

Spoznavanje okolja

Osnove programiranja

Nejc Ilc

Programski jezik Python

Osnove programiranja bomo odkrivali z visokonivojskim jezikom  Python

<https://www.python.org>

"Si moram nanestiti Python?"

Ni nujno, lahko pa. Uporabljali bomo namreč orodje, ki ima Python že vgrajen.

Zakaj ravno Python?

- Python je eden najbolj priljubljenih programskih jezikov
- Enostaven za začetnike, a hkrati dovolj močan za stare mačke
- Koda je pregledna, čista, berljiva → zelo blizu psevdo kodi
- "Baterije so priložene" → obširna knjižnica razpoložljivih modulov
- Je kdo omenil kemijo? Python se lepo razume z orodji računalniške kemije (PyMOL, Maestro, RDkit)

Prvi zmenek

Python je interaktivni tolmač → ukaze sproti tolmači v jezik računalnika, tj. strojni jezik

Zagon ukaznega poziva na Windows

Pritisnite tipko , vtipkajte `cmd` in pritisnite tipko `enter`.

Nato vpišite `python` in pritisnite `enter`. To bo seveda delovalo le, če imate nameščen Python.

Začnimo pogovor v slogu osnovnošolske matematike:

```
>>> 1 + 1
2
>>> 0 - 42
-42
>>> 3.14 * 2
6.28
```

Označba `>>>` predstavlja naš vnos. Ko pritisnemo `enter`, se v vrstici nižje izpiše odgovor.

Osnovni operatorji

Matematični operatorji, ostale pogledamo pozneje

operator	opis	primer
+	seštevanje	$1 + 2 \rightarrow 3$
-	odštevanje	$1 - 2 \rightarrow -1$
*	množenje	$3.14 * 2 \rightarrow 6.28$
/	deljenje	$3 / 2 \rightarrow 1.5$
//	celoštevilsko deljenje	$3 // 2 \rightarrow 1$
%	ostanek pri deljenju (modulo)	$8 \% 3 \rightarrow 2$
**	potenciranje	$2 ** 3 \rightarrow 8$
()	oklepaji (določanje prednosti)	$(1 + 2) * 3 \rightarrow 9$

Podatkovni tipi

Vsaka vrednost oziroma podatek ima posebno lastnost: **podatkovni tip**.

Tip podatka torej govori o tem, kaj podatek predstavlja in kaj lahko z njim počnemo.

int

celo število (ang. *integer*)

Primeri: 0, 1, 42, -273, 1234567891234567890000000000000000000001

float

necelo število (ang. *floating point number*). Pozor: decimalna pika namesto "naše" decimalne vejice.

Primeri: 0.0, -42.33, 3.141592653589793

Podatkovni tipi: nadaljevanje

str

niz, zaporedje znakov (ang. *string*). Niz zapišemo v enojne `'`, dvojne `"` ali trojne `'''` narekovaje.

```
'Živjo, svet!'
```

```
"0.0"
```

```
"Moj program je kot 🚀"
```

```
"Sean O'Connery"
```

```
'F4+FvM CRFXRFMIRF+jf'
```

```
'Vzkliknil je: "Pri moji duši!"'
```

```
'42'
```

```
'''Vzkliknil je: "Pr' méj duš'!"'''
```

bool

logična vrednost (ang. *boolean*), lahko je bodisi *resnično* (`True`) bodisi *neresnično* (`False`). O tem podatkovnem tipu bomo precej govorili čez teden dni. Oplazili pa smo ga tudi pretekli teden, ko smo govorili o Evklidovem algoritmu in odločitvah ter zankah.

Podatkovni tipi: igranje

Kateri podatkovni tip dobimo, če ...

```
>>> 1 + 2
3
>>> 1 + 1.5
2.5
>>> 1 * 2
2
>>> 1 * 2.2
2.2
>>> 1 / 2
0.5
>>> 2 / 2
1.0
>>> 10 // 3
3
>>> 10 // 2.5
4.0
>>> 10 // 2.3
4.0
>>> 10 % 3
1
```

```
>>> 'Ana' + 'Jaka'
'AnaJaka'
>>> 'Ana+' + 'Jaka'
'Ana+Jaka'
>>> 'Dober' + ' ' + 'dan' + '!'
'Dober dan!'
>>> '1' + '1'
'11'
>>> '1 + 1'
'1 + 1'
>>> 'bla' * 3
'blablabla'
```

Kdor dela, greši

```
>>> 6 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

Hm, delili smo z 0, ups. Tega ne prenese niti papir.

