PROGRAMIRANJE 1

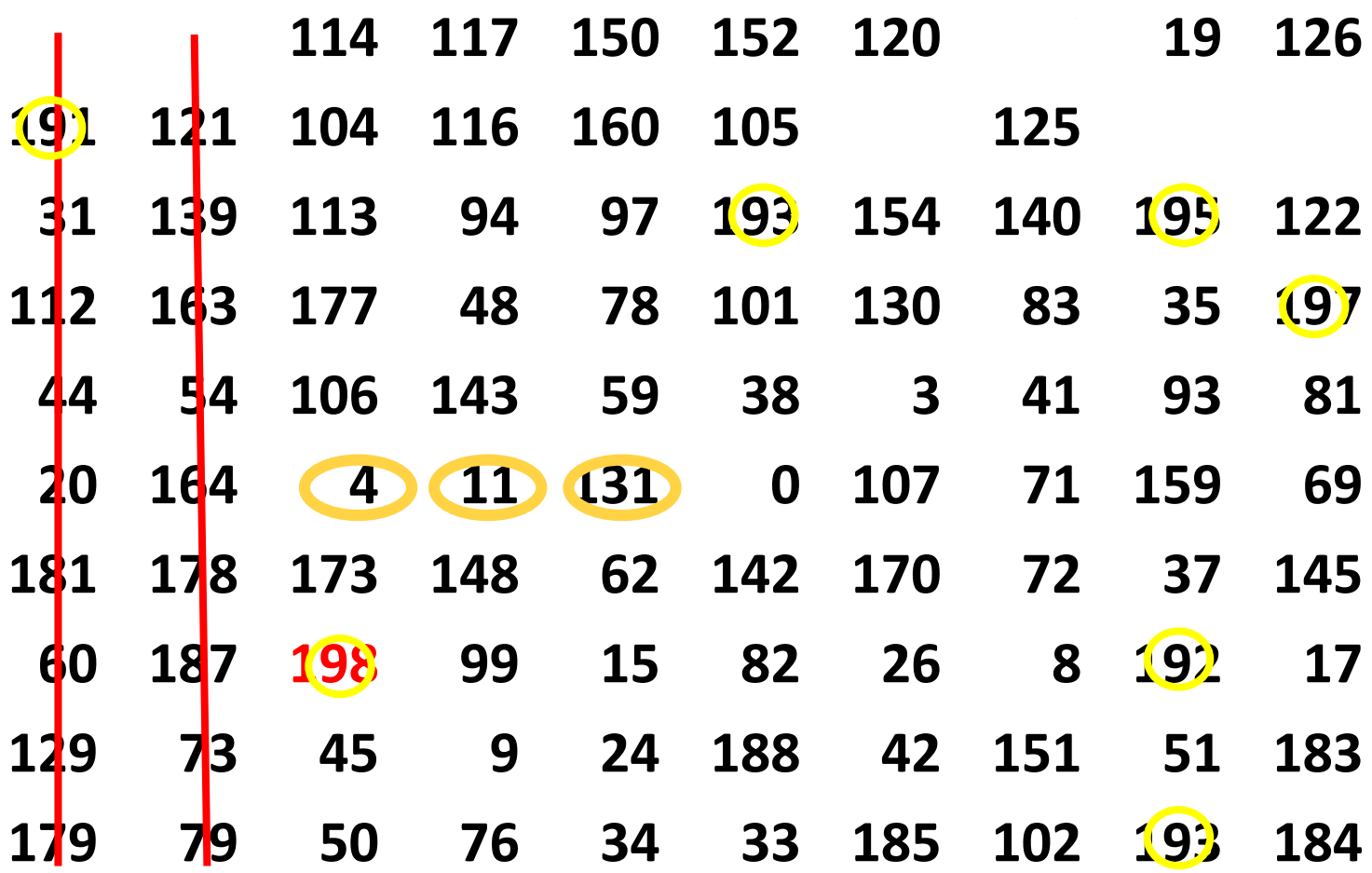
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
SVEUČILIŠTE U SPLITU

python™

- Pronađite najveći broj u popisu brojeva
- Obratite pažnju na koji to radite, nije bitno da to uspijete napraviti

Pronađite broj 198

25	1	114	117	150	152	120	46	19	126
191	121	104	116	160	105	89	125	40	14
31	139	113	94	97	193	154	140	195	122
112	163	177	48	78	101	130	83	35	197
44	54	106	143	59	38	3	41	93	81
20	164	4	11	131	0	107	71	159	69
181	178	173	148	62	142	170	72	37	145
60	187	198	99	15	82	26	8	192	17
129	73	45	9	24	188	42	151	51	183
179	79	50	76	34	33	185	102	193	184


