Uma introdução à

Localização (l10n) e Internacionalização (i18n)

Live de Python #290

Enquanto eu preparava este material, entrei na toca do coelho. Achei que era melhor termos duas lives boas do que uma única corrida!

Aviso

Enquanto eu preparava este material, entrei na toca do coelho. Achei que era melhor termos duas lives boas do que uma única corrida!

N	Assunto	Data
290	Teoria e internacionalização de mensagens (gettext)	28/07
292	**Localização (locale) e bibliotecas complementares**	11/08

Aviso

Roteiro

1. Uma breve revisão

Sobre padrões regionais

2. Locale

Trabalhando com padrões POSIX

3. Babel

Uma biblioteca genérica de localização

4. Algumas coisas mais

Externals e dicas

apoia.se/livedepython

pix.dunossauro@gmail.com

patreon.com/dunossauro

Ajude o projeto <3

Albano Maywitz, Alexandre Costa, Alexandre Harano, Alexandre Lima, Alexandre Takahashi, Alexandre Villares, Alfredo Braga, Alfredomorais, Alfredo Neto, Alysson Oliveira, Andre, Andre Makoski, André Oliveira, Andre Paula, Antonio Filho, Apc 16, Apolo Ferreira, Augusst0o, Augusto Domingos, Aurelio Costa, Belisa Arnhold, Beltzery, Bernardo Fontes, Bernarducs, Biancarosa, Brisa Nascimento, Bruno Batista, Bruno Bereoff, Bruno Freitas, Bruno Ramos, Bruno Russian, Bug_elseif, Canibasami, Carlos Gonçalves, Carlos Henrique, Cauã Oliveira, Celio Araujo, Christian Fischer, Claudemir Cruz, Cleiton Fonseca, Controlado, Curtos Treino, Daniel Aguiar, Daniel Brito, Daniel Bruno, Daniel Souza, Daniel Wojcickoski, Danilo Boas, Danilo Silva, Darcioalberico_sp, David Couto, Dh44s, Diego Guimarães, Dilan Nery, Dunossauro, Edgar, Elias Soares, Emerson Rafael, Érico Andrei, Esdras, Everton Silva, Ewertonbello, Fábio Belotto, Fabio Faria, Fabiokleis, Felipe Adeildo, Felipe Augusto, Felipe Corrêa, Fernanda Prado, Fernandocelmer, Ferrabras, Fichele Marias, Fightorfall, Francisco Aclima, Franklin Sousa, Frederico Damian, Fulvio Murenu, Gabriel Lira, Gabriel Mizuno, Gabriel Paiva, Gabriel Simonetto, Geilton Cruz, Geisler Dias, Giuliano Silva, Glauber Duma, Gnomo Nimp, Grinaode, Guibeira, Guilherme Felitti, Guilherme Ostrock, Gustavo Suto, Harold Gautschi, Heitor Fernandes, Hellyson Ferreira, Helton, Helvio Rezende, Henri Alves, Henrique Andrade, Henrique Machado, Henriquesebastiao, Herian Cavalcante, Hiago Couto, Idlelive, Ivan Santiago, Ivansantiagojr, Janael Pinheiro, Jean Victor, Jeferson Vitalino, Jefferson Antunes, Jerry Ubiratan, Jhonata Medeiros, Jhon Gonçalves, Joaocuginotto, João Pena, Joao Rocha, Joarez Wernke, Jonas Araujo, Jonatas Leon, Jonatas_silva11, Jose Barroso, Joseíto Júnior, José Predo), Josir Gomes, Jota_lugs, Jplay, Jrborba, Ju14x, Juan Felipe, Juli4xpy, Juliana Machado, Julio Franco, Julio Gazeta, Julio Silva, Kacoplay, Kaio Peixoto, Kakaroto, Knaka, Krisquee, Lara Nápoli, Leandro Pina, Leandro Vieira, Leonardo Mello, Leonardo Nazareth, Lilian Pires, Lisandro Pires, Lucas Carderelli, Lucas Castro, Lucasgcode, Lucas Mello, Lucas Mendes, Lucas Nascimento, Lucas Polo, Lucas Schneider, Luciano_ratamero, Luciano Ratamero, Lúcia Silva, Luidduarte, Luis Ottoni, Luiz Duarte, Luiz Martins, Luiz Paula, Luiz Perciliano, Marcelo Araujo, Marcelo Fonseca, Marcio Freitas, Marcos Almeida, Marcos Oliveira, Marina Passos, Mateusamorim96, Mateus Ribeiro, Matheus Mendez, Matheus Vian, Medalutadorespacialx, Mlevi Lsantos, Mrnoiman, Murilo Carvalho, Nhambu, Omatheusfc, Oopaze, Ostuff_fps, Otávio Carneiro, Patrick Felipe, Pytonyc, Rafael Ferreira, Rafael Fontenelle, Rafael Lopes, Rafael Romão, Raimundo Ramos, Ramayana Menezes, Ramon Lobo, Renan, Renan Sebastião, Renato José, Rene Pessoto, Renne Rocha, Ricardo Silva, Ricardo Viana, Ric_fv, Richard Sousa, Rinaldo Magalhaes, Robsonpiere, Rodrigo Barretos, Rodrigo Santana, Rodrigo Vieira, Rogeriocampos7, Rogério Nogueira, Rui Jr, Rwallan, Samael Picoli, Samanta Cicilia, Santhiago Cristiano, Scrimf00x, Scrimfx, Sherlock Holmes, Shinodinho, Shirakawa, Tarcísio Miranda, Tenorio, Téo Calvo, Teomewhy, Tharles Andrade, Thiago, Thiago Araujo, Thiago Lucca, Thiago Paiva, Tiago, Tomás Tamantini, Trojanxds, Valdir, Varlei Menconi, Viniciusfk9, Vinicius Silva, Vinicius Souza, Vinicius Stein, Vladimir Lemos, Waltennecarvalho, Williamslews, Willian Lopes, Will_sca, Xxxxxxxxx, Zero! Studio