```
>>> 'Ana' * 'Jaka'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
```

Nismo mislili takšnega razmnoževanja, kajne?

```
>>> 'bla' * 3.14
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'float'
```

Nize lahko razmnožujemo samo s celimi števili.

Kdor veliko dela, še bolj greši

```
>>> 1 + '1'  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

1 očitno ni isto kot '1', saj to že vemo. Števil in nizov ne znamo seštevati. Kaj pa naj bi dobili?

```
>>> '1' + 1  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can only concatenate str (not "int") to str
```

Podobno kot prej. Nizu '1' smo želeli prilepiti celo število 1, kar pa ne gre kar tako zlahka.

```
>>> 'Dober' + dan  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'dan' is not defined
```

Opa, na nekaj smo pozabili. dan bi moral biti niz 'dan', pri nas pa je kaj? "name"? Kaj je to?

Funkcije

Funkcije sprejemajo argumente in (lahko) vračajo rezultat.

Klic funkcije: `ime_funkcije(argument_1, argument_2, ...)`

- funkcijo vedno kličemo z oklepaji, tudi če ni ničesar v njih
- argumenti funkcije so navedeni v oklepajih in so ločeni z vejicami
- nekaj primerov:

funkcija	opis	primer
<code>abs</code>	absolutna vrednost	<code>abs(-42) → 42</code>
<code>pow</code>	potenca (podobno kot <code>**</code>)	<code>pow(2, 3) → 8</code>
<code>min</code>	najmanjša vrednost	<code>min(2, -42, 8) → -42</code>
<code>max</code>	največja vrednost	<code>min(2, -42, 8) → 8</code>
<code>print</code>	izpis v terminal	<code>print("Rezultat je", 42) → Rezultat je 42</code>

Vgrajene funkcije


Te funkcije so vedno dostopne. Seznam za Python 3.11.

Built-in Functions			
A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip() __import__()

Izraz

"Nekaj", kar se da izračunati

```
1 + 2
1+5 * 8+1
(1+5)*(8+1)
2**3 % 7
pow(2, 3, 7)
((4-2.2)**2 + (-2-1)**2)**0.5
max(min(0.5, 1), 0)
'Rekel je: ' + 'bla'*3 + '".'
```

Ko Pythonovi ukazni vrstici podamo izraz in pritisnemo , dobimo **rezultat** izraza.

```
>>> 1+5 * 8+1
42
```

Si lahko kam shranimo ta rezultat za mrzlo zimo?

a = 42

spremenljivka = izraz

Spremenljivka je na levi, na desni pa je izraz. Nikoli obratno!

Vmes je =, ki je operator prirejanja. To ni navaden enačaj.

Ta rezultat se nato shrani v spomin, ki ga lahko označimo, poimenujemo.

Namesto = bi si lahko prirejanje predstavili tudi s puščico ← :

$$a \leftarrow 42$$

Namesto 42 bi seveda lahko napisali poljuben izraz, recimo:

$$a = 1+5 * 8+1$$

Pravkar smo spoznali prireditveni stavek.

Spremenljivke

Včasih jim bomo rekli tudi preprosto **imena**.

Spremenljivko (ang. *variable*) uporabimo za shranjevanje vrednosti izraza.

```
>>> a = 42
```

Ko želimo uporabiti to shranjeno vrednost, preprosto rečemo `a`, takole:

```
>>> b = a + 1
>>> b
43
```

Če naslovim spremenljivko, ki ne obstaja:

```
>>> c * 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'c' is not defined
```



Fotografija: Jesse Orrico [vir]

Spremenljivke in uganke

Kakšno vrednost ima `b` po naslednjem zaporedju stavkov?

```
a = 15
b = a + 2 * a
```

Rešitev: `b` postane 45. Prvi stavek spremenljivki `a` priredi celo število 15. Nato se obdela drugi stavek, kjer se najprej izračuna izraz na desni strani: $15 + 2 * 15$, ki ima rezultat 45. Zatem se ta rezultat priredi spremenljivki `b`.

Še ena:

```
a = 15
b = a + 30 % a
```

Rešitev: `b` postane 15. Podobno kot prej, `a` najprej postane 15. V izrazu $15 + 30 \% 15$ se najprej izračuna ostanek pri deljenju 30 s 15, ki je 0, nato se ta 0 prišteje k 15 in rezultat je 15, ki se zatem priredi `b` ju.

Kaj pa zdaj?

```
a = 15
b = a
b = 'a'
```

Rešitev: `b` postane niz `'a'`. Najprej `a` postane 15. Nato `b` ju priredimo vrednost `a` ja, torej 15. V tretjem stavku pa `b` ju priredimo vrednost `'a'`, ki je mimogrede niz, in s tem povežimo prejšnjo vrednost.

