EDICE
myslíme v ...

knihovna programátora

- Nejdůležitější zabudované i externí knihovny
- Podrobný výklad práce s regulárními výrazy
- Práce s texty ve formátu JSON a CSV
- Virtuální prostředí a práce s ním
- Knihovny NumPy, Matplotlib a pandas
- Práce s textovými i binárními daty

Knihovny pro práci s daty

PRO VERZI 3.11

Python

RUDOLF PECINOVSKÝ

GRADA

GRADA

EDICE
myslíme v ...

knihovna programátora

RUDOLF PECINOVSKÝ

Python

Knihovny pro práci s daty
PRO VERZI 3.11

GRADA
Publishing

Rudolf Pecinovský

Python – knihovny pro práci s daty
PRO VERZI 3.11

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401
jako svou 8709. publikaci

Odpovědný redaktor Petr Somogyi
Grafická úprava a sazba Rudolf Pecinovský
Počet stran 328
První vydání, Praha 2023
Vytiskla TISKÁRNA V RÁJI, s.r.o., Pardubice

ISBN 978-80-271-6715-9 (pdf)
ISBN 978-80-271-0659-2 (print)

Mé ženě Jarušce a dětem
Štěpánce, Pavlínce, Ivance a Michalovi

Stručný obsah

Úvod **23**

Část A Práce se stringy 29

1 Metody třídy `str` **30**
2 Úvod do regulárních výrazů **44**
3 Využití metaznaků **54**
4 Základy práce se skupinami **64**
5 Náhledy a náhrady **75**
6 Použití regulárních výrazů v programu **84**
7 Bajtové objekty **95**

Část B Vstup a výstup dat 101

8 Základní informace o souborech **102**
9 Modul pathlib a abstraktní cesty **111**
10 Konkrétní cesty a práce se složkami a soubory **122**
11 Datové proudy **130**
12 Čtení a zápis dat **146**

Část C Základy práce s daty 159

13 Datové třídy **160**
14 Doprovodný program **177**
15 Dokumenty ve formátu JSON **187**
16 Dokumenty ve formátu CSV **203**
17 Další užitečné moduly **220**

Část D Důležité externí knihovny 237

18 Virtuální prostředí a externí knihovny 238

19 Knihovna NumPy 250

20 Knihovna Matplotlib 275

21 Knihovna pandas 299

Literatura 323

Rejstřík 325

Podrobný obsah

Úvod **23**
Komu je kniha určena **23**
Koncepce výkladu a jeho uspořádání **24**
První část: Práce se stringy 24
Druhá část: Vstup a výstup dat 24
Třetí část: Základy práce s daty 24
Čtvrtá část: Důležité externí knihovny 24
Jazyk identifikátorů **24**
Potřebné vybavení **25**
Doprovodné programy **25**
Použité typografické konvence **25**
Odbočka – podšeděný blok **27**
Zpětná vazba **27**

Část A Práce se stringy 29

1 Metody třídy `str` **30**
1.1 Efektivnější spojování stringů **30**
1.2 Rozdělování stringů na části **31**
`str.split (sep=None, maxsplit=-1) -> list[str...]`
`str.rsplit(sep=None, maxsplit=-1) -> list[str]` 31
`partition (sep:str) -> tuple[str,str,str] rpartition(sep:str) ->`
`tuple[str,str,str]` 32
`str.splitlines(keepends=False) -> list[str]` 32
1.3 Jednoduché úpravy **33**
Ořezávání stringů **33**
Nahrazování **34**
`str.replace(old:str, new:str, count:int=0) -> str` 34
Příklad 34
`str.expandtabs(tabsize=8)` 35
Vyhledávání **35**
`str.count(sub:str, start:int=0, end:int=None, /)` 36
`str.find(sub:str, start:int=0, end:int=None, /) str.rfind(sub:str,`
`start:int=0, end:int=None, /)` 36
`str.index(sub:str, start:int=0, end:int=None, /)`
`str.rindex(sub:str, start:int=0, end:int=None, /)` 36
`str.startswith(prefix:(str|tuple[str]), start:int=0, end:int=None,`
`/) str.endswith (suffix:(str|tuple[str]), start:int=0,`
`end:int=None, /)` 36
1.4 Jednoduché formátování **36**
Změna velikosti písmen **37**
Zarovnávání **37**

center(width, fillchar=' ', /) 37
ljust(width, fillchar=' ', /) 37
rljust(width, fillchar=' ', /) 37
Šablony 38
template 38
substitute(mapping={}, /, **kwds) 38
safe_substitute(mapping={}, /, **kwds) 38
Dokonalejší formátování 39
1.5 Zjišťovací metody 39
1.6 Surové stringy 41
Pravidla zápisu surových stringů 42
1.7 Zdrojové kódy 43
2 Úvod do regulárních výrazů 44
2.1 Co jsou regulární výrazy 44
Historie 45
2.2 Testovací programy 45
Testery na webu 46
Doprovodný GUI tester 47
Doprovodný textový tester v modulu rtc 48
2.3 Standardní znaky a metaznaky 50
Zadání metaznaku jako standardního znaku 51
2.4 Vyhledání textu odpovídajícího regulárnímu výrazu 52
2.5 Zdrojové kódy 53
3 Využití metaznaků 54
3.1 Množiny znaků 54
3.2 Skupinové metaznaky 56
3.3 Hranice 58
3.4 Režimy 58
A; ASCII 58
L; LOCALE 58
U; UNICODE 59
I; IGNORECASE 59
M; MULTILINE 59
S; DOTALL 59
X; VERBOSE 59
DEBUG 59
NOFLAG 59
3.5 Kvantifikátory (počet opakování) 60
Agresivita kvantifikátorů 61
3.6 Alternativa 63
3.7 Zdrojové kódy 63
4 Základy práce se skupinami 64
4.1 Skupiny 64
Odvolávky na skupinu 65
Odvolávky následované číslicí 66
Alternativa a skupiny 67
4.2 Otazníkové skupiny 68
4.3 Pojmenování skupiny 68
4.4 Komentáře 69
4.5 Změny režimu 70
Dočasné nastavení režimu 72
4.6 Obecné nezachytávané skupiny 73
4.7 Zdrojové kódy 74