Obrigado você

Uma breve

Revisão

O que vimos na 290?

Conversamos um pouco sobre a parte teórica. O que define:

- **l10n**: Aplicação única adaptada para atuar em diferentes países e idiomas
  - **Língua**:
    - Traduções
  - **Padrões regionais**:
    - Tempo, data, moedas, medidas, ordenação
- **i18n**: Construir um codebase "adaptável"
  - Traduções: **gettext**, vimos na live passada
  - Padrões regionais: **locale**, viemos aqui pra isso!

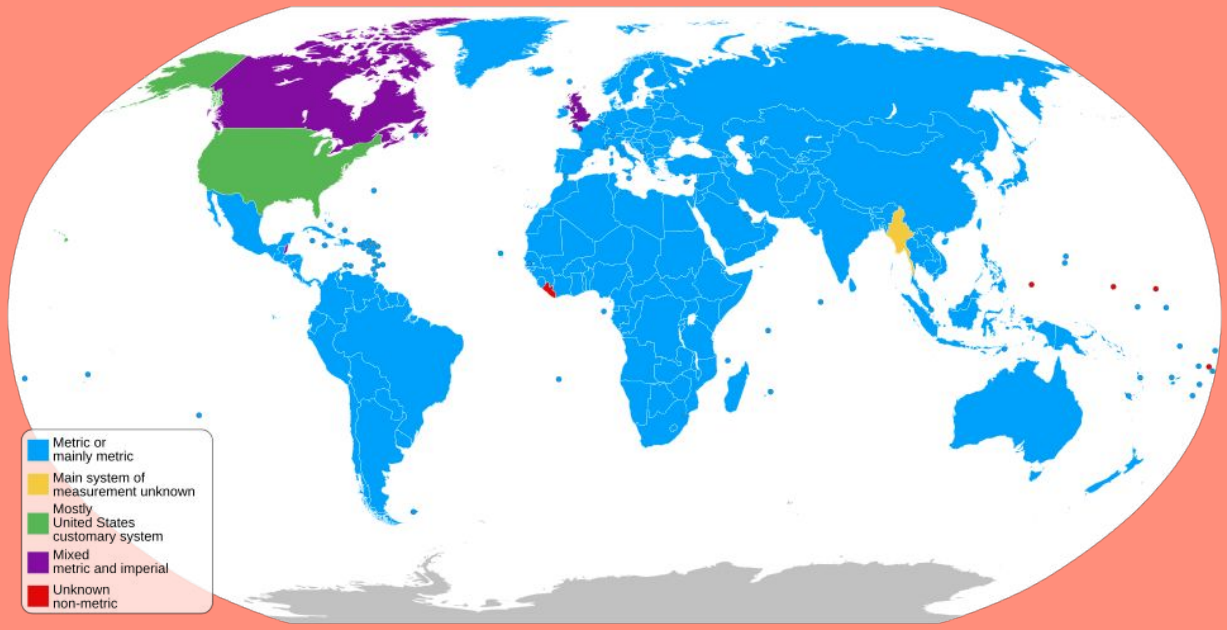
"Padrões" regionais

Os "**padrões regionais**" que devem ser levados em conta. Como:

