

Računalniški praktikum (fizika) - vaje

Rok Kuk

2022-01-06

Contents

O strani	5
1 Namestitev okolja za vaje	7
1.1 Namestitev Pythona	7
1.2 Namestitev Visual Studio Code	8
1.3 Numpy	8
1.4 Pogoste težave	8
2 Uvod v Python	9
3 Zanke	11
4 Seznami	13
5 Delo z objekti	15
5.1 Metode za nize	15
6 Numpy	17
7 Datoteke	19
7.1 Datotečni sistem	19
7.2 Pisanje	20
7.3 Branje	21
7.4 Mode	22

O strani

Na tej strani so zbrani zapiski za vaje predmeta računalniški praktikum v 1. letniku študija fizike na Fakulteti za matematiko in fiziko Univerze v Ljubljani.

Zapiski so mišljeni le kot opora pri izvajanju vaj in ne obsegajo čisto vseh obravnavanih vsebin. Zapiski zato ne morejo nadomestiti obiskovanja predavanj in vaj.

Ta spletna stran je dostopna na <https://python.rokuk.org> in <https://rokuk.github.io/rp-fiz-notes>

Markdown koda za strani je dostopna na <https://github.com/rokuk/rp-fiz-notes> po licenci GNU GPLv2.

Chapter 1

Namestitev okolja za vaje

Stran je v delu.

Da na računalniku uporabljate Python in rešujete naloge je potrebno namestiti nekaj programov. Spodaj je opisan okvirni postopek in pogoste težave pri nameščanju programov. Če imate težave, je opis problema dobro pogooglati, sicer pa pišite asistentu ali postavite vprašanje na forumu.

1.1 Namestitev Pythona

1. Namestite Python (najbolje kar verzijo 3.10) s te strani (zavihek Downloads): <https://www.python.org> Če uporabljate Windows 7 ali še starejši Windows, boste morali namestiti starejšo verzijo Pythona (npr. 3.8.9 ali manj). Najdete jo tule: <https://www.python.org/downloads/>

Ko poženete program za namestitev, v oknu, ki se odpre, odkljukajte “Add Python 3.x to PATH”. Nato nadaljujte z namestitvijo (opcija Install now).

2. Preverite ali se je Python uspešno namestil. Odprite program Ukazni poziv (Windows) ali Terminal (macOS in Linux), ki je že na vašem računalniku. V okno, ki se odpre vpišite ukaz `python --version` in pritisnite tipko Enter. Če je Python uspešno nameščen, bi se vam v novi vrstici moralo izpisati `Python 3.x.y` (kjer je `x.y` verzija nameščenega Pythona).

Če ste na Windowsu in ukaz `pyhon --version` ne izpiše verzije, temveč javi napako, poskusite ukaz `py --version`.

1.2 Namestitev Visual Studio Code

3. Namestitev Visual Studio Code: <https://code.visualstudio.com>
4. Namestitev Python extension - Odprite Visual Studio Code. Morda se vam bo V VSCode oknu pojavil zavihek z naslovom **Get Started**, ki ga lahko kar zaprete. Na levi kliknite na **Extensions** (ikona s štirimi kvadrati), vpišite **Python**, izberite **Python** in na desni kliknite **Install**. Morda se bo odprlo okno **Get Started**, ki ga lahko zaprete.
5. Dobro je, da si nekje na računalniku ustvarite mapo, v katero boste shranjevali vso vašo kodo. V VSCode v meniju kliknite **Open Folder...** in izberite to mapo. Morda se bo pojavilo okno, ki vas sprašuje, če zaupate avtorju datotek v tej mapi: kliknite **Yes**, ker sebi zaupate. Ustvarite novo datoteko (meni **File > New File**), jo shranite (meni **File > Save**) in jo poimenujte **test.py** (na Windowsu izberete **Save as type: Python**). Vsebina datoteke se vam odpre kot zavihek v VSCode. Vanj vpišite `print("Pozdravljen svet!")`. V desnem zgornjem kotu bi morali imeti gumb (**|>**), s katerim lahko poženete napisani program. Če ga nimate, si lahko namestite extension z imenom **Code Runner**. Sicer lahko program poženete tudi tako, da desno kliknete v območju urejevalnika in nato v meniju, ki se pojavi izberete **Run Python File in Terminal**.
6. Priporočam, da vklopite tudi "linter". To je program, ki je del VSCode in v vaši kodi sproti preverja ali ste se kje zmotili. Ne ujame vseh možnih napak, mnoge pa zazna in vas nanje opozori, še preden poženete program. VSCode pritisnite **Ctrl+Shift+P** in vpišite "linter", kliknite na **Python: Select Linter**. Pojavi se meni z različnimi možnostmi za različne linterje. Priporočam **flake8**, ki nas poleg napak opozori tudi na kršitve priporočil za stil PEP8. VSCode vas bo levo spodaj obvestil, da ta linter ni nameščen; namestite ga tako, da kliknete **Install** v tem obvestilu. Po nekaj sekundah bo nameščen.