5 Náhledy a náhrady 75
5.1 Náhled vpřed 75
Zajímavý příklad 76
5.2 Náhled vzad 77
5.3 Rozhodování 78
5.4 Atomické skupiny 79
5.5 Nahrazování 80
Použití pojmenovaných skupin 81
Další příklady 81
5.6 Zdrojové kódy 83
6 Použití regulárních výrazů v programu 84
6.1 Konstanty v modulu `re` 84
6.2 Třídy `re.Pattern` a `re.Match` 85
Třídy `re.Pattern` 85
`re.compile(pattern:str, flags:int=0) -> Pattern` 85
Funkce modulu `re` versus metody třídy `Pattern` 85
Odchylky v argumentech funkcí modulu `re` a metod třídy `Pattern` 86
Třída `re.Match` a její instance 86
6.3 Vyhledávací funkce a metody 86
`re.search(pattern:str, string:str, flags:int=0) -> Match | None`
`Pattern.search(string:str[, pos:int[, endpos:int]])` 87
`re.match(pattern, string, flags=0) -> Match | None`
`Pattern.match(string[, pos[, endpos]])` 87
`re.fullmatch(pattern, string, flags=0) -> Match | None`
`Pattern.fullmatch(string[, pos[, endpos]])` 87
`re.findall(pattern, string, flags=0) Pattern.findall(string[, pos[,`
`endpos]])` 87
`re.finditer(pattern, string, flags=0) Pattern.finditer(string[,`
`pos[, endpos]])` 87
`re.split(pattern, string, maxsplit=0, flags=0)`
`Pattern.split(string, maxsplit=0)` 87
6.4 Atributy objektů typu `re.Match` 89
Datové atributy 89
`Match.string` 89
`Match.re` 89
Funkční atributy – metody 89
`Match.group([group1, ...])` 89
`Match.__getitem__(g)` 89
`Match.groups(default=None)` 89
`Match.groupdict(default=None)` 89
`Match.start(group=0) Match.end (group=0)` 90
`Match.span([group])` 91
`Match.expand(template)` 91
Příklady 91
6.5 Nahrazovací funkce a metody 92
`re.sub(pattern, repl, string, count=0, flags=0) Pattern.sub(repl,`
`string, count=0)` 92
`re.subn(pattern, repl, string, count=0, flags=0) Pattern.subn(repl,`
`string, count=0)` 92
6.6 Další atributy modulu `re` a jeho tříd 93
Atributy modulu `re` 93
`re.escape(pattern)` 93
Atributy instancí třídy `Match` 93

Match.pos ... 93
Match.endpos ... 93
Match.lastindex ... 94
Match.lastgroup ... 94
Atributy instancí třídy Pattern ... 94
Pattern.flags ... 94
Pattern.groups ... 94
Pattern.groupindex ... 94
Pattern.pattern ... 94
6.7 Zdrojové kódy ... 94
7 Bajtové objekty ... 95
7.1 Úvod ... 95
7.2 Třída bytes – zadávání bajtových stringů ... 96
Bajtové literály ... 96
Použití konstruktoru ... 97
Použití třídní tovární metody fromhex() ... 98
7.3 Třída bytearray – zadávání bajtových polí ... 98
7.4 Metody tříd bytes a bytearray ... 99
hex(sep='', bytes_per_sep=1) ... 99
7.5 Modifikační metody třídy bytearray ... 100
7.6 Zdrojové kódy ... 100
Část B Vstup a výstup dat 101
8 Základní informace o souborech ... 102
8.1 Posixové operační systémy ... 102
8.2 Soubory: bleskové opakování ... 102
Soubor, souborový systém, cesta ... 103
Ohlédnutí do historie za používáním zpětného lomítka ... 104
Absolutní a relativní cesta ... 105
Substituované disky ve Windows ... 105
8.3 Práce se soubory v *Pythonu* ... 106
Starší koncepce souborů v jazycích C nebo Pascal ... 106
Novější koncepce datových proudů ... 106
Koncepce Pythonu ... 107
Shrnutí používané terminologie ... 107
Soubor (anglicky file) ... 107
Složka ... 107
Cesta (anglicky path) ... 108
Datový proud nebo jenom proud ... 108
8.4 Dva způsoby práce s cestami ... 108
8.5 Pracovní složka ... 109
8.6 Zdrojové kódy ... 110
9 Modul pathlib a abstraktní cesty ... 111
9.1 Představení ... 111
9.2 Konstrukce instancí ... 112
Terminologické úvahy ... 113
9.3 Vlastnosti a metody abstraktních cest ... 113
Prezentace a reprezentace ... 114
Vlastnosti abstraktních cest ... 115
Operace s abstraktními cestami ... 116
Operátory < <= == != >= > ... 117
Operátor slučování / ... 117

Konverzní metody ... 118
PurePath.as_posix() -> str ... 118
PurePath.as_uri() -> str ... 118
Zjišťovací metody ... 118
PurePath.is_absolute() -> bool ... 119
PurePath.is_relative_to(*other:PLO) -> bool ... 119
PurePath.is_reserved() -> bool ... 119
PurePath.match(pattern:str) -> bool ... 119
Sestavovací metody ... 119
PurePath.joinpath(*other : PLO) -> PurePath ... 119
PurePath.relative_to(*other:PLO) -> PurePath ... 120
PurePath.with_name(name:str) -> PurePath ... 120
PurePath.with_stem(stem:str) -> PurePath ... 121
PurePath.with_suffix(suffix:str) -> PurePath ... 121
9.4 Konkrétní cesty ... 121
9.5 Zdrojové kódy ... 121
10 Konkrétní cesty a práce se složkami a soubory ... 122
10.1 Úvod ... 122
Kompatibilní potomek a PLO ... 122
10.2 Aktuální složky ... 123
@classmethod Path.cwd()-> Path ... 123
@classmethod Path.home() -> Path ... 123
10.3 Informace o cestě a souboru ... 123
Path.absolute() -> Path ... 123
Path.exists() -> bool ... 123
Path.resolve(strict=False) -> Path ... 124
Path.samefile(other_path:PLO) -> ... 124
Path.is_dir() -> bool ... 124
Path.is_file() -> bool ... 124
10.4 Manipulace se soubory a složkami ... 125
Path.mkdir(mode=0o777, parents=False, exist_ok=False) -> None ... 125
Path.touch(mode=0o666, exist_ok=True) -> None ... 125
Path.rename(target: PLO) -> Path ... 125
Path.replace(target:PLO) -> Path ... 125
Path.rmdir() -> None ... 126
os.removedirs(target:PLO) -> None ... 126
Path.unlink(missing_ok=False) -> None ... 127
10.5 Procházení obsahu složky ... 127
Path.iterdir() -> iterable[Path] ... 127
Path.glob(pattern) -> generator[Path] ... 127
Path.rglob(pattern:str) -> list[Path] ... 127
10.6 Primitivní zápis a čtení dat ... 128
Path.read_bytes() -> bytes ... 128
Path.read_text(encoding=None, errors=None) -> str ... 128
Path.write_bytes(data: bytes-like object) -> int ... 128
Path.write_text(data:str, encoding=None, errors=None, newline=None) -> int ... 128
Vlastní zápis a čtení dat ... 128
10.7 Zdrojové kódy ... 129
11 Datové proudy ... 130
11.1 Otevření datového proudu ... 130
Path.open(mode:str='rt', buffering:int=-1, encoding:str=None, errors:str=None, newline:str=None) -> io.IOBase ... 130