114 117 150 152 120 19 126
121 104 116 160 105 125
31 139 113 94 97 193 154 140 195 122
112 163 177 48 78 101 130 83 35
44 54 106 143 59 38 3 41 93 81
20 164 4 11 131 0 107 71 159 69
181 178 173 148 62 142 170 72 37 145
60 187 198 99 15 82 26 8 17
129 73 45 9 24 188 42 151 51 183
179 79 50 76 34 33 185 102 184

python™

Pronađite broj 197

184	183	152	153	154	155	156	157	158	159
185	182	151	150	149	148	147	146	145	160
186	181	180	179	178	177	176	175	144	161
187	188	189	190	191	192	193	174	143	162
208	207	206	205	204	203	194	173	142	163
209	210	211	212	213	202	195	172	141	164
228	227	226	225	214	201	196	171	140	165
229	230	231	224	215	200	197	170	139	166
248	247	232	223	216	199	198	169	138	167
249	246	233	222	217	218	219	168	137	136
250	245	234	221	220	255	254	253	252	135
251	244	235	236	237	238	239	240	241	242
128	129	130	131	132	133	134	243	1	127

python™

179	79	50	76	34	33	185	102	193	184
129	73	45	9	24	188	42	151	51	183
60	187	198	99	15	82	26	8	192	17
181	178	173	148	62	142	170	72	37	145
20	164	4	11	131	0	107	71	159	69
44	54	106	143	59	38	3	41	93	81
112	163	177	48	78	101	130	83	35	197
31	139	113	94	97	193	154	140	195	122
191	121	104	116	160	105	89	125	40	14
25	1	114	117	150	152	120	46	19	126

Nađite najveći broj

3 41 12 9 74 15

3 41 12 9 74 15

najveći **-1 3 41 74**

Tipovi podataka

- **Podatak** je broj, pojedinost, nevrednovana činjenica, koncept, opis, pojava, a što se može prevesti u računalu prihvatljiv (i razumljiv) oblik.
- Skup podataka i operacije na njima čine **tip podataka**. Tip podataka je, dakle, skup vrijednosti i skup operatora koji se mogu primijeniti na te vrijednosti.
- Svaki podatak je prikazan **literalom** (niz simbola koji ima konstantnu vrijednost), na primjer,

```
3.1415, 1234, 999L, 3+4j, 'more', "neš' ti", true
```

Jednostavni tipovi podataka

- **brojčani tipovi podataka** (cijeli, realni i kompleksni brojevi)
- **tekstualni tipovi podataka** (nizovi znakova)
- **logički tipovi podataka** (istina i laž, 0 i 1)

Brojčani tip podataka

- sadrže samo sljedeće literale:
  - znamenke 0-9
  - opcionalni znak za predznak (+ ili -)
  - moguću decimalnu točku (u prikazima brojeva se nikad ne koristi zarez)
  - slovo e se koristi u znanstvenom prikazu brojčanih podataka

- **Brojčani podaci** mogu biti s ili bez decimalne točke.
- Ako broj sadrži decimalnu točku, onda se radi o **broju s pomičnom decimalnom točkom** (eng. floating-point value - float), inače se radi o **cijelim brojevima** (eng. integer – int, dekadski, binarni, oktalni, heksadekadski) ili **kompleksnim brojevima** (eng. complex)

Literal	Brojčani tip podatka
1234, -24, 0	cijeli brojevi
1.23, 3.14e-10	broj s pomičnom decimalnom točkom
0b1111, 0o177, 0x9ff	binarni, oktalni i heksadekadski brojevi
3+4j, 3.0+4.0j	kompleksni brojevi

Tekstualni tip podataka

- **Tekstualni podaci** ili **stringovi** (eng. string – **str**) su nizovi znakova (slova, znamenke, specijalni znakovi, praznina), na primjer,

```
'Dobar dan'
'Ivo, Ivić'
"Teslina 12, Split 21000"
"1+1"
"!@#$%^ &*()"
```

- U Pythonu, stringovi mogu biti ograničeni parom jednostrukih (') ili dvostrukih (“) navodnika.
- String koji se sastoji samo od para navodnika (ništa između njih) se naziva **prazan string** (eng. empty string), te se on razlikuje od stringa koji sadrži samo praznine (eng. blank string).