VSCode ima veliko funkcionalnosti, ki nam lahko pomagajo pri programiranju. Več o tem v uradni dokumentaciji: <https://code.visualstudio.com/docs/editor/codebasics>

1.3 Numpy

1.4 Pogoste težave

Chapter 2

Uvod v Ptyhon

Stran je v delu.

Chapter 3

Zanke

Stran je v delu.

Chapter 4

Seznami

Stran je v delu.

Chapter 5

Delo z objekti

Gradivo za to poglavje je <https://automatetheboringstuff.com/2e/chapter6/>

Metod je veliko, spodaj je naštetih nekaj najpogostejše uporabljanih. Celoten seznam je v uradni dokumentaciji: <https://docs.python.org/3/library/stdtypes.html#string-methods>

Ponavadi lahko z Googlom, najdemo metodo, ki jo potrebujemo, če opišemo, kaj želimo narediti (npr. s `python count characters in string` hitro najdemo `count()` in primere uporabe).

5.1 Metode za nize

- `niz.count(znak)` vrne kolikokrat se znak pojavi v nizu
- `niz.index(znak)` vrne indeks, na katerem se znak prvič pojavi
- `niz.replace(prviniz, druginiz)` vrne niz, kjer so vsi nizi enaki `prviniz` zamenjani z `druginiz`
- `niz.lower()` in `niz.upper()` vrne niz, kjer iz malih črk naredi velike ali obratno
- `niz.islower()` in `niz.isupper()`
- `niz.strip()` vrne niz, kjer z leve in desne strani odstrani "whitespace characters" (presledki, tab, "\n"). Lahko podamo neobvezni argument, s katerim določimo, katere znake naj odstrani z leve in desne. Obstajata tudi metodi `rstrip()` in `lstrip()`, ki odstranjujeta le z leve in desne.

```
print('    Hello, World    \n'.strip())
```

```
## Hello, World
```

- `"locilo".join(seznam)` združi elemente seznama v niz in postavi `locilo` med posamezne elemente

```
print('ABC'.join(['Moje', 'ime', 'je', 'Rok']))
```

```
## MojeABCimeABCjeABCRok
```

- `niz.split(locilo)` vrne seznam, kjer so elementi posamezni deli niza, ki jih ločuje `locilo`. Privzeta vrednost za `locilo` je presledek.

```
print("Moje ime je Rok.".split())
```

```
## ['Moje', 'ime', 'je', 'Rok.']
```


Chapter 6

Numpy

Stran je v delu.

Chapter 7

Datoteke

Gradivo za to poglavje je <https://automatetheboringstuff.com/2e/chapter9/>

7.1 Datotečni sistem

Datoteke so shranjene na različnih nosilcih (npr. trdi disk, SSD, DVD, USB ključ, ...). Na računalniku datoteke organiziramo po mapah, ki so lahko gnezdene. Na vrhu imamo korensko mapo (root folder). Na Linux in macOS je to / na Windowsu pa C:\, kjer je C ime particije.

Prostor, ki je na voljo na nosilcu lahko razdelimo na več ločenih delov, ki jim rečemo particije. npr. trdi disk z 1000 GB bi lahko razdelili na dve particiji C z 100 GB in D z 900 GB). Vsaka particija na nosilcih, ki so priklopljeni na računalnik, dobi svojo črko (to velja za Windows, drugje je drugače). npr. USB ključi so pogosto pod E ali F.

7.1.1 Absolutna in relativna pot

Vsaki datoteki ustreza ena absolutna pot. To je “naslov”, pod katero jo lahko najdemo. Primer: C:\eggs\bacon\spam.txt. Pot vsebuje vse mape, v katerih se datoteka nahaja, ločene z \ (na Windowsu; na Linux in macOS je ločilo /), ime datoteke, piko in končnico datoteke, ki določa njen tip.

Relativna pot do datoteke je pot glede na neko drugo mapo. Za zgornji primer: glede na mapo eggs je relativna pot do datoteke .\bacon\spam.txt, kjer pika pomeni trenutno mapo. Če bi imeli mapo tomatoes v mapi C: in v njej datoteko dat.txt, bi bila relativna pot glede na eggs enaka ..\tomatoes\dat.txt. Dve piki pomenita eno mapo višje v hierarhiji (parent folder) glede na trenutno mapo. Če bi želeli iti dve mapi višje bi uporabili ../../\nekadrugamapa, itd.

Za podrobnejši razlago in več primerov glej gradivo: <https://automatetheboringstuff.com/2e/chapter9/>

7.1.2 Delo z ukaznim pozivom

Podobno kot v Raziskovalcu (File Explorer) se tudi v ukaznem pozivu (Terminal) v nekem trenutku nahajamo v neki mapi (ang. Current working directory ali CWD). Ta mapa je vedno napisana na začetku vrstice. V ukaznem pozivu najprej napišemo ukaz nato parametre, ki jih želimo podati, ločene s presledki. Ukaz izvedemo s tipko Enter.

