# Stringy, regulární výrazy a parsování textu

*Předmluva: Nejdříve si přečtěte první stranu, poté si můžete vyzkoušet metody, které jsou zmíněné na straně 2. Regulární výrazy je ideální si v programování vyzkoušet (Visual Studio), a tím pochopit, jak fungují.*

*Vhodné videa pro pochopení regulárních výrazů:*

[*AJ video - 40m všechno vysvětlí*](https://www.youtube.com/watch?v=sa-TUpSx1JA)

[*AJ video z kterého jsem se učil já - 20m*](https://www.youtube.com/watch?v=rhzKDrUiJVk)

## Zpracování a pasování dat

Se stringem můžeme pracovat jako s polem a tedy vybírat si jednotlivé znaky pomocí [index]

1. Rozdělení stringu

Rozdělení řetězce na podřetězce může být užitečné v mnoha situacích. V Pythonu můžete použít metodu split(), která rozdělí řetězec na podřetězce podle zadaného oddělovače. Například, následující příklad rozdělí řetězec text na slova a uloží je do seznamu.

text = "The quick brown fox jumps over the lazy dog"

words = text.split()

print(words)

#nebo

new\_text = text[1:2] #"h"

new\_text = text[2:] #"e quick brown fox jumps over the lazy dog"

new\_text = text[:2] #"Th"

1. Spojení stringů

Spojení dvou nebo více řetězců do jednoho řetězce může být také užitečné. V Pythonu můžete použít operátor + pro spojení dvou řetězců. Například, následující příklad spojí řetězce first\_name a last\_name a uloží je do řetězce full\_name.

first\_name = "John"

last\_name = "Doe"

full\_name = first\_name + " " + last\_name

print(full\_name)

strings=[“as”,”sa”]

new\_text = strings.join(“==”) #"as==sa"

name = "John Doe"

new\_string = name.replace(“Doe”,”Novak”) vsechny

new\_string = name.replace(“Doe”,”Novak”,1) jenom jendou

## Regulární výrazy

Regulární výrazy vznikly z důvodu potřeby práce s textovými řetězci určitým unifikovaným způsobem. Základní myšlenkou je porovnání textového řetězce s předpřipraveným vzorem. Nejčastěji se zjišťuje, zda řetězec odpovídá vzoru, lze také rozdělit řetězec na části (rozdělovačem je zadaný vzor) nebo části řetězce odpovídajícího vzoru nahradit jiným řetězcem.

Jednotlivé konkrétní znaky

|  |  |
| --- | --- |
| x | Konkrétní znak |
| [abc] | Jednoduchá **množina**(jeden ze znaků *a b c*) |
| [^abc] | Žádný ze znaků *abc*(**negace**) |
| [a-zA-Z] | **Rozsah** znaků |
| . (tečka) | **Libovolný znak** s výjimkou konce řádku |
| ^fugo | Najde „*fugo*“ které musí být **na začátku řádku** |
| fugo$ | Najde „*fugo*“ které musí být **na konci řádku** |
| X|Z | Najde *X* **nebo** *Z* |
| XZ | Najde *X* **následované** písmenem *Z* |
| [abc][vz] | Najde *a* nebo *b* nebo *c* následované písmenem *v* nebo *z* |

Zástupné znaky

|  |  |
| --- | --- |
| \d | Jakákoliv **číslice** |
| \D | Cokoliv kromě číslic(negace) |
| \s | Jakýkoliv bílý znak (mezera) |
| \S | Cokoliv kromě bílých znaků (mezer) |
| \w | Libovolný písmenný znak(vč. číslic) |
| \W | negace \w |

Znaky pro vyjádření počtu

|  |  |
| --- | --- |
| \* | Vyskytne se „nulakrát“ a více, zkratka pro {0,} |
| + | Vyskytne se jednou a více, zkratka pro {1,} |
| ? | Vyskytne se jednou nebo vůbec, zkratka pro {0,1} |
| {n} | Vyskytne se n-krát |
| {n,m} | Vyskytne se v rozmezí od n do m – krát |

Třída re

import re

m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")

**>>>** m[0] *# The entire match*

'Isaac Newton'

**>>>** m[1] *# The first parenthesized subgroup.*

'Isaac'

**>>>** m[2] *# The second parenthesized subgroup.*

'Newton'

re.search(pattern, string, flags=0)

re.match(pattern, string, flags=0)

re.fullmatch(pattern, string, flags=0)

re.split(pattern, string, maxsplit=0, flags=0)

re.findall(pattern, string, flags=0)

re.sub(pattern, repl, string, count=0, flags=0)