Še zadnja! Koliko je `a` na koncu?

```
a = 15
a = a + 1
```

Rešitev: Točno tako, `a` postane 16. Najprej `a` postane 15. Nato v drugi vrstici izračunamo izraz na desni, ki je $a + 1$, torej $15 + 1$. Rezultat, ki je seveda 16, se nato vpiše v `a` in s tem poveži prejšnjo vrednost.

Imena spremenljivk

Še nekaj bontona, ko spremenljivkam dajemo imena

- vsebujejo lahko črke – najbolje, da samo črke angleške abecede
- vsebujejo lahko številke, vendar ne na prvem mestu

```
24kur = 'www.24kur.si'
```

- začnejo naj se z malo črko

```
pi = 3.14 in ne Pi = 3.14
```

- ne smejo vsebovati presledkov – več besed ločimo s podčrtajem

```
novo_geslo in ne novo_geslo ali novogeslo ali  
novoGeslo
```

- majhna ni velika

```
novo_geslo ni niti približno isto kot Novo_GESLO
```

- izogibamo se imenom vgrajenih funkcij

```
raje rečemo maks = max(2, 3)
```

```
in ne max = max(2, 3)
```

- ne moremo uporabiti teh (rezerviranih) besed:

```
False, None, True, and, as, assert, async,  
await, break, class, continue, def, del, elif,  
else, except, finally, for, from, global, if,  
import, in, is, lambda, nonlocal, not, or, pass,  
raise, return, try, while, with, yield
```


Dovolj!

Napišimo že kakšen program

Danes bo delno oblačno s temperaturo do 30 stopinj.

Oprostite, pingvini, za paniko, mislil sem 30 °F.

To je udobnih -1 °C.

Napišimo program, ki bo pretvarjal med °C in °F.

Prvi program: pretvornik med °C in °F

Kakšen bo naš algoritem, postopek? Kako ga bomo zakodirali?

1. Imamo številčno vrednost T_F , ki pomeni stopinje Fahrenheita.
2. Uporabimo naslednjo enačbo za pretvorbo v stopinje Celzija:

$$T_C = \frac{(T_F - 32)}{1,8} .$$

3. Izpišemo T_C .

Programsko kodo lahko napišemo v ukazno vrstico ...

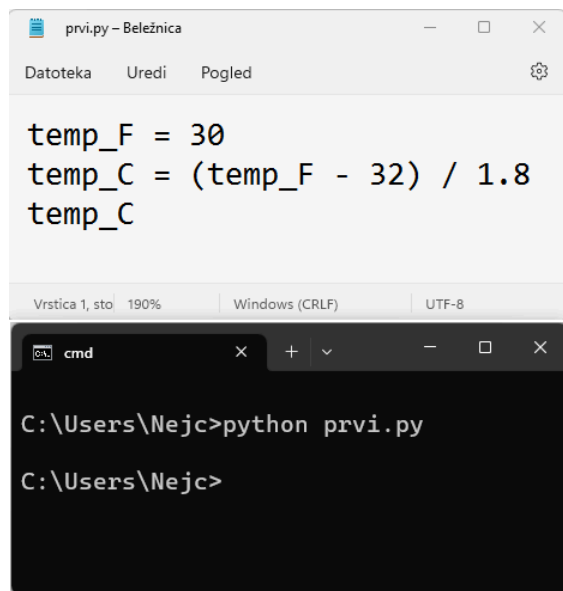
```
>>> temp_F = 30
>>> temp_C = (temp_F - 32) / 1.8
>>> temp_C
-1.1111111111111112
```

... kar pa je precej neuporabno. Za ponovni izračun pri drugi vrednosti T_F , bi morali ponoviti vse ukaze. Bolje?

Prvi program: napišimo ga

Kam? Na papir? Dober začetek. Potem pa v Wordov dokument! Saj te prime, pa te mine ...