- **Medidas**
- **Datas**
- **Moeda**
- **Telefones**
- **Ordenação**
- E muito mais...

"Padrões" regionais

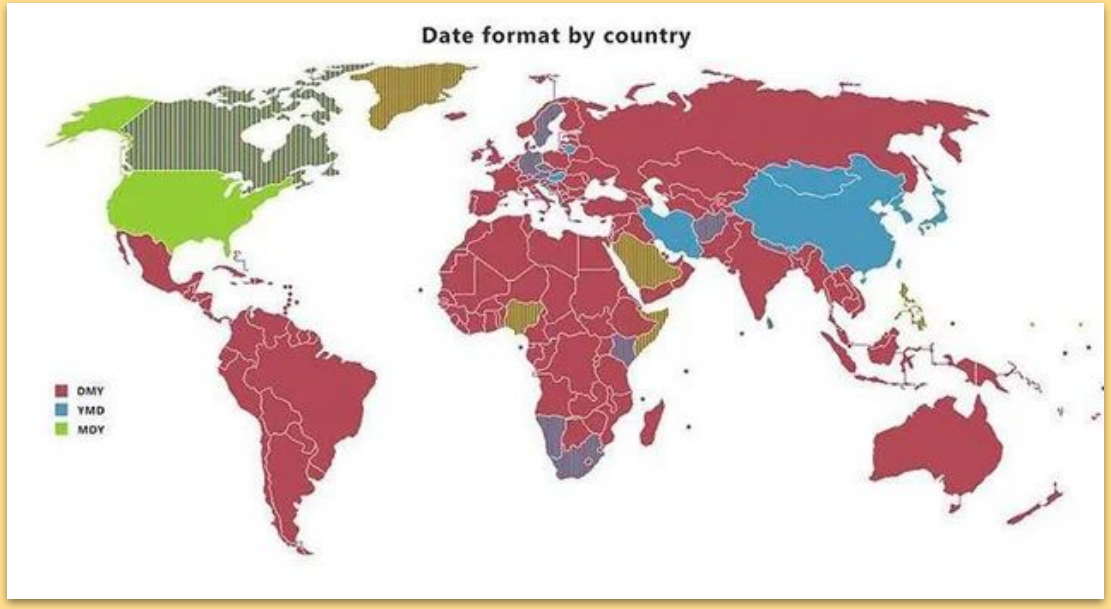
- **Medidas**: Diferentes locais usam sistemas de unidades diferentes
  - Sistema métrico (metros, litros, quilos)
  - Sistema imperial (pés, galões, libras)
- Datas
- Moeda
- Telefones
- Ordenação
- E muito mais...



https://pt.m.wikipedia.org/wiki/Ficheiro:Metric_and_imperial_systems_%282019%29.svg

"Padrões" regionais

- Medidas
- **Datas**: Os formatos de datas variam dependendo da região
  - No Brasil, usamos dia/mês/ano
  - China: ano/mês/dia.
- Moeda
- Telefones
- Ordenação
- E muito mais...