open(path:PLO|int, mode:str='rt', buffering:int=-1, encoding:str=None, errors:str=None, newline:str=None, closefd:bool=True, opener=None) -> io.IOBase ... 130
Textová versus binární data ... 133
11.2 Problematika kódování ve Windows ... 134
Nastavení systémové proměnné PYTHONUTF8 ... 134
Znaková sada Unicode a kódování UTF-8 ... 135
Kódování znaků sady Unicode ... 135
11.3 Architektura proudů ... 137
Abstraktnost rodičovských tříd ... 138
Dělení datových proudů ... 138
Souborové proudy ... 139
Paměťové proudy ... 139
Metody používané v dalším výkladu ... 139
11.4 Souborové proudy ... 140
Textové souborové proudy ... 140
Binární souborové proudy ... 142
11.5 Paměťové proudy ... 143
Třída StringIO ... 143
StringIO(initial_value='', newline='\n') ... 143
getvalue() -> str ... 144
Třída BytesIO ... 144
BytesIO(initial_bytes=b'') ... 144
getbuffer() ... 144
getvalue() -> bytes ... 144
11.6 Nezmíněné proudy ... 144
11.7 Zdrojové kódy ... 145
12 Čtení a zápis dat ... 146
12.1 Úvod ... 146
12.2 Zavírání proudu ... 146
close() -> None ... 147
closed:bool ... 147
12.3 Čtení uložených dat ... 147
readable() -> bool ... 147
read(size=- 1, /) -> str|bytes ... 147
readline(size=- 1, /) -> str|bytes ... 147
readlines(size=- 1, /) -> list[str|bytes] ... 147
readinto(b, /) -> int ... 148
peek(size=0, /) -> bool ... 148
Příklad ... 148
12.4 Zápis dat ... 149
writable() -> bool ... 150
write(obj:str|byte_object, /) -> int ... 150
writelines(lines:list[str|byte_object], /) -> None ... 150
12.5 Vyrovnávací paměť a splachování ... 151
Technické pozadí ... 151
Vyrovnávací paměť v proudech ... 151
flush(self, *args, **kwargs) -> None ... 152
12.6 Konstrukce with a správce kontextu ... 153
12.7 Datový kurzor ... 154
tell() -> int ... 154
seekable() -> bool ... 154
seek(offset, whence=SEEK_SET, /) -> int ... 155
truncate(size=None, /) -> int ... 155

Demonstrační programy **155**
12.8 Zdrojové kódy **158**

Část C Základy práce s daty **159**

13 Datové třídy **160**
13.1 Představení **160**
13.2 Definice **161**
`dataclass(*, init=True, repr=True, eq=True, order=False, unsafe_hash=False, frozen=False, match_args=True, kw_only=False, slots=False, weakref_slot=False)` 161
Omezení implicitních počátečních hodnot **163**
Pořadí 163
Proměnnost 163
Užitečné datové atributy **164**
`KW_ONLY` 164
`MISSING` 164
13.3 Funkce `field()` **165**
`field(*, default=MISSING, default_factory=MISSING, init=True, repr=True, hash=None, compare=True, metadata=None, kw_only=MISSING)` 165
Příklady použití funkce `field()` s argumentem `default_factory` 166
Lambda-výrazy **168**
13.4 Vylepšujeme vytváření instancí **169**
Třídní datové atributy **169**
Vliv třídního atributu na implicitní hodnotu 169
Zdůraznění třídnosti atributu v anotaci 170
Pole v hierarchii dědění **171**
Přerovnávání zděděných polí při povinném pojmenování **172**
Metoda `__post_init__()` **173**
Doplnění volání initoru předka 173
Inicializační proměnné **174**
13.5 Další užitečné funkce **175**
`asdict(obj, *, dict_factory=dict)` 175
`astuple(obj, *, tuple_factory=tuple)` 175
`replace(obj, /, **changes)` 175
13.6 Zdrojové kódy **176**

14 Doprovodný program **177**
14.1 Představení doprovodné hry **177**
Idea hry **178**
Scénáře **178**
14.2 Trocha terminologie **182**
Databáze 182
Systém řízení báze dat – SŘBD (Database management system – DBMS) 182
Relační databáze (Relational database) 182
Záznam (record) 182
Položka (field) 182
Sloupec (column) 182
14.3 Architektura databáze hry **183**
Organizace tabulek a jejich polí **183**
`Games` – tabulka her **183**
`Places` – tabulka prostorů **183**
`Items` – tabulka h-objektů (objektů hry) **184**
`Neighbors` – tabulka průchodů mezi prostory **184**
`Actions` – tabulka akcí hry **185**

TypeOfStep – akce hry ... 185
Zobrazení architektury ... 185
14.4 Příprava tabulek ... 186
14.5 Zdrojové kódy ... 186
15 Dokumenty ve formátu JSON ... 187
15.1 Formát JSON ... 187
Syntaxe ... 188
15.2 Převod objektu do formátu JSON ... 188
dump(obj:object, fp:IOBase, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw) -> None ... 188
dumps(obj, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw) -> str ... 188
Přímo převeditelné typy Pythonu ... 190
15.3 Vlastní kodér ... 191
Alternativní přístup ... 193
Porovnání přístupů ... 195
15.4 Načítání dat ... 196
load(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw) -> object ... 196
loads(s, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw) -> object ... 196
Přímo převeditelné typy ... 197
Problémy s hodnotou NaN ... 197
Problémy se zpětným převodem polí ... 198
15.5 Vlastní dekodér ... 198
15.6 Druhá etapa zpětného převodu ... 200
15.7 Zpracování rozsáhlých dat ... 202
15.8 Zdrojové kódy ... 202
16 Dokumenty ve formátu CSV ... 203
16.1 Formát CSV ... 203
Syntaxe ... 203
16.2 Třída Dialect a její potomci ... 204
Zabudované dialekty ... 205
Registrace dialektů ... 205
list_dialects() -> list[str] ... 205
register_dialect(name:str, dialect:Dialect=None, **fmtparams=None) -> None ... 205
unregister_dialect(name) -> None ... 205
get_dialect(name:str) -> Dialect ... 206
Příklad ... 206
16.3 Externí zdroje CSV souborů ... 207
Excel ... 207
Calc ... 208
Problémy s desetinnou čárkou ... 208
16.4 Čtení CSV souborů ... 209
reader(csvfile, dialect:str|Dialect='excel', **fmtparams) ... 209
Příklad ... 210
DictReader(csvfile, fieldnames=None, restkey=None, restval=None, dialect='excel', *args, **fmtparams) ... 211
Další atributy ... 212

Příklad ... 212
16.5 Ukládání CSV souborů ... 213
writer(csvfile, dialect='excel', **fmtparams) ... 213
DictWriter(csvfile, fieldnames, restval='', extrasaction='raise', dialect='excel', *args, **kwds) ... 214
Další atributy ... 214
writerow(record) ... 214
writerows(records) ... 214
dialect ... 214
writeheader() ... 214
Příklad ... 215
16.6 Složitější příklad ... 216
16.7 Zdrojové kódy ... 219
17 Další užitečné moduly ... 220
17.1 Práce s náhodou ... 220
Náhodná versus pseudonáhodná čísla ... 220
Modul random ... 221
Základní funkce ... 221
seed(a=None) -> None ... 221
getstate() -> state ... 221
setstate(state) -> None ... 221
Random([seed]) ... 221
Generování celých čísel ... 222
randrange(stop:int) -> int ... 222
randrange(start:int, stop:int [, step:int]) -> int ... 222
randint(a:int, b:int) -> int ... 222
Operace s posloupnostmi ... 222
choice(seq) ... 222
choices(seq, weights=None, *, cum_weights=None, k=1) ... 222
sample(seq, k, *, counts=None) ... 222
shuffle(seq) ... 223
Příklad ... 223
Generování reálných čísel ... 224
random() ... 224
uniform(a, b) ... 224
gauss(mean=0.0, sigma=1.0) ... 224
17.2 Čísla typu Decimal ... 224
Kontext ... 225
Konstruktor ... 226
Decimal(value='0', context=None) ... 226
Operace ... 226
17.3 Datum a čas ... 226
Trocha terminologie ... 227
Uvědomělé a naivní objekty ... 227
Epocha ... 228
Časové razítko ... 228
Třída time.struct_time ... 228
Modul time ... 229
time() -> float ... 229
ctime([secs:float]) -> str ... 229
gmtime([secs:float]) -> struct_time ... 229
localtime([secs:float]) -> struct_time ... 229
mktime(t: struct_time|tuple[int]) -> float ... 229
asctime([t:struct_time]) -> str ... 229