Asketsko



```
prvi.py - Beležnica
Datoteka  Uredi  Pogled

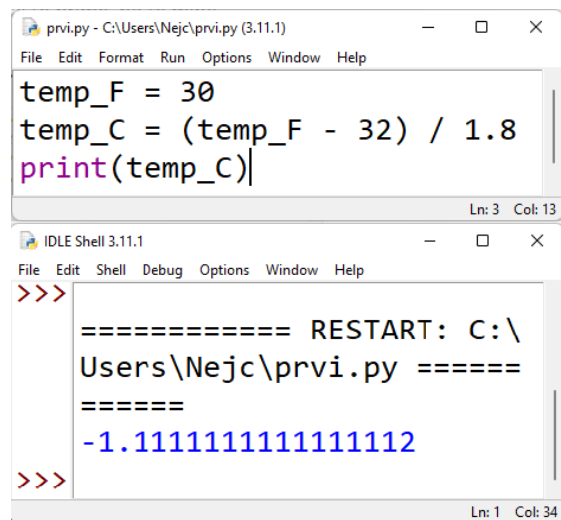
temp_F = 30
temp_C = (temp_F - 32) / 1.8
temp_C

Vrstica 1, sto 190%  Windows (CRLF)  UTF-8

cmd
C:\Users\Nejc>python prvi.py
C:\Users\Nejc>
```

Ups, program ničesar ne izpiše. V zadnji vrstici dodajmo `print`.

Za silo



```
prvi.py - C:\Users\Nejc\prvi.py (3.11.1)
File Edit Format Run Options Window Help

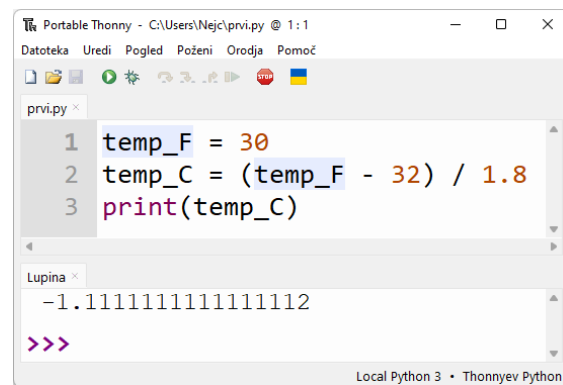
temp_F = 30
temp_C = (temp_F - 32) / 1.8
print(temp_C)
Ln: 3 Col: 13

IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help

>>>
===== RESTART: C:\
Users\Nejc\prvi.py =====
=====
-1.1111111111111112
>>>
Ln: 1 Col: 34
```

Razvojno okolje IDLE

Pr' Toniju



```
Portable Thonny - C:\Users\Nejc\prvi.py @ 1:1
Datoteka  Uredi  Pogled  Poženi  Orodja  Pomoč

prvi.py
1 temp_F = 30
2 temp_C = (temp_F - 32) / 1.8
3 print(temp_C)

Lupina
-1.1111111111111112
>>>

Local Python 3 • Thonnyev Python
```

Razvojno okolje Thonny

Urejevalnik kode - razvojno okolje

Programiranje je udobnejše z integriranim razvojnim okoljem (ang. integrated development environment - IDE)

IDLE

- Zelo preprost (zanimivost: napisan je v Pythonu)
- Ko namestite Python, se privzeto namesti tudi IDLE
- IDLE → Integrated Development and Learning Environment

Thonny

- Nekoliko, khm, "lepši" kot IDLE
- Zasnovan za učenje programiranja
- Navodila za namestitev

Drugo: Visual Studio Code, PyCharm, Spyder

Pogovor z uporabnikom

print

Ko želimo uporabnika našega programa o čem obvestiti, to najlažje storimo s funkcijo `print`, ki izpisuje v terminal (lupina/konzola/ukazni poziv). Pravimo tudi, da izpisuje na **standardni izhod**.

```
>>> print('Odgovor na vprašanje o veselju, življenju in sploh vsem je:', 1+5*8+1, '!'*3)
Odgovor na vprašanje o veselju, življenju in sploh vsem je: 42 !!!
```

input

Ko želimo od uporabnika našega programa dobiti kakšen podatek, ga zanj prosimo s funkcijo `input`.

```
>>> najljubsi_okus_sladoleda = input('Tvoj najljubši okus 🍌? ')
Tvoj najljubši okus 🍌? Jogurt z gozdnimi sadeži
>>> print(najljubsi_okus_sladoleda, 'je tvoj najljubši okus, kajne?')
Jogurt z gozdnimi sadeži je tvoj najljubši okus, kajne?
```

Pravimo, da funkcija `input` bere s **standardnega vhoda**.

Prvi program: poskus izboljšave

Psihološki profil našega prvega programa (glej tri strani nazaj)

Ta program je asocialen. Ne zanima ga mnenje drugega. Ima samo svoj prav.

Njegovo vedenje je družbeno nesprejemljivo, ne pozna osnov bontona. Ni vljuden. Je pa matematični genij, to pa.