V neko mapo se lahko premaknemo z ukazom `cd`, ki mu kot argument podamo pot do mape, v katero se želimo premakniti.

Ukaz `dir` izpiše vse datoteke in mape, ki se nahajajo v trenutni mapi.

Glej tudi: <https://ucilnica.fmf.uni-lj.si/mod/page/view.php?id=2505>

7.1.3 Mape in datoteke v Pythonu

Za delo z datotečnim sistemom je na voljo knjižnica `os`. Posamezne funkcije in njihovo uporabo lahko poiščete v uradni dokumentaciji. Nekaj najbolj uporabnih je `os.getcwd()`, `os.chdir()`, `os.listdir()`, `os.mkdir()`, `os.rename()`, `os.remove()`, `os.rmdir()`.

Za delo s potmi je uporabna knjižnica `os.path`, kjer so uporabne funkcije `os.path.join()`, `os.path.exists()`, `os.path.abspath()`, `os.path.relpath()`.

7.2 Pisanje

Datoteko odpremo v načinu za pisanje `mode="w"` in uporabimo funkcijo `write()`, ki zapiše niz v datoteko. Znak `\n` pomeni novo vrstico. Če želimo zapisati znak `\` moramo v Pythonu napisati `\\`. Več o uporabi `\` v Pythonu: https://www.w3schools.com/python/gloss_python_escape_characters.asp

```
potdodatoteke = "datoteka.txt"
with open(potdodatoteke, mode="w", encoding="utf-8") as dat:
    dat.write("To je ")
    dat.write("en stavek.\nTo je drugi.")
```

```
## datoteka.txt
## To je en stavek.
## To je drugi.
```

Namesto `dat.write("niz")` se lahko uporablja tudi `print("niz", file=dat)`, kjer odprto datoteko podamo kot parameter.

7.3 Branje

7.3.1 read()

Datoteko odpremo v načinu za branje `mode="r"` in uporabimo metodo `read()`, ki vrne celotno vsebino datoteke naenkrat v obliki niza.

```
with open("datoteka.txt", mode="r", encoding="utf-8") as datoteka:
    vsebina = datoteka.read()
print(vsebina)
```

```
## To je en stavek.
## To je drugi.
```

Uporaba argumenta `mode` je opisana na dnu strani. Klicu `open` lahko podamo tudi neobvezni argument `encoding`, ki poda kodno tabelo, v kateri je napisana datoteka. Privzeta vrednost tega argumenta je na Windowsu `cp1250`, kar je nekoliko zastarel standard, zato je dobra praksa uporaba parametra `encoding="utf-8"`, s čimer uporabimo Unicode, ki se danes uporablja skoraj povsod. Na macOS in Linux je vrednost `utf-8` že privzeta.

7.3.2 readlines()

Z metodo `readlines()` dobimo seznam, v katerem so posamezne vrstice iz datoteke.

```
with open("datoteka.txt", mode="r", encoding="utf-8") as datoteka:
    vrstice = datoteka.readlines()
print(vrstice)
```

```
## ['To je en stavek.\n', 'To je drugi.']
```

7.3.3 zanka

Po vrsticah datoteke lahko gremo z zanko `for`.

```
vrstice = []
with open("datoteka.txt", mode="r", encoding="utf-8") as datoteka:
    for line in datoteka:
        vrstice.append(line)
print(vrstice)
```

```
## ['To je en stavek.\n', 'To je drugi.']
```

7.4 Mode

je neobvezni argument funkcije `open()`. Privzeta vrednost je `mode="rt"`. Zato nam v zgornjih primerih ni bilo treba pisati `t` (je že privzet poleg druge črke, ki jo podamo (`r` ali `w`)). S posameznimi črkami povemo, kaj želimo z datoteko početi.

oznaka	opis	opomba
r	branje	če ne obstaja, javi napako
w	pisanje	če ne obstaja, ustvari novo, izbriše prejšnjo vsebino datoteke
a	append	če ne obstaja, ustvari novo, ne izbriše prejšnje vsebine
x	ustvari datoteko, pisanje	če že obstaja, javi napako
+	pisanje in branje	
t	za delo s tekstovnimi datotekami	npr. <code>.txt</code> , <code>.csv</code> , <code>.tex</code> , <code>.html</code> , <code>.py</code>
b	za delo s binarnimi datotekami	npr. slike

Nekaj lastnosti je zbranih v spodnji tabeli:

lastnost \ kombinacija črk	r	r+	x	x+	w	w+	a	a+
branje	x	x		x		x		x
pisanje		x	x	x	x	x	x	x
datoteka mora obstajati	x	x						
datoteka ne sme obstajati			x	x				
zbriše prejšnjo vsebino datoteke					x	x		
pisanje na konec datoteke							x	x

K zgornjim kombinacijam lahko dodamo še `t` ali `b`.