficiente de atributos para formatar resultados e manipulá-los.

Classes do módulo datetime

No módulo `datetime` temos a disposição as seguintes classes:

- date
- time
- timedelta
- timezone

As classes do módulo `datetime` representam abstrações de data e tempo. Desse modo o Python irá entender o que é uma data, hora e fuso horário.

Classe date

Date é uma classe que representa uma data (ano, mês e dia) e seu construtor possui três argumentos, sendo eles: `year`, `month` e `day`. Em resumo, todos os argumentos são obrigatórios e devem ser números pertencentes aos intervalos abaixo:

- `year`: de 0 a 9999
- `month`: de 0 a 12

- `day`: de 0 a 31, dependendo do mês esse valor pode variar

Caso algum argumento esteja fora do intervalo, a exceção `ValueError` será lançada.

```
from datetime import date

data1 = date(year=2022, month=12, day=1)
print(data1)
#2022-12-1

data2 = date(year=2022, month=13, day=1)
print(data2)
#ValueError: month must be in 1..12. O mês está fora do intervalo que é de 1 a 12
```

Principais métodos:

Na classe `date` já vem implementado vários métodos úteis para aumentar a sua produtividade, veja alguns dos métodos mais conhecidos.

- `today`: retorna uma instância da classe date correspondente a data atual.
- `fromisoformat`: retorna uma instância da classe date correspondente a string passada por parâmetro.
- `strftime`: retorna uma instância da classe date correspondente a formatação passada por parâmetro.

Ainda sobre o método `strftime`, temos uma tabela com códigos para formatação seguida de exemplos. Veja mais.

Códigos de formatação de strftime

Código	Exemplo	Descrição
%a	Sun	Dia da semana como nome abreviado do local.
%A	Sunday	Dia da semana como nome completo do local.
%w	0	Dia da semana como número decimal, onde 0 é domingo e 6 é sábado.
%d	08	Dia do mês como número decimal preenchido com zeros.

Código	Exemplo	Descrição
%-d	8	Dia do mês como número decimal.
%b	Sep	Mês como nome abreviado da localidade.
%B	September	Mês como nome completo do local.
%m	09	Mês como número decimal preenchido com zeros.
%-m	9	Mês como número decimal.
%y	13	Ano sem século como número decimal preenchido com zeros.
%Y	2013	Ano com século como número decimal.
%H	07	Hora (24 horas) como número decimal preenchido com zeros.
%-H	7	Hora (24 horas) como número decimal.
%I	07	Hora (12 horas) como número decimal preenchido com zeros.
%-I	7	Hora (12 horas) como número decimal.
%p	AM	O equivalente da localidade AM ou PM.
%M	06	Minuto como número decimal preenchido com zeros.
%-M	6	Minuto como número decimal.
%S	05	Segundo como número decimal preenchido com zeros.
%-S	5	Segundo como número decimal.
%f	000000	Microssegundo como número decimal, preenchido com zeros à esquerda.

Código	Exemplo	Descrição
%z	+0000	Deslocamento UTC no formato ±HHMM[SS[.ffffff]].
%Z	UTC	Nome do fuso horário.
%j	251	Dia do ano como número decimal preenchido com zeros.
%-j	251	Dia do ano como número decimal.
%U	36	Número da semana do ano como número decimal preenchido com zeros.
%W	35	Número da semana do ano como número decimal.
%c	Sun Sep 8 07:06:05.2013	Representação apropriada de data e hora da localidade.
%x	09/08/13	Representação de data apropriada da localidade.
%X	07:06:05	Representação de tempo apropriada do local.
%%	%	Um caractere '%' literal.

Em seguida, veja como utilizar os métodos apresentados:

```
from date import datetime

data_atual = date.today( )
print(data_atual)
#2022-12-20

data_string = date.fromisoformat("2000-01-10")
print(data_string)
#2000-01-10

data4 = date(year=2000, month=12, day=4)
data_brasileira = data4.strftime(""%d/%m/%Y")
print(data_brasileira)
#04/12/2002
```

Classe Time

Time é uma classe que representa uma hora do dia, seu construtor possui argumentos como: `hour`, `minute`, `second` e `microsecond`. Esses argumentos são opcionais com valores padrões iguais a 0, e devem ser números pertencentes aos intervalos abaixo:

- `hour`: de 0 a 23
- `minute`: de 0 a 59
- `second`: de 0 a 59
- `microsecond`: de 0 a 999 999

Fique atento aos intervalos dos argumentos! Caso algum argumento esteja fora do intervalo, a exceção `ValueError` será lançada.

```
from datetime import time

tempo = time()
print()
#00:00:00

tempo1 = time(hour=2, minute=15, second=59, microsecond=9999)
print(tempo1)
#02:15:59.009999
```

Principais métodos

A classe `time` também possui alguns métodos que nos auxiliam na manipulação e criação de objetos, são eles:

- `fromisoformat`: retorna uma instância de datetime com o valor passado como parâmetro.
- `replace`: retorna uma instância de datetime com o valor alterado conforme o parâmetro passado.
- `isoformat`: retorna uma string com o valor no formato da instância time, a partir do parâmetro passado.