monotonic() -> float ... 230
process_time() -> float ... 230
perf_counter() -> float ... 230
sleep(secs:float) -> None ... 230
Rozšíření ve verzi 3.7 ... 230
Příklad ... 230
Modul datetime ... **231**
Třída datetime.date ... **232**
Konstruktory a tovární metody ... 232
date(year:int, month:int, day:int) -> date ... 232
date.today() -> date ... 232
date.fromtimestamp(timestamp:int) -> date ... 232
date.fromordinal(ordinal:int) -> date ... 232
date.fromisoformat(date_string:str) -> date ... 232
date.fromisocalendar(year:int, week:int, day:int) -> date ... 232
Další metody ... 233
replace(year:int=self.year, month:int=self.month, day:int=self.day) -> date ... 233
strftime(format:str) -> str ... 233
timetuple() -> struct_time ... 233
Třída datetime.time ... **233**
Konstruktory a tovární metody ... 233
time(hour=0, minute=0, second=0, microsecond=0, tzinfo:tzinfo=None, *, fold=0) ... 233
fromisoformat(time_string) ... 233
Další metody ... 233
replace(hour=self.hour, minute=self.minute, second=self.second, microsecond=self.microsecond, tzinfo=self.tzinfo, *, fold=0) ... 233
isoformat(timespec='auto') ... 234
strftime(format:str) -> str ... 234
utcoffset() ... 234
tzname() ... 234
Třída datetime.datetime ... **234**
datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0) ... 234
Přidané metody ... 234
datetime.now(tz=None) -> datetime ... 234
datetime.combine(date, time, tzinfo=self.tzinfo) -> datetime ... 235
Třída datetime.timedelta ... **235**
timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0) ... 235
17.4 Zdrojové kódy ... **236**

Část D Důležité externí knihovny 237

18 Virtuální prostředí a externí knihovny ... 238

18.1 Balíčkový systém Pythonu ... **238**
Druhy distribucí ... **238**
Zdrojové distribuce ... 239
Binární distribuce ... 239
Problémy a jejich řešení ... **239**
18.2 Program/modul pip ... **240**
Ověření instalace modulu pip ... **241**
Aktualizace ... **241**

18.3 Vzájemná nekompatibilita aplikací ... 242
Virtuální prostředí ... 243
Spuštění virtuálního prostředí ... 244
18.4 Instalace nových modulů ... 246
Ovlivnění verze ... 247
Aktualizace instalovaného modulu ... 248
18.5 Zdrojové kódy ... 249
19 Knihovna NumPy ... 250
19.1 Představení ... 250
Instalace ... 251
19.2 Pole typu ndarray ... 251
AL-objekty ... 252
19.3 Vytvoření nd-pole funkcí array ... 252
`array(object:AL, dtype=None, *, ndmin=0) -> ndarray` ... 252
19.4 Datové atributy nd-polí ... 254
19.5 Další způsoby vytvoření nd-pole ... 255
`empty(shape, dtype=None) -> ndarray` ... 255
`zeros(shape, dtype=None) -> ndarray` ... 255
`ones(shape, dtype=None) -> ndarray` ... 256
`identity(n, dtype=None) -> ndarray` ... 256
`arange(`*`[`*`start,`*`]`* `stop`*`[`*`, step,`*`]`* `dtype=None) -> ndarray` ... 256
`linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None) -> ndarray` ... 257
`geomspace(start, stop, num=50, endpoint=True, dtype=None) -> ndarray` ... 257
`logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None) -> ndarray` ... 257
Pole náhodných čísel ... 257
`np.random.default_rng(seed:int = None)` ... 257
Generovaná čísla ... 257
`integers(low, high=None, size=None, dtype=np.int64, endpoint=False)` ... 258
`random(size=None, dtype=np.float64, out=None)` ... 258
`normal(loc=0.0, scale=1.0, size=None)` ... 258
19.6 Úpravy vytvořených polí ... 258
Vytvoření pole se stejnými daty, ale jiným tvarem ... 259
`numpy.reshape(a:AL, newshape) -> ndarray` ... 259
`ndarray.reshape(newshape) -> ndarray` ... 259
`numpy.transpose(a:AL, axes=None) -> ndarray` ... 259
`ndarray.transpose(*axes) -> ndarray` ... 259
`ndarray.T` ... 259
Slučování polí ... 260
`hstack(arrays:tuple) -> ndarray` ... 261
`vstack(arrays:tuple) -> ndarray` ... 261
`dstack(arrays:tuple , axis=0) -> ndarray` ... 261
`column_stack(arrays:tuple) -> ndarray` ... 261
Rozdělení polí na několik menších ... 262
`hsplit(array:AL, parts:int|AL) -> tuple[ndarray]` ... 262
`vsplit(array:AL, parts:int|AL) -> tuple[ndarray]` ... 263
`dsplit(array:AL, parts:int|AL) -> tuple[ndarray]` ... 263
Kopírování – funkce `copy()` ... 263
Pohledy ... 263
Tvorba pohledů ... 264
Tvorba pohledů indexováním a vykrajováním ... 264
Jak rozeznat, zda tvorba pohledu vyžadovala vytvoření kopie ... 266

Zvětšení hodnosti (dimenze) pohledu na pole ... 267
19.7 Operace s nd-poli ... 268
Základní operace ... 268
Pole logických hodnot ... 269
Násobení polí a operátor @ ... 269
Složené operátory ... 271
19.8 Tisk polí ... 272
19.9 Co ve výkladu chybí ... 273
19.10 Zdrojové kódy ... 274
20 Knihovna Matplotlib ... 275
20.1 Představení ... 275
20.2 Instalace ... 275
20.3 Terminologie ... 276
Význam objektů typu Figure a Axes ... 278
Ovládací prvky okna obrázku ... 280
20.4 Dva přístupy k tvorbě ... 281
Objektově orientovaný přístup ... 282
Procedurální přístup ... 283
20.5 Formátování vytvářeného obrázku ... 284
Nastavení implicitní konfigurace ... 284
Použité metody ... 285
plot([x:AL], y:AL, [fmt:str], /, *, **kwargs) ... 285
plot([x:AL], y:AL, [fmt:str], [x2:AL], y2:AL, [fmt2:str], /, ..., **kwargs) ... 285
set_title(label:str, fontdict:dict=None, loc:str=None, pad:float=None, *, y:float=None, **kwargs) ... 286
set_xlabel(label:str, fontdict=None, labelpad:float=None, *,loc:str=None, **kwargs) ... 286
set_ylabel(label, fontdict=None, labelpad=None, *,loc=None, **kwargs) ... 286
Styl čáry ... 287
set_ls (ls : str|tuple[int]) ... 287
set_linestyle(ls : str|tuple[int]) ... 287
Barvy ... 288
set_c (color : str|tuple[int]) ... 288
set_color(color : str|tuple[int]) ... 288
Šířka čar ... 289
set_lw (w : float) ... 289
set_linewidth(w : float) ... 289
Zadání formátu v argumentech funkce plot() ... 289
Podoba značek ... 290
set_marker(m : str) ... 290
Lineární a logaritmická osa ... 291
20.6 Jiné druhy grafů ... 292
Sloupcové grafy ... 292
bar(x:AL, height:AL, width=0.8, bottom=None, *, align='center', yerr=None, **kwargs) ... 292
barh(y:AL, width:AL, height=0.8, left=None, *, align='center', xerr=None, **kwargs) ... 292
Histogram ... 294
hist(x, bins=None, range=None, density=False, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', log=False, color=None, label=None, stacked=False, **kwargs) ... 294
Koláčový graf ... 296

pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, *, normalize=True) 296
20.7 Zdrojové kódy 298
21 Knihovna pandas 299
21.1 Předmluva 299
21.2 Datové rámce a série 300
Datové rámce – DataFrame 300
Série – Series 301
Společný předek NDFrame a účel typy NDFrameT 302
Jednoduché operace 302
Terminologický zmatek 303
21.3 Čtení tabulkových dat 303
Načtení dat z CSV souboru 303
read_csv(source: PLO|io.IOBase, sep=',', delimiter=None, header='infer', names=None, index_col=None, usecols=None, dtype=None, converters=None, skipinitialspace=False, skiprows=None, nrows=None, skip_blank_lines=True, thousands=None, decimal='.', quotechar='"', comment=None, encoding=None, dialect=None) -> DataFrame 304
Atributy head(), tail(), dtypes, describe() 305
head(self: NDFrameT, n: int = 5) -> NDFrameT 306
tail(self: NDFrameT, n: int = 5) -> NDFrameT 306
dtypes(self: NDFrameT) -> dtype|Series[dtype] 306
describe(self: NDFrameT, percentiles=None, include=None, exclude=None) 306
Načtení dat z excelovské tabulky 307
read_excel(source: PLO|io.IOBase|ExcelFile, sheet_name=0, header=0, names=None, index_col=None, usecols=None, squeeze=None, dtype=None, converters=None, skiprows=None, nrows=None, thousands=None, decimal='.', skipfooter=0) -> DataFrame | dict[str,DataFrame] 307
21.4 Zápis tabulkových dat do souborů 310
to_csv(dest=None, sep=',', float_format=None, columns=None, header=True, index=True, index_label=None, decimal='.') -> None | str 310
to_excel(dest, sheet_name='Sheet1', float_format=None, columns=None, header=True, index=True, index_label=None, startrow=0, startcol=0, freeze_panes=None) 311
21.5 Jak vybrat podmnožinu dat z rámce 311
Výběr zadaných sérií – sloupců 311
Filtr záznamů – řádků 312
Současný výběr zadaných sloupců a řádků 314
loc[] 314
iloc[] 314
21.6 Přidání a odstranění série – sloupce 315
21.7 Přejmenování sloupců (sérií) a/nebo řádků (záznamů) 316
rename(*, index=None, columns=None, copy= True) 316
21.8 Nastavování různých parametrů 317
Organizace voleb 317
Funkce pro práci s volbami 318
get_option(pat:str) -> object 318
set_option(pat:str, value:object) -> None 318
reset_option(pat:str) -> None 318

describe_option(pat:str='', _print_desc=False) -> None | str 318
option_context(*args) -> <context manager> 318
Příklad 319
Nastavování voleb jako zdánlivých atributů objektu options 319
21.9 Analýza zpracovávaných dat 320
Dostupné funkce 320
Omezení dalších příkladů 321
Rozdělení funkcí 321
Použití agregačních funkcí 321
Sdružování statistik podle kategorií 322
21.10 Zdrojové kódy 322

Literatura 323

Rejstřík 325

Poděkování

Vím, že se v českých knížkách většinou neděkuje, ale tvorba knih v současném tempu je spojena s takovými oběťmi řady lidí z mého blízkého i vzdálenějšího okolí, že bych měl velkou újmu na duši, kdybych tak neučinil.

Chtěl bych především nesmírně poděkovat své ženě Jarušce, která byla po celou dobu mojí největší oporou a jejíž nekonečná trpělivost a vstřícnost mi pomohla dokončit knihu v termínu, který se příliš nelišil od toho, jejž jsme původně s nakladatelem dohodli, a ne až někdy za rok po něm. Stále marně přemýšlím, kde má schovanou tu svatozář.

Na vylepšování textu nového vydání se podílela řada dalších lidí. Musím poděkovat především těm, kteří si dali práci s odhalováním případných chyb ve vznikajícím rukopisu. Mezi nimi pak především Luďkovi Šťastnému, který po celou přípravu rukopis pročítal a odhaloval v něm pasáže, které by si zasloužily vylepšit. Neméně velkou zásluhu na současné podobě má i Jirka Kofránek, který mne průběžně upozorňoval na některé problémy s výukou podle běžně používaných postupů. Řadou podnětných myšlenek přispěl Michal Palas, jenž pracuje v oblasti analýzy a zpracování dat a přispěl tak řadou poznatků z praxe.

Nezanedbatelnou zásluhu má Adam Záhorský, který v rámci své technologické praxe nastudoval pravidla práce s knihovnou *tkinter* a vybavil můj testový tester regulárních výrazů grafickým uživatelským rozhraním. Poté ještě připravil převodníky scénářů mých studentů do formátů JSON a CSV a odevzdané práce převedl.

Svoji zásluhu má i Martina Jandová, která mne osvobozovala od nejrůznějších administrativních povinností a umožnila mi soustředit se pouze na výuku a psaní knih.

Velký dík patří i redaktoru Petrovi Somogyimu, který musel opakovaně procházet některé již zredigované pasáže, protože jsem je znovu a znovu upravoval. Nemalý dík patří i šéfredaktoru Radku Matulíkovi, který mne k napsání jednotlivých knih z posledních let vyhecoval, a byl pak ochoten týden či dva počkat, když se mi nepodařilo přesně dodržet původně dohodnutý termín.

Úvod

Python je moderní programovací jazyk, který umožňuje velmi jednoduše navrhovat jednoduché programy, ale na druhou stranu nabízí dostatečně mocné prostředky k tomu, abyste mohli s přiměřeným úsilím navrhovat i programy poměrně rozsáhlé. Je pro něj vyvinuto obrovské množství knihoven a frameworků, které uživatelům umožňují soustředit se na řešený úkol a nerozptylovat se vývojem nejrůznějších pomocných podprogramů.

Jednou z oblastí, v nichž má *Python* v současné době dominantní postavení, je zpracování dat. Tato kniha vychází vstříc všem, kteří potřebují pracovat s daty nejrůznějšího druhu. Představuje základní moduly standardní knihovny, jež se této oblasti týkají, a seznamuje se třemi nejpoužívanějšími externími knihovnami: knihovnou *NumPy* nabízející rozsáhlý matematický aparát pro práci s poli, knihovnou *Matplotlib* definující funkce, které velmi usnadňují tvorbu nejrůznějších grafů, a knihovnou *pandas* nabízející bohatou sadu nástrojů pro práci s daty uspořádanými do tabulek.