"Padrões" regionais

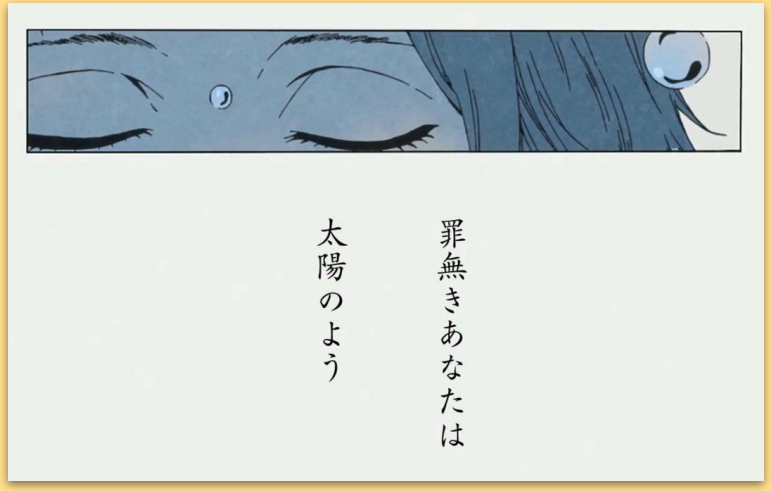
- Medidas
- Datas
- **Moeda**: A moeda varia conforme a região.
  - Real no Brasil
  - Euro na Europa
  - Dólar nos Estados Unidos
- Telefones
- Ordenação
- E muito mais...

"Padrões" regionais

- Medidas
- Datas
- Moeda
- **Telefones**: Em localidades diferentes, os telefones têm formatos diferentes.
  - Na Índia, por exemplo, temos códigos de área de quatro dígitos
- Ordenação
- E muito mais...

"Padrões" regionais

- Medidas
- Datas
- Moeda
- Telefones
- **Ordenação**: Dependendo do alfabeto usado, as regras de ordenação são diferentes.
- **E muito mais...**
  - **Direção do texto**
  - **Linhas ou colunas**
  - **Charset**
  - **...**



Poema - Masaki (Bleach)

Locale

codando os padrões

locale

locale em Python segue parcialmente o padrão **POSIX**, oferecendo suporte às funcionalidades mais comuns para detectar e configurar localidades (locales).

Lida com aspectos regionais, como:

- Formatos de datas
- Números
- Moedas e
- Ordenação
- ...

Indo além do idioma em si. Fornecendo um toolkit genérico para trabalhar com essas configurações de ambiente.

- Locale é de "baixo nível"
- Das 19 plataformas suportadas (PEP 11), cada uma funciona **diferente**
- Windows não implementa POSIX
  - Entre o NT 4 e o windows 8.1 existiu o "Subsystem for UNIX-based Applications [SUA]"
  - Agora é BCP 47, do vista pra frente
  - Mas, usa LCDI para localidades em conjunto (que será deprecada)

⚠ Observações antes de avançar ⚠

Se precisar fazer locale no windows / POSIX ao mesmo tempo, recomendo **fortemente** que use o **babel**

⚠️ Observações antes de avançar ⚠️

Obtendo a localidade

Diante de várias funções e constantes providas pelo **locale** acho que devemos considerar primeiro as funções referentes à localidade em geral:

- **getlocale**(): Fornece uma tupla com o idioma_Localização e o charset
- **setlocale**(): Aplica uma localização específica.

getlocale

A função **getlocale** vai nos retornar o locale atual:

posix

```
>>> import locale
>>> locale.getlocale()
('en_US', 'UTF-8')
```

Windows

```
>>> locale.getlocale()
('English_United States', '1252')
```

LCID vs POSIX

Não existe uma forma única de pegar o valor do locale em LCID e converter em POSIX, para criarmos e usarmos os arquivos **.po** em múltiplos sistemas operacionais. Mas, para obter esse código, podemos usar **ctypes**:

```
>>> import ctypes
>>> import locale
>>> ctypes.windll.kernel32.GetUserDefaultLCID()
1033
>>> locale.windows_locale.get(ctypes.windll.kernel32.GetUserDefaultLCID())
'en_US'
```

LCID vs POSIX

Não existe uma f
converter em POS
múltiplos sistema
usar **ctypes**:

> Existe a função **locale.getdefaultlocale**() que faz essa tradução de forma automática, porém ela será removida na versão 3.15.
> **Embora exista uma discussão ativa sobre esse tema**

> Lembrando que o LCID está sendo descontinuado também nos sistemas. Mas, não consegui até o momento encontrar uma referencia de como pegar o BCP 47 no windows.

```
>>> import ctypes
>>> import locale
>>> ctypes.windll.kernel32.GetUserDefaultLCID()
1033
>>> locale.windows_locale.get(ctypes.windll.kernel32.GetUserDefaultLCID())
'en_US'
```

setlocale

setlocale adapta o sistema a uma determinada regionalidade. Isso significa que, ao configurá-la, o comportamento do sistema em relação a datas, números, moedas e outros formatos regionais serão alterados conforme o locale definido.

```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')
```

Assim, o sistema "reage" ao valor passado, aplicando as regras daquela localidade.