Exemplos de uso:

```
from datetime import time

horario = time(hour=6, minute=12, second=30, microsecond=5000)
print(horario)
#06:12:30.005000
```

```
horario_formatado1 = horario.replace(minute=59)
print(horario_formatado1)
#06:59:30.005000

horario_formatado2 = horario.isoformat('minutes')
print(horario_formatado2)
#06:12
```

Classe Datetime

Datetime é uma classe que representa data e hora juntas, em outras palavras, podemos dizer que ela é uma combinação das classes `date` e `time`. Seu construtor possui os argumentos `year`, `month`, `day` e são obrigatórios, enquanto, `hour`, `minute`, `second` e `microsecond` são opcionais. Segue abaixo os intervalos de cada argumento:

- `year`: de 0 a 9999
- `month`: de 0 a 12
- `day`: de 0 a 31, dependendo do mês esse valor pode variar
- `hour`: de 0 a 23
- `minute`: de 0 a 59
- `second`: de 0 a 59
- `microsecond`: de 0 a 999 999

Observe o exemplo:

```
from datetime import datetime

data5 = datetime(year=2010, month=6, day=15)
print(data5)
#2010-06-15 00:00:00

data6 = datetime(year=2010, month=6, day=15, hour=18, minute =30)
print(data6)
#2010-06-15 18:30:00
```

Principais métodos

Os métodos da classe `datetime` são semelhantes aos métodos das classes `date` e `time`. Veja alguns dos métodos mais conhecidos.

- `today`: retorna instância de datetime com valor da data e hora atual.
- `now`: retorna instância da classe datetime com valor da data e hora atual, podendo ter um atributo de `tz` que indica o fuso horário.

- `fromisoformat`: retorna uma instância de `datetime` a partir de uma string passada como parâmetro.

Veja como usar estes métodos:

```
from datetime import datetime

data7 = datetime.now()
print(data1)
#2022-12-19 09:46:37.961831

data8 = datetime.today()
print(data2)
#2022-12-19 09:46:37.961831

data9 = datetime.fromisoformat('2015-03-30')
print(data9)
#2015-03-30 00:00:00
```

Classe timedelta

Timedelta é uma classe que representa uma duração, ou diferença entre duas datas. O construtor de `timedelta` possui argumentos como: `weeks`, `days`, `hous`, `minutes`, `seconds`, `microseconds` e `milliseconds`, todos são opcionais.

Veja os exemplos abaixo:

```
delta1 = timedelta()
print(delta1)
#0:00:00

delta2 = timedelta(weeks=2, seconds = 10000)
print(delta2)
#14 days, 2:46:40
```

Operações com objetos datetime

Operação	Equação	Resultado
Somar tempo	date + timedelta	date
Diminuir tempo	date - timedelta	date
Diferença entre datas	date - date	timedelta
Comparação de datas	data1 < data2	boolean

Veja alguns exemplos:

```
from datetime import datetime, timedelta

tempo = timedelta(weeks=2, hours=1)
data = datetime(2000, 1, 1)
data2 = datetime(2010, 1, 1)
print(data+tempo)
#2000-01-15 01:00:00
print(type(data+tempo))
#<class 'datetime.datetime'>
print(data - data2)
#-3653 days, 0:00:00
print(type(data - data2)
#<class 'datetime.timedelta'>
```

Como vimos anteriormente, importamos o módulo `datetime` e declaramos um objeto `timedelta` com o valor de duas semanas e uma hora. Logo após, declaramos dois objetos `datetime` com as datas `2000-1-1` e `2010-1-1`. Em seguida, realizamos as seguintes operações:

1. Adicionamos 2 semanas e uma hora a data `2000-1-1`, que resultou em `2000-01-15 01:00:00`;
2. Exibimos o tipo do objeto que resultou em `class 'datetime.datetime'`;
3. Subtraímos duas datas para ver o intervalo de tempo entre elas que resultou em `-3653 days`;
4. Por fim, exibimos o tipo do objeto que resultou em `class 'datetime.timedelta'`;

Classe timezone

Timezone é uma classe que representa um fuso horário definido por um deslocamento fixo do UTC (Tempo universal coordenado).

No entanto, para instanciar um objeto `timezone`, primeiramente você deve instanciar um objeto `timedelta`. Esse objeto `timedelta` representará o tempo em horas do fuso horário, podendo chegar a ser de 0 a 23 horas negativas ou positivas.

Veja como instanciar um objeto `timezone`.

```
from datetime import timedelta, timezone

hora = timedelta(hours=-3)
fuso = timezone(hora)
print(fuso)
#UTC-03:00
```

1. Importamos o módulo `datetime`, em seguida, instanciamos um objeto `timedelta` com valor de três horas negativas;
2. Logo após instanciamos o objeto `timezone` passando o objeto `timedelta` como argumento;
3. Assim, obtivemos o resultado `UTC-03:00`, isso indica que o fuso horário é de três horas negativas;

Exemplo de quando utilizar o módulo datetime

Você precisa saber quantos dias faltam para sua CNH (carteira nacional de habilitação) perder o prazo de validade. Sabendo que a data de vencimento é `2024-3-10`, quantos dias faltam para expirar sua CNH?
Para resolver o problema, primeiramente precisamos saber qual a data atual.

Copiar

```
from datetime import datetime

data_atual = datetime.now( )
print(data_atual)
#2022-12-20 17:38:41.588275
```

Logo após descobrir a data atual, precisamos instanciar um objeto `datetime` com a data de vencimento da CNH.


```
from datetime import datetime

data_atual = datetime.now( )
print(data_atual)
#2022-12-20 17:38:41.588275
```

Logo após descobrir a data atual, precisamos instanciar um objeto `datetime` com a data de vencimento da CNH.
Copiar

```
data_vencimento = datetime(year=2024, month=3, day=10)
```

Por fim, devemos calcular a diferença entre as datas.

```
print(data_vencimento - data_atual)
#445 days, 6:21:18.411725
```

Conclusão

Em resumo, vimos como utilizar o módulo `datetime` com alguns exemplos de código, e como ele pode ser útil na resolução de problemas com datas e horas. Para saber mais sobre o módulo `datetime`, acesse a documentação oficial do Python.