Komu je kniha určena

Tato kniha je určena všem, kteří znají jazyk *Python* přibližně na úrovni prvních tří částí učebnice [19] nebo prvních čtyř částí příručky [17], a chtěli by se seznámit se základními možnostmi práce s daty poskytovanými jeho knihovnami.

Kniha není učebnicí programování (tou je např. zmíněná učebnice [19]), ani učebnicí jazyka *Python* (tou je např. zmíněná příručka [17]). Proto se nesnaží probrat jeho konstrukce, ale omezuje se při výkladu pouze na základní možnosti využití jeho knihoven. Narazí-li však při výkladu na konstrukci, která nebyla vysvětlena v učebnici [19], tak ji dovysvětlí.

Bohužel se musí omezit opravdu pouze na základní a široce používanou podmnožinu dostupných funkcí a metod, protože při podrobnějším výkladu by kniha dalece přesáhla únosnou velikost.

Koncepce výkladu a jeho uspořádání

Knihu jsem se snažil koncipovat tak, aby mohla sloužit jako učebnice i jako referenční příručka. Pokusil jsem se do ní zahrnout nejčastěji zpracovávané oblasti.

Kniha je rozdělena do čtyř částí.

První část: Práce se stringy

První část rozšiřuje znalosti o práci se stringy a poté se věnuje především regulárním výrazům, které jsou mocným nástrojem usnadňujícím mnohé aspekty práce s texty. Na závěr představuje bajtové objekty používané při práci s binárními daty.

Druhá část: Vstup a výstup dat

Druhá část se zabývá nástroji pro práci s daty v nejrůznějších typech souborů. Vysvětlí principy práce se soubory, ukáže jak pracovat s datovými proudy a jak s jejich pomocí číst a zapisovat data.

Třetí část: Základy práce s daty

Třetí část seznámí s datovými třídami a pak probere možnosti ukládání dat do souborů typu CSV a JSON a čtení dat uložených v těchto souborech. Na závěr se pak bude věnovat několika užitečným modulům. Nejprve představí možnosti pro generování pseudonáhodných čísel, poté probere typ `Decimal` umožňující práci s velkými čísly, u nichž je důležité zachování přesnosti, a na závěr se věnuje modulům definujícím datové typy pro práci s datem a časem.

Čtvrtá část: Důležité externí knihovny

Čtvrtá část se nejprve věnuje balíčkovacímu systému *Pythonu* a programu *pip* sloužícímu k instalaci externích knihoven. Seznámí s koncepcí virtuálního prostředí umožňujícího paralelně pracovat na projektech s různými potřebnými konfiguracemi používaných nástrojů. V závěrečných třech kapitolách postupně probere výše zmíněné tři knihovny: *NumPy*, *Matplotlib* a *pandas*.

Jazyk identifikátorů

Doprovodné programy bychom mohli rozdělit do dvou skupin:

- První skupinu tvoří tzv. AHA-příklady, tj. příklady, jejichž jediným cílem je, aby si čtenář řekl: „Aha, takto to funguje.“ V těchto příkladech jsou pro maximální názornost používané české identifikátory.
- Druhou skupinu tvoří programy, které se snaží řešit nějaký praktický problém. V těchto příkladech jsem podle všeobecně dodržovaných konvencí dal přednost identifikátorům anglickým. Komentáře a tištěné texty však zůstávají nadále české.

Potřebné vybavení

Pro úspěšné studium celé knihy je vhodné mít instalovanou platformu *Python* ve verzi nejméně 3.11. Tu lze stáhnout na adrese https://www.python.org/downloads/. Instrukce pro stažení všech dalších potřebných programů budete dostávat v okamžiku, kdy tyto programy začnou být potřeba.

Doprovodné programy

Text knihy je prostoupen řadou doprovodných programů. Budete-li si je chtít spustit a ověřit jejich funkci, potřebujete je nejprve stáhnout. Najdete je na stránce knihy na adrese[1] http://knihy.pecinovsky.cz/66_py311data.

Zdrojové soubory s doprovodnými programy (moduly) se nacházejí ve složce `66_INP`. Jejich názvy začínají vždy písmenem `m` následovaným dvojmístným číslem kapitoly. Následují-li dvě podtržítka a text (např. `m01__prolog`), jedná se o soubor příkazů použitých ve výpisech v průběhu kapitoly. Je-li za číslem malé písmeno následované jedním podtržítkem (např. `m01a_script`), jedná se o samostatný modul.

Soubory příkazů použitých ve výpisech mají ještě partnera. Ten se nachází ve složce `88_IWD` a má stejný název, ale odlišnou příponou. Soubory s příponou `py` jsou zdrojové soubory *Pythonu*, soubory s příponou `pydoc` obsahují záznam konverzace se systémem zaznamenaný ve výpisech v knize i s uvedenými čísly řádků.

Tyto soubory vám mají usnadnit sledování rozboru některých programů, abyste při něm nemuseli neustále přecházet mezi rozborem a rozebíraným výpisem. Soubory `pydoc` si můžete vytisknout nebo otevřít v jiném okně, v textu knihy pak číst rozbor s odkazy na čísla řádků a vedle sledovat příslušný výpis na papíře či v paralelně otevřeném okně textového editoru.

Použité typografické konvence

K tomu, abyste se v textu lépe vyznali a také abyste si vykládanou látku lépe zapamatovali, používám několik prostředků pro odlišení a zvýraznění textu.

Termíny První výskyt nějakého termínu a další texty, které chci zvýraznit, vysazuji **tučně**.

Název Názvy firem a jejích produktů vysazuji *kurzivou*. Kurzivou vysazuji také názvy kapitol, podkapitol a oddílů, na něž v textu odkazuji.

Citace Texty, které si můžete přečíst na displeji, např. názvy polí v dialogových oknech či názvy příkazů v nabídkách, vysazuji **tučným bezpatkovým písmem**.

[1] Číslem 66 v adrese se nevzrušujte, jedná se pouze o moje interní označení pořadí vytvářené knihy, protože bych v nich jinak bloudil.

Odkaz Celá kniha je prošpikovaná křížovými odkazy na související pasáže. Není-li odkazovaný objekt (kapitola, obrázek, výpis programu, …) na stejné stránce nebo na některé ze sousedních stránek, je pro čtenáře tištěné verze doplněn o číslo stránky, na níž se nachází. Čtenářům elektronické verze stačí, když na něj klepnou. Použitý prohlížeč by je měl na odkazovaný objekt ihned přenést.

Adresa Názvy souborů a internetové adresy vysazuji obyčejným bezpatkovým písmem.

`Program` Identifikátory a další části programů zmíněné v běžném textu vysazuji `neproporcionálním písmem`, které je v elektronických verzích pro zvýraznění tmavě červené.

`zadání` V záznamech komunikace se systémem budou texty, které zadává uživatel, vysazeny **`tučně`** (v elektronických verzích pro zvýraznění tmavě modře).

`Komentář` Nejrůznější komentáře v ukázkových příkladech jsou pak *`vysazeny kurzivou a podbarveny světle zeleně`*.

`klíč` Klíčová slova jsou v programech pro zvýraznění vysazena tučně a tmavě červeně a pro zvýraznění v tištěných knihách jsou navíc **`podtržena`**.