*** setlocale se prolifera no sistema!**

Variações no setlocale

setlocale pode ser aplicado de diferentes maneiras, dependendo da área que você deseja configurar. O primeiro parâmetro define qual aspecto da localidade será alterado:

- **LC_ALL**: modifica todas as áreas de localidade, incluindo data, hora, moeda, números, entre outras.
- **LC_TIME**: altera a formatação de data e hora.
- **LC_MONETARY**: modifica o formato de valores monetários, incluindo a moeda e os separadores decimais.
- **LC_NUMERIC**: define como os números serão formatados, como o uso de vírgula ou ponto como separador decimal.
- **LC_COLLATE**: controla a ordenação de strings (importante para buscas e classificações).

Exemplos

Por exemplo, se você quiser mudar somente o formato de datas, pode configurar o **LC_TIME**:

```
import locale
locale.setlocale(locale.LC_TIME, 'en_US.UTF-8')
```

Ou, somente a parte de moedas para o Brasil:

```
import locale
locale.setlocale(locale.LC_MONETARY, 'pt_BR.UTF-8')
```

Formatação de datas

Algumas constantes são ofertadas no módulo como **D_T_FMT**, **D_FMT** e **T_FMT**, que garantem que as datas sejam apresentadas corretamente conforme a localidade configurada no sistema.

```python
import locale, datetime

locale.setlocale(locale.LC_TIME, 'pt_BR.UTF-8')
data = datetime.datetime(2025, 7, 21, 14, 55, 2)

print(data.strftime(locale.nl_langinfo(locale.D_T_FMT)))
# seg 21 jul 2025 14:55:02

print(data.strftime(locale.nl_langinfo(locale.D_FMT)))
#21/07/2025

print(data.strftime(locale.nl_langinfo(locale.T_FMT)))
#14:55:02
```

Formatação de datas

Algumas constantes são ofertadas no módulo como **D_T_FMT**, **D_FMT** e **T_FMT**, que garantem que as datas sejam apresentadas corretamente conforme a localidade configura

nl_langinfo só existe em sistemas posix

```
import locale, datetime

locale.setlocale(locale.L
data = datetime.datetime(

print(data.strftime(locale.nl_langinfo(locale.D_T_FMT)))
# seg 21 jul 2025 14:55:02

print(data.strftime(locale.nl_langinfo(locale.D_FMT)))
#21/07/2025

print(data.strftime(locale.nl_langinfo(locale.T_FMT)))
#14:55:02
```

Formatação de moedas

Podemos usar o **currency()** para formatar valores monetários. Formatando o valor com o símbolo da moeda e a separação correta de milhares e decimais, conforme a convenção local.

```
locale.setlocale(locale.LC_MONETARY, 'pt_BR.UTF-8')

print(locale.currency(123456.789))
# R$ 123456,79

# Agrupando os milhares
print(locale.currency(123456.789, grouping=True))
# R$ 123.456,79

# Sem o símbulo respectivo da moeda
print(locale.currency(123456.789, symbol=False))
# 123456,79
```

Formatação de números

O separador de milhares e o símbolo para a vírgula ou ponto como separador decimal são exemplos de diferenças que podem ocorrer entre as localidades.

```
locale.setlocale(locale.LC_NUMERIC, 'pt_BR.UTF-8')
print(locale.format_string("%0.2f", 1234567.89, grouping=True))
# 1.234.567,89
```

Ordenação de strings (collation)

Ordenar textos corretamente em diferentes idiomas é mais complexo do que parece. Cada idioma tem regras próprias para comparar letras, principalmente com acentos e caracteres especiais. Para isso, usamos **strxfrm** em conjunto ao **sorted**. ***(Existem uma série de limitações no strxfrm)***

```
locale.setlocale(locale.LC_COLLATE, 'pt_BR.UTF-8')

lista = ['zebra', 'ação', 'abacaxi', 'Árvore']
lista_ordenada = sorted(lista, key=locale.strxfrm)
print(lista_ordenada)
# ['abacaxi', 'ação', 'Árvore', 'zebra']
```

Pontos onde o locale não atua

É importante notar que nem todos os padrões regionais são cobertos pelo locale, mesmo nos sistemas POSIX:

- Unidades de medidas
- Fuso horários
- Direcionalidade de texto
- Calendários não gregorianos
- Formatação de endereços e nomes
- Formatação de telefones
- Informações gerais de localidade
- ...