* re.search(pattern, string, flags=0) hledá v řetězci string první výskyt podřetězce, který odpovídá regulárnímu výrazu pattern. Pokud shoda není nalezena, vrací None.
* re.match(pattern, string, flags=0) hledá v řetězci string shodu s regulárním výrazem pattern od začátku řetězce. Pokud shoda není nalezena, vrací None.
* re.fullmatch(pattern, string, flags=0) hledá v řetězci string shodu s regulárním výrazem pattern, která musí odpovídat celému řetězci. Pokud shoda není nalezena, vrací None.
* re.split(pattern, string, maxsplit=0, flags=0) rozděluje řetězec string na podřetězce podle regulárního výrazu pattern. Parametr maxsplit určuje maximální počet rozdělení, které se mají provést. Výsledkem je seznam podřetězců.
* re.findall(pattern, string, flags=0) hledá v řetězci string všechny nekřížící se shody s regulárním výrazem pattern a vrací je jako seznam.
* re.sub(pattern, repl, string, count=0, flags=0) nahrazuje všechny výskyty regulárního výrazu pattern v řetězci string řetězcem repl. Parametr count určuje maximální počet nahrazení, které se mají provést. Výsledkem je řetězec s provedenými nahrazeními

Příklady užitečných reg. výrazů

**Devítimístné telefonní číslo s CZ nebo SK předvolbou**

@"^(\+420|\+421) ?[0-9]{3} ?[0-9]{3} ?[0-9]{3}$"

**IP adresa**

@"^((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))$"

## Kodování a stringy

Kodovani

Python podporuje různá kódování pro reprezentaci znaků. Standardní kódování v Pythonu je UTF-8, což je rozšířené kódování Unicode. To znamená, že Python podporuje celou řadu jazyků a znakových sad, včetně latinky, cyrilice, arabštiny, čínštiny a mnoha dalších.

Je důležité dbát na kódování při interakci s externími zdroji dat, jako jsou soubory, databáze nebo webové stránky. Pokud jsou data v jiném kódování, může dojít k problémům s interpretací a ztrátě informací.

Pro práci s různými kódováními v Pythonu se používají funkce pro převod řetězců do a ze specifických kódování, například:

* **encode(encoding)** - převede řetězec do zadaného kódování
* **decode(encoding)** - převede řetězec z daného kódování do Unicode

# převedení řetězce na bajty v UTF-8

s = "Hello, world!"

b = s.encode("utf-8")

print(b) # b'Hello, world!'

# převedení bajtů na řetězec v UTF-8

s2 = b.decode("utf-8")

print(s2) # Hello, world!

# zápis řetězce do souboru s kódováním UTF-8

with open("output.txt", "w", encoding="utf-8") as f:

f.write(s)

s = "Hello, world!"

b = bytes(s, 'utf-8') # převod řetězce na bajty v kódování UTF-8

print(b) # b'Hello, world!'

s = "Hello, world!"

ba = bytearray(s, 'utf-8') # převod řetězce na bytearray v kódování UTF-8

print(ba) # bytearray(b'Hello, world!')

String

**String je řetězec znaků(chars), to znamená, že se objekt String chová jako pole. Lze na něj uplatnit hledání podle indexu(písmenka-znaku) jako v poli.**

Můžeme použít jak jednoduchých ('...') nebo dvojitých ("...") uvozovek.

string1 = "Hello"

string2 = "world"

result = string1 + " " + string2

print(result) # vypíše "Hello world"

string = "ha"

result = string \* 5

print(result) # vypíše "hahahahaha"

string = "The quick brown fox jumps over the lazy dog"

result = "brown" in string

print(result) # vypíše True

string = "Hello, World!"

result = string.replace("World", "Python")

print(result) # vypíše "Hello, Python!"

string = "Hello, World!"

result = string[7:12]

print(result) # vypíše "World"

string = "apple,banana,orange"

result = string.split(",")

print(result) # vypíše ["apple", "banana", "orange"]

String je **neměnný objekt** – po vytvoření instance objektu String jej není možné měnit. Třída sice obsahuje několik metod, které by mohly vypadat, že objekt pozmění, ty ale vytvoří nový objekt a nahradí jím ten starý.

* Je v něm zachováno pořadi ve kterém jse prvky do kolekce vkládají
* Jako přístup k hodnotám se pužívají indexy
* Pomocí len() můžeme zjistit velikost kolekce

Porovnávání stringů:

string1 = "hello"

string2 = "Hello"

result1 = string1 == string2

result2 = string1 != string2

result3 = string1 < string2

result4 = string1 > string2

print(result1) # vypíše False

print(result2) # vypíše True

print(result3) # vypíše True

print(result4) # vypíše False

string = "Hello, world!"

result1 = string.startswith("Hello")

result2 = string.endswith("world!")

result3 = string.find("world")

result4 = string.index("Python") # vyhodí výjimku ValueError, protože "Python" není v řetězci

result5 = "Hello" in string

print(result1) # vypíše True

print(result2) # vypíše True

print(result3) # vypíše 7

print(result5 ) # vypíše True