Kromě výše zmíněných částí textu, které považuji za důležité zvýraznit nebo alespoň odlišit od okolního textu, najdete v knize ještě řadu doplňujících poznámek a vysvětlivek. Všechny budou v jednotném rámečku, jenž bude označen ikonou charakterizující druh informace, kterou vám chce poznámka či vysvětlivka předat.

Symbol jin-jang bude uvozovat poznámky, s nimiž se setkáte na počátku každé kapitoly. Zde vám vždy prozradím, co se v dané kapitole naučíte.

Píšící ruka označuje obyčejnou poznámku, která pouze doplňuje informace z hlavního proudu výkladu o nějakou zajímavost.

Ruka s hrozícím prstem upozorňuje na věci, které byste měli určitě vědět a na které byste si měli dát pozor, protože jejich zanedbání vás většinou dostane do problémů.

Usměváček vás bude upozorňovat na různé tipy, jimiž můžete vylepšit svůj program nebo zefektivnit svoji práci.

Mračoun vás naopak bude upozorňovat na různá úskalí programovacího jazyka nebo programů, s nimiž budeme pracovat, a bude vám radit, jak se těmto nástrahám vyhnout či jak to zařídit, aby vám alespoň pokud možno nevadily.

Obrázek knihy označuje poznámku týkající se používané terminologie. Tato poznámka většinou upozorňuje na další používané termíny označující stejnou skutečnost nebo na konvence, které se k probírané problematice vztahují. Seznam všech terminologických poznámek najdete v rejstříku pod heslem „terminologie“.

Brýle označují tzv. „poznámky pro šťouraly“, ve kterých se vás snažím seznámit s některými zajímavými vlastnostmi probírané konstrukce nebo upozorňuji na některé souvislosti, které však nejsou k pochopení látky nezbytné.

Odbočka – podšeděný blok

Občas je potřeba vysvětlit něco, co nezapadá přímo do okolního textu. V takových případech používám podšeděný blok se silnou čarou po straně. Tento podšeděný blok je takovou drobnou odbočkou od ostatního výkladu. Nadpis podšeděného bloku pak najdete i v podrobném obsahu mezi nečíslovanými nadpisy.

Zpětná vazba

Tato kniha je sice mou šedesátou šestou učebnicí, ale současně je to moje první kniha, v níž se zabývám daným tématem, tj. knihovnami pro práci s daty. Nelze proto vyloučit, že se v ní mohou objevit přehlédnutí, která nemá redaktor šanci zachytit a opravit.

Pokud vám proto bude někde připadat text nepříliš srozumitelný nebo budete mít nějaký dotaz (ať už k vykládané látce či použitému vývojovému prostředí), anebo v knize objevíte nějakou chybu či budete mít návrh na nějaké její vylepšení v případné příští verzi, neostýchejte se poslat na adresu rudolf@pecinovsky.cz e-mail s předmětem mailto:rudolf@pecinovsky.cz?subject=66_PYTHON_DATA_DOTAZ.

Bude-li se dotaz týkat něčeho obecnějšího nebo to bude upozornění na chybu, pokusím se co nejdříve zveřejnit na stránce knihy http://knihy.pecinovsky.cz/66_py311data odpověď i pro ostatní čtenáře, kteří by mohli o danou chybu zakopnout, nebo by je mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že si netroufnou se sami zeptat.

Předem se však omlouvám, že budu určitě odpovídat se značným zpožděním, protože při počtu knížek, které si Grada objednala, a paralelní výuce už nemám čas na včasné odpovídání na e-maily.

I když možná bude brzy všechno jinak, protože už nehodlám nadále sponzorovat naše školství (za situace, kdy u nás vysokoškolského učitele platí méně než prodavačku v supermarketu, nemohu svoji výuku považovat za něco jiného než sponzoring). Velká zátěž spojená s výukou na univerzitách tak možná brzy odpadne a budu se moci více věnovat svým knihám a jejich čtenářům.

Část A
Práce se stringy

První část knihy prohloubí vaše znalosti práce se stringy, které jsou jedním z nejpoužívanějších datových typů (a pro mnohé dokonce typem nejpoužívanějším). Seznámí vás nejprve s metodami, které třída `str` a její instance poskytují, a poté se plně soustředí na výklad regulárních výrazů, což je nástroj, jenž umožňuje výrazné zefektivnění práce se stringy a uplatní se nejenom při tvorbě programů, ale velice často jej využijeme i při ostatních počítačových aktivitách. Na závěr pak představí bajtové stringy jako způsob práce s binárními daty.

Kapitola 1
Metody třídy str

str

Co se v kapitole naučíte
Bez ohledu na to, jaký typ programu budete vytvářet, dříve nebo později budete potřebovat nějakým způsobem pracovat s textem uloženým ve strinzích. Tato kapitola prohloubí vaše znalosti metod, které stringy nabízejí. Nebude ale probírat otázku jejich formátování – ta je řešena v příručce [17].

1.1 Efektivnější spojování stringů

Pamatujte si, že stringy v Pythonu jsou neměnné (dokonce hešovatelné), což znamená, že když chcete sloučit dva stringy dohromady, vytvoří se nový string, do kterého se nakopírují ty slučované. To má ovšem značný vliv na výkon celé operace.

Spojujete-li jen dva stringy, není to problém, jestliže byste však měli spojovat pomocí operátoru `+` více stringů, anebo spojovat stringy v cyklu, může se kvůli neustálému vytváření nových stringů a kopírování těch předešlých celý výpočet výrazně zpomalit. V takovém případě je vhodnější použít metodu `join()`.

```
join(iterable)
```

Metoda `join()` očekává v argumentu zdroj stringů, které sloučí, přičemž mezi ně vloží vždy obsah vlastního objektu. Takto vytvořený string pak vrátí jako svoji návratovou hodnotu. Ukázku použití najdete ve výpisu 1.1.

Výpis 1.1: *Slučování stringů metodou `join()`*

```
>>> jhs = ["Jablka", "Hrušky", "Švestky"]
>>> ', '.join(jhs);   ''.join(jhs)
'Jablka, Hrušky, Švestky'
'JablkaHruškyŠvestky'
>>> "; ".join("Jablka")
'J; a; b; l; k; a'
>>>
```

Na řádku 1 je definován zdroj slučovaných stringů. Na řádku 2 jsou pak dva příkazy k jejich sloučení, přičemž v prvním je oddělovač definován jako string tvořený čárkou a mezerou, v druhém je oddělovačem prázdný string. Druhý příkaz je ukázkou postupu, kdy chcete stringy pouze sloučit a nechcete je ničím oddělovat.

Příkaz na řádku 5 upozorňuje na to, že parametr metody `join()` je zdroj stringů. Pokud byste do parametru použili jen jeden string, *Python* by ho považoval za generátor jednotlivých znaků a ty by pospojoval zadaným oddělovačem.

1.2 Rozdělování stringů na části

Stejně jako je třeba někdy stringy spojovat, občas se hodí string rozložit na jednotlivé části. Pojďme se postupně seznámit s některými metodami, které nám takovouto funkcionalitu nabízejí.

```
str.split (sep=None, maxsplit=-1) -> list[str...]
str.rsplit(sep=None, maxsplit=-1) -> list[str]
```

Metoda `split()` má dva parametry, přičemž oba mají definovánu implicitní hodnotu. Prvním parametrem je podstring oddělující jednotlivé části. Výsledkem volání je seznam stringů, který obsahuje části původního stringu oddělené zadaným oddělovačem. Ukázku použití najdete ve výpisu 1.2 na řádku 1.