Algumas opções pra isso

- Unidades de medidas
  - Pint
- Fuso horários
  - Pendulum (ldp #137), Pytz, datepaser (para ler)
- Direcionalidade de texto
  - Babel**, python-bidi
- Calendários não gregorianos
  - Babel**, Converdate
- Formatação de endereços e nomes
  - Não conheço nenhuma ativa :(
- Formatação de telefones
  - phonenumbers
- Ordenação
  - Babel**, pyUCA, PyICU
- Informações gerais de localidade
  - pycountry

** Existe suporte, mas não completo

Exemplo do Pint

```python
from pint import Quantity  # pip install pint

kms = Quantity(45, 'kilometer')
miles = kms.to('miles')
print(kms, miles) # 45 kilometer 27.96170365068003 mile

temp_c = Quantity(30, 'celsius')
temp_f = temp_c.to('fahrenheit')
print(temp_c, temp_f) # 30 degree_Celsius 85.99999999999993 degree_Fahrenheit

peso_kg = Quantity(70, 'kilogram')
peso_lb = peso_kg.to('pound')
print(peso_kg, peso_lb) # 70 kilogram 154.3235835294143 pound
```

Exemplo pendulum

```python
import pendulum  # pip install pendulum

dt_meu = pendulum.now(tz='America/Sao_Paulo')
dt_will = dt_meu.in_timezone('Europe/Berlin')
print(dt_meu, '\n', dt_will)
# 2025-08-10 22:27:38.766820-03:00
# 2025-08-11 03:27:38.766820+02:00
```

Um projetinho

Prática

Assunto deveras abstrato

Então vamos usar uma aplicaçãozinha que fiz especialmente para essa live.

Sempre que aprendermos um conceito, vamos aplicar nesse projeto.

projetinho de moedas

Pegando a localização atual

Se precisamos trabalhar com a localização do sistema. Podemos fazer algo parecido com isso (lembrando que nem todos os sistemas são POSIX)

```
localization, encode = locale.getlocale()

locale.setlocale(locale.LC_ALL, f'{localization}.{encode}')

t = translation('messages', localedir='locale', fallback=True)
t.install(f'{localization}.{encode}')
_ = t.gettext
```

Definição de localização manual

Isso serve para nosso CLI, mas existem diversas formas de pegar esse input. Em GUI, idioma do navegador e etc...

```
cli.add_argument(
    '--language',
    '-l',
    choices=['pt_BR', 'en_US', 'eo'],
    default=localization,
    help=_('Preferred languange for the answer'),
)
```

Atualizando a localização

Aqui o problema de "contaminação" do locale fica evidente.

```
if args.language != localization:
    locale.setlocale(locale.LC_ALL, f'{args.language}.{encode}')
    t = translation(
        'messages',
        localedir='locale',
        languages=[args.language],
        fallback=True,
    )
    t.install()
    _ = t.gettext
```

Trabalhando os valores

```
print(
    _('Last trade: {trade_time}').format(
        trade_time=datetime.strftime(
            trade_time, locale.nl_langinfo(locale.D_T_FMT)
        )
    )
)

print(
    _('Bid price: {price}').format(
        price=locale.currency(Decimal(parsed['bid']), grouping=True)
    )
)
```

Resolvendo problemas sem baterias

Babel

Babel

Babel é uma caixinha de ferramentas para internacionalização e localização com **foco em aplicações web**.

Permite formatar dados como números, datas, horas, e textos conforme as convenções de diferentes idiomas e regiões, ...

- Criada por: Armin Ronacher
- Licença: BSD-3-Clause
- Primeira release: 06/2007
- Release atual: 2.17.0 (2/2025)
- ***Em python puro (funciona em todas as plataformas)***

Ferramentas do Babel

O Babel é distribuído em módulos especializados:

- **Locale**: módulo base
- **dates**: para formatação de datas e horas
- **numbers**: para formatação de números
- **Catalogo de mensagens**: (GNU/gettext) – *não vamos ver...*
  - CLI em python – sem dependência de xgettext
- ...