- U Pythonu ne postoji tip znak – char
  - Znak je naprosto vrijednost tipa str duljine jedan

- Posebni znaci za oblikovanje teksta

Posebni znak	Opis djelovanja
`\n`	prijelaz u novi redak
`\t`	tabulator
`\\`	ispisati lijevo ukošenu crtu
`\'`	ispisati jednostruki navodnik
`\"`	ispisati dvostruki navodnik

Logički tip podataka

- Tip **bool** ima dvije vrijednosti **True** i **False**.
  - Vrijednosti logičkog tipa mogu se zapisati i kao brojevi 1 i 0
  - Python će svaki broj različit od 0 smatrati da je True, dok se samo broj 0 smatra False
  - true i True nije jednako → case sensitive

Operatori i izrazi

- **Operator** je simbol koji predstavlja operaciju koja se treba izvršiti nad jednim ili više operanada.
- **Operand** je vrijednost na koju djeluje operator
  - **Unarni operator** djeluje nad samo jednim operandom
  - **Binarni operator** djeluje nad dva operatora
- **Izraz** je svaka kombinacija **operatora** i **operanada** koju programski jezik dozvoljava.
  - Svaki izraz ima svoju vrijednost (određenog tipa) koja se dobiva izvršavanjem svih operacija u izrazu, redoslijedom prema prioritetu i asocijativnosti operacija.
  - **Podizraz** je bilo koji izraz koji je dio većeg izraza.

Aritmetički operatori i izrazi

Struktura	Naziv	Primjer	Rezultat
- x	promjena predznaka	- 9	-9
x + y	zbrajanje	15 + 25	40
x - y	oduzimanje	15 - 25	-10
x * y	množenje	8 * 5	40
x / y	dijeljenje	35 / 10	3.5
x // y	cjelobrojno dijeljenje	35 // 10	3
x % y	ostatak pri dijeljenju	35 % 10	5
x ** y	potenciranje	5 ** 3	125

- Isti simbol – se koristi kao unarni operator za promjenu predznaka i kao binarni operator za oduzimanje.

```
20 - 5 → 15 ( - kao binarni operator)
- 10 * 2 → - 20 ( - kao unarni operator)
```

- Izrazi koji rezultiraju brojevima se nazivaju **aritmetičkim izrazima**.
- Na primjer, u izrazu 5 + (3 * 2), dva operanda za operator zbrajanja su 5 i (3 * 2), pa je rezultat jednak 11. Da je izraz bio napisan kao (5 + 3) * 2, rezultat bi bio jednak 16.

Logički operatori i izrazi

- Algebra sudova sadrži skup **logičkih operatora** – I (and), ILI (or), NE (not).
- **Logički izraz** je izraz koji **rezultira logičkom vrijednosti** i ne mora uvijek sadržavati logičke operatore.
- Relacijski izrazi su, stoga, također logički izrazi. Logičke vrijednosti istina i laž su, po definiciji, logički izrazi.
- Svaki logički izraz ima svoju tablicu istinitosti u kojoj se prikazuje vrijednost koju će rezultat logičkog izraza imati za određene vrijednosti operanada.

x	y	x I y - konjukcija	x ILI y - disjunkcija	NE x - negacija
laž	laž	laž	laž	istina
istina	laž	laž	istina	laž
laž	istina	laž	istina	
istina	istina	istina	istina	

python™

Relacijski operatori i izrazi

- **Relacijski operatori** su operatori kojima vršimo usporedbe vrijednosti operanada.
- **Relacijski izrazi rezultiraju logičkom vrijednosti** istina (true) ili laž (false).
- Relacijski operatori se mogu primijeniti na bilo koji skup vrijednosti koji ima redoslijed
- Relacijski operatori se dijele na: standardne ili "uređajne", te na relacijske operatore jednakosti.

Struktura	Naziv	Primjer	Rezultat
x == y	jednako	12 == 12	istina
x != y	različito	12 != 12	laž
x < y	manje od	12 < 20	istina
x > y	veće od	'Ana' > 'Branka'	laž
x <= y	manje ili jednako od	12 <= 12	istina
x >= y	veće ili jednako od	'B' >= 'G'	laž

Operatori za stringove

Struktura	Naziv	Primjer	Rezultat
str1 + str2	nadovezivanje	'Ana'+'Anić'	'AnaAnić'
str1 * x	uvišestručenje	'Ana'*3	'AnaAnaAna'
str1 in str2	prvi string sadržan u drugom stringu	'an' in 'banana'	istina
str1 not in str2	prvi string nije sadržan u drugom stringu	'ab' in 'banana'	laž

- Za operator **uvišestručenja** * vrijedi zakon komutativnosti, tj. str1 * x = x * str1.
- Operatori **pripadnosti** (eng. membership operators) in i not in su logički operatori za rad sa stringovima, imaju rezultat istina ili laž, te su komplementarni (ako je izraz str1 in str2 istina, onda je tvrdnja str1 not in str2 laž, vrijedi i obrnuto).