Výpis 1.2: *Ukázky rozdělení stringu použitím metod `split()` a `rsplit()`*

```
1  >>> 'Jablka a Hrušky a Švestky'.split(' a ')        # Standardní použití
2  ['Jablka', 'Hrušky', 'Švestky']
3  >>> '1,,2'.split(',')   # Oddělovače se neslučují
4  ['1', '', '2']
5  >>> '1 2  3   4    5     6      7'.split()  # Bezparametrický odděluje sadou mezer
6  ['1', '2', '3', '4', '5', '6', '7']
7  >>> 'Jablka a Hrušky a Švestky'.split(' a ', 1)     # Maximální počet částí
8  ['Jablka', 'Hrušky a Švestky']
9  >>> 'Jablka a Hrušky a Švestky'.rsplit(' a ', 1)    # Začínáme zprava
10 ['Jablka a Hrušky', 'Švestky']
11 >>>
```

Narazí-li metoda na dva sousední oddělovače, nesloučí je v jeden, ale umístí na příslušné místo do seznamu prázdný string – viz reakce na příkaz na řádku 3.

Nezadáte-li první argument nebo bude-li mít hodnotu `None`, rozdělovací algoritmus se mění a za oddělovač se považuje jeden či více sousedících bílých znaků. Jinými slovy: v takovém případě se oddělovače slučují. Dokládají to i reakce na příkaz na řádku 5.

Hodnota parametru `maxsplit` určuje, maximálně kolikrát se daný oddělovač při rozdělování použije. Je-li v stringu nalezen vícekrát, další výskyty se ignorují – viz reakce na příkaz na řádku 7.

Pokud byste potřebovali začít s vyhledáváním oddělovačů od konce stringu, stačí nahradit volání metody `split()` voláním metody `rsplit()`. Jak ukazuje reakce na příkaz na řádku 9, drobnou úpravou předchozího příkladu snadno smícháte hrušky a jablka.

`partition (sep:str) -> tuple[str,str,str]`
`rpartition(sep:str) -> tuple[str,str,str]`

Pro rozdělování stringů *Python* nabízí ještě metodu `partition()`. Na rozdíl od metody `split()` však nevrací seznam oddělených částí, ale n-tici `(začátek, oddělovač, konec)`. Ukázku použití najdete ve výpisu 1.3.

Výpis 1.3: *Ukázky rozdělení stringu použitím metod `partition()` a `rpartition()`*

```
1 >>> 'Mléko, Máslo, Chleba'.partition(', ')      # Standardní použití
2 ('Mléko', ', ', 'Máslo, Chleba')
3 >>> 'Název_souboru.přípona'.partition('.')      # Práce se soubory
4 ('Název_souboru', '.', 'přípona')
5 >>> 'Mléko, Máslo, Chleba'.partition('; ')      # Nenalezený oddělovač
6 ('Mléko, Máslo, Chleba', '', '')
7 >>> 'Mléko, Máslo, Chleba'.rpartition(', ')     # Prohledávání zprava
8 ('Mléko, Máslo', ', ', 'Chleba')
9 >>>
```

Jak ukazuje reakce na příkaz na řádku 1, metoda vezme string a vyhledá v něm první výskyt oddělovače. Pak jako nultou položku vracené n-tice uloží vše před ním, další položkou bude vlastní oddělovač a poslední položkou vše ostatní, až do konce stringu.

Příklad na řádku 3 naznačuje, že se tato metoda může hodit např. při práci se soubory.

Příklad na řádku 5 demonstruje situace, kde zadaný oddělovač ve stringu není. Metoda pak vrátí jako počáteční prvek n-tice celý string a na dalších pozicích dva prázdné stringy.

Chcete-li, aby se string prohledával od konce, použijte metodu `rpartition()`. Demonstrační příkaz je na řádku 7.

Pro rafinovanější rozdělování stringů je třeba použít regulární výrazy, které budeme probírat v několika následujících kapitolách.

`str.splitlines(keepends=False) -> list[str]`

V řadě případů potřebujeme nějaký víceřádkový text rozdělit na jednotlivé řádky, které pak budeme zpracovávat. Stringy k tomu nabízejí použití metody `splitlines()`, která vrátí seznam stringů jednotlivých řádků.

Stringy ve vraceném seznamu již neobsahují znaky ukončení řádku, ledaže byste o to explicitně požádali zadáním argumentu s hodnotou `True`.

Tato metoda rozeznává jako konec řádku nejenom standardní konec řádku, ale i několik dalších znaků, jako např. konec stránky nebo vertikální tabulátor. Nebudu je tu vyjmenovávat, jejich seznam najdete v dokumentaci metody `splitlines()`.

Od metody `split()` se kromě toho, že akceptuje libovolný z celé sady oddělovačů (konců řádku), liší také tím, že končí-li zadaný string koncem řádku, neobsahuje vrácený seznam jako poslední prvek prázdný string reprezentující prostor za tímto oddělovačem.

Vše ilustruje výpis 1.4, v němž najdete sadu příkladů převzatých z dokumentace či dokumentací inspirovaných.

Výpis 1.4: *Demonstrace funkce metody `splitlines()`*

```
>>> s = 'ab c\n\nde fg\rkl\r\n'
>>> print(f'{s.splitlines()     = }\n{s.splitlines(True) = }')
s.splitlines()     = ['ab c', '', 'de fg', 'kl']
s.splitlines(True) = ['ab c\n', '\n', 'de fg\r', 'kl\r\n']
>>> LF='\n';   print(f"{''.splitlines()=}  -  {''.split(LF)}")
''.splitlines()=[]  -  ['']
>>> line="Jeden řádek\n";  print(f'{line.splitlines()=}  -  {line.split(LF)=}')
line.splitlines()=['Jeden řádek']  -  line.split(LF)=['Jeden řádek', '']
>>>
```

1.3 Jednoduché úpravy

Stringy je potřeba velmi často nějak drobně upravit. Nejčastější operací je oříznutí přebytečných znaků na kraji stringu, o něco málo složitější je pak nahrazení některých jejich částí jinými stringy.

Vzhledem k neměnnosti stringů snad nemusím připomínat, že **všechny uvedené modifikační metody vracejí nově vytvořený string** se zapracovanými modifikacemi a že původních oslovených stringů se požadované modifikace nijak nedotknou.

Ořezávání stringů

Častou úpravou stringů je odstranění některých znaků z jejich začátku nebo konce. Nejčastěji se takto ořezávají bílé znaky, ale nemusí to tak být vždy. Ukázky použití různých metod najdete ve výpisu 1.5.

Příkaz na řádku 1 demonstruje nejčastější použití, kdy se ořezávají počáteční a koncové bílé znaky – většinou mezery (v kódu je sice zapsáno *mezery*, ale platí to pro všechny bílé znaky). Používáme k tomu volání metody **`strip()`**[2] bez argumentů.

[2] Když jsem začal pracovat v *Pythonu*, pletly se mi názvy metod s názvy jejich ekvivalentů v knihovnách jiných jazyků. Název metody `strip()` jsem si zapamatoval tak, že jsem si řekl, že se jedná o „striptýz stringu“, z nějž „svlékneme“ jeho překážející okraje.