Locale: O módulo base

O módulo **Locale** permite trabalhar com convenções regionais. Com ele, podemos obter o nome do idioma, a moeda, entre outros aspectos específicos de uma região. O que é a base para chamadas seguintes

```
from babel import Locale

locale = Locale('pt', 'BR')

locale.display_name  # Português (Brasil)
locale.territory_name # Brasil

l2 = Locale.parse('pt')
l2.display_name # Português
```

babel.dates

Também é possível formatar horas com o Babel, utilizando o mesmo objeto de localidade

```python
from babel.dates import LOCALTZ, format_date, format_datetime, format_time

locale = Locale('pt', 'BR')
hora = datetime(2023, 8, 11, 15, 30, tzinfo=LOCALTZ)

format_date(hora, locale=locale)      # '11 de ago. de 2023'
format_time(hora, locale=locale)      # 15:30:00
format_datetime(hora, locale=locale)  # '11 de ago. de 2023 15:30:00'

format_date(hora, format='short', locale=locale) # 11/08/2023
format_date(hora, format='full', locale=locale)
# sexta-feira, 11 de agosto de 2023
```

babel.numbers

A biblioteca de números trabalha com as moedas também

```python
from babel.numbers import format_currency, format_decimal, parse_decimal
from decimal import Decimal

locale = Locale('pt', 'BR')
numero = Decimal('12345.67')

format_decimal(numero, locale=locale)  # 12.345,67

format_currency(numero, currency='BRL', locale=locale) # R$ 12.345,67

parse_decimal('12.345,67', locale=locale)  # 12345.67
```

Adaptando para o babel

Então vamos usar uma aplicaçãozinha que fiz especialmente para essa live.

Sempre que aprendermos um conceito, vamos aplicar nesse projeto.

projetinho de moedas

Extras

Cousitas mais...

Bibliotecas adicionais que podem te ajudar

- Humanize: Para tratar respostas "duras" de código em uma forma humana de mostrar a informação (tem vídeo no canal)
- "sualib"-i18n: Diversas bibliotecas tem personalização das mensagens padrões, como:
  - fastapi-i18n
  - pydantic-i18n
  - ...

Referências

1. ESSELINK, Bert; ESSELINK, Bert (ORGS.). A Practical guide to localization. Amsterdam Philadelphia: John Benjamins Pub. Co, 2000.
2. FREE STANDARDS GROUP. OpenI18N 1.3. , 2003. Disponível em: <http://openi18n.web.fc2.com/numa/OpenI18N1.3.pdf>
3. Free Standards Group. , 10 mar. 2025. (Nota técnica).
4. LEARN MICROSOFT. POSIX and UNIX Support in Windows. Disponível em: <https://learn.microsoft.com/en-us/archive/technet-wiki/10224.posix-and-unix-support-in-windows>. Acesso em: 23 jul. 2025.
5. LINUX STANDARD BASE. The Open Group Base Specifications Issue 8, 2018 edition. Disponível em: <https://pubs.opengroup.org/onlinepubs/9799919799/nframe.html>. Acesso em: 20 jul. 2025.
6. [MS-LCID]: Windows Language Code Identifier (LCID) Reference. Disponível em: <https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-lcid/70feba9f-294e-491e-b6eb-56532684c37f>. Acesso em: 21 jul. 2025.
7. PEP 11 – CPython platform support. Disponível em: <https://peps.python.org/pep-0011/>. Acesso em: 23 jul. 2025.
8. PYTHON. Deprecation of locale.getdefaultlocale breaks POSIX compatibility on Windows platform · Issue #130796 · python/cpython. Disponível em: <https://github.com/python/cpython/issues/130796>. Acesso em: 21 jul. 2025.
9. PYTHON SOFTWARE FOUNDATION. cpython/Modules/_localemodule.c at main · python/cpython · GitHub. Disponível em: <https://github.com/python/cpython/blob/777159fa318f39c36ad60039cdf35a8dbb319637/Modules/_localemodule.c>. Acesso em: 23 jul. 2025a.
10. PYTHON SOFTWARE FOUNDATION. Internacionalização. Documentação. Disponível em: <https://docs.python.org/3/library/i18n.html>. Acesso em: 20 jul. 2025c.
11. PYTHON SOFTWARE FOUNDATION. locale — Serviços de internacionalização. Disponível em: <https://docs.python.org/3/library/locale.html>. Acesso em: 20 jul. 2025d.