Prioritet i asocijativnost operatora

Redni broj	Operator
1	** potenciranje
2	- (promjena predznaka)
3	množenje (ili uvišestručenje), / dijeljenje, // cjelobrojno dijeljenje, % ostatak pri dijeljenju
4	+ zbrajanje (ili nadovezivanje), - oduzimanje
5	<, >, <=, >=, !=, == relacijski operatori
6	in, not in - operatori pripadnosti
7	NE negacija
8	I konjukcija
9	ILI disjunkcija

- Redoslijed izvođenja operacija

Redni broj	Operacija
1	aritmetički
2	relacijski
3	logički

- Postavlja se pitanje kako razriješiti redoslijed izvršavanja operatora koji imaju isti prioritet izvršavanja.
- Kad imamo više operatora istog prioriteta, redoslijed izvršavanja određen je pravilom **asocijativnosti operatora** (eng. operator acociativity).
- Jedino se kod potenciranja izraz izvršava **s desna na lijevo**

- Svi aritmetički operatori se izvršavaju prije relacijskih ili logičkih operatora.
- Svi relacijski operatori se izvršavaju prije logičkih operatora
- Kao i kod aritmetičkih operatora, i kod logičkih je dobro koristiti zagrade, iako nisu potrebne, zbog povećanja jasnoće i čitljivosti izraza

Ekvivalentnost logičkih izraza

- Zanimljivo je razmotriti kako se zamjenom relacijskih operatora pri oblikovanju nekog uvjeta mijenja oblik logičkog izraza koji opisuje taj uvjet.
- Nekada nam to može pomoći pri pojednostavljivanju programa i, što je još važnije, olakšati razumijevanje problema koji rješavamo.

Operator	Suprotni operator
>	<=
<	>=
>=	<
<=	>
==	!=
!=	==

Izraz		Ekvivalentni izraz
x < y	⇔	NE (x >= y)
x <= y	⇔	NE (x > y)
x == y	⇔	NE (x != y)
x != y	⇔	NE (x == y)
NE (x I y)	⇔	(NE x) ILI (NE y)
NE (x ILI y)	⇔	(NE x) I (NE y)

Varijable

- Svi računalni programi na neki način obrađuju podatke.
- Podaci se sakupljaju, pohranjuju, obrađuju i mijenjaju tijekom izvođenja programa.
- Pojedinačni **podatak** nazivat ćemo **vrijednošću** (eng. value)
- Svaka vrijednost je određenog **tipa podataka**
- **Varijabla** je simboličko ime koje je povezano s vrijednošću i pridruženo lokacijama u memoriji u koje možemo pohraniti neke vrijednosti. Dakle, svaka varijabla ima svoje **ime** i vrijednost (ili sadržaj) određenog **tipa podataka**

- Svi podaci koje program koristi moraju biti pohranjeni u memoriji.
- Problem pamćenja vrijednosti riješen je uvođenjem varijabli.
- Varijable su imena koja će biti pridružena pojedinim vrijednostima.

- **Memorijska lokacija** u kojoj je pohranjen podatak je jedinstveno određena svojom adresom. Da biste mogli koristiti pohranjene podatke, morate znati adresu memorijske lokacije na kojoj su pohranjeni.

- Da bi se olakšao pristup podacima, koriste se varijable koje služe kao spona između programa i memorijskih lokacija s podacima, jer se varijabla pridružuje adresi memorijske lokacije.

- Važno je znati da se prema različitim tipovima podataka treba različito odnositi.
  - Na primjer, različiti tipovi brojeva zahtijevaju različitu količinu memorije za pohranu.

- Također, nekim s tipovima podataka možemo raditi ono što s nekim drugima ne možemo.
  - Na primjer, brojeve možemo zbrajati, oduzimati, množiti i dijeliti, dok stringove ne možemo.
  - S druge strane, stringove možemo ulančavati.

Imenovanje varijabli

- Svako **ime varijable** mora započeti sa slovom (preporuča se) ili podvlakom (eng. underscore character _).
- U nazivima varijabli se ne mogu koristiti interpunkcijski znakovi i operatori.
- Python razlikuje velika i mala slova u nazivima varijable, pa razlikuje varijable sunce i Sunce.
- Imena varijabli koja započinju s podvlakom se koriste za imenovanje varijabli s posebnim karakteristikama.
- Nakon prvog slova u imenu može slijediti bilo