Dodajmo ščepec komunikacijskih veščin in nekaj osnovne čustvene inteligence.

```
temp_F = input('Vnesi temperaturo v °F: ')
temp_C = (temp_F - 32) / 1.8
print(temp_F, '°F je', temp_C, '°C.')
```

Ojoj, spet rdeče! Kaj je tu narobe? Kje tiči napaka? Berimo ...


```
Traceback (most recent call last):
  File "prvi-v2.py", line 2, in <module>
    temp_C = (temp_F - 32) / 1.8
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

V katerem grmu torej tiči zajec? No, a je zajec? Zajčki so navadno srčkani. Zato raje recimo ščurek ali kar hrošč. Kje v našem programu torej tiči hrošč?

Razhroščevanje

Podobno kot razkoščičevanje češenj: da si ne polomimo zob, ko jemo češnjev štrudelj.

Navodila za Thonnyja:

1. Imamo programsko kodo v urejevalniku (bodisi jo napišemo na novo bodisi odpremo obstoječo datoteko).
2. Zaženemo razhroščevalnik na enega od načinov:
 - v orodni vrstici kliknemo na gumb  ali
 - v meniju izberemo Poženi → Razhroščevanje trenutne skripte (lepše) ali
 - pritisnemo kombinacijo tipk `Ctrl + F5`.
3. Aktivira se razhroščevalnik, ki ustavi izvajanje programa **pred izvajanjem** prvega stavka.

```
1 temp_F = input('Vnesi temperaturo v °F: ')
```

4. Skozi program se sedaj premikamo z ukazi:



Stopi čez stavek `F6`



Stopi v stavek `F7`



Stopi ven (iz česa?)



Nadaljuj `F8`

(se nadaljuje)

Razhroščevanje: nadaljujmo

5. Izvedimo prvi stavek tako, da pritisnemo `F6`. Pozvani bomo, da vnesemo temperaturo v °F. Ubogajmo in nato pritisnimo `enter`.

6. Ko smo v drugi vrstici, pritiskajmo `F7` in opazujmo postopno računanje izraza. Kmalu zagledamo tole:

```
1 temp_F = input('Vnesi temperaturo')
2 temp_C = ('30' - 32) / 1.8
```

7. Spremenljivka `temp_F` očitno vsebuje *niz* `'30'` in ne *števila* `30`. Zahtevamo, da Python od niza odšteje celo število. Preveč smo zahtevni. Pametnejši odneha. Dobimo obvestilo o napaki.

8. Našli smo hrošča! Tiči v drugi vrstici našega programa.

Kako je prišel tja? Napačno smo predpostavljali, da se bo vpisana temperatura uporabila kot število.

Pa se ni. Korenine te napake rastejo iz funkcije `input`, ki vrne uporabnikov odgovor in to **vedno kot niz**.

9. Kako streti tega hrošča? Vrednost v `temp_F` moramo pretvoriti v število.

Pretvorba podatkovnih tipov

Spomnimo se: podatki (spremenljivke) imajo svoj podatkovni tip. Funkcija `type()` nam vrne podatkovni tip.

niz → število

Uporabimo funkcijo `int()` ali `float()`.

```
a = '1'
b = '2.2'
a_num = int(a)
b_num = float(b)
print(a_num + b_num)
```

Programček izpiše `3.2`, lepo!

Kaj bi dobili, če bi rekli `float(a)`? `1.0`.

Kaj bi dobili, če bi rekli `int(b)`?

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10:'2.2'
```

število → niz

Uporabimo funkcijo `str()`.

```
a = 1
b = 2.2
a_str = str(a)
b_str = str(b)
print(a_str + b_str)
```

Programček izpiše `12.2`, tudi lepo!

```
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> type(c)
<class 'str'>
```

Prvi program: to je to

Na krilih novih spoznanj dokončajmo delo

Našo kodo tudi dokumentirajmo: uporabimo komentarje (znak #) ali več-vrstične nize.

```
"""Pretvornik enot
```

```
Program pretvarja vnešeno temperaturo  
iz stopinj Fahrenheita v stopinje Celzija.  
"""
```

```
temp_F = input('Vnesi temperaturo v °F: ')  
temp_F = float(temp_F) # Niz pretvorimo v število  
temp_C = (temp_F - 32) / 1.8 # 1.8 = 9/5  
print(temp_F, '°F je', temp_C, '°C.') # Izpis
```

Pritisnemo na **F5** in zadržimo dih ...

```
Vnesi temperaturo v °F: 30  
30.0 °F je -1.111111111111112 °C.
```

👏🌟🏆💪👉 Juhuhu! 🙌



Fotografija: Ilse Orsel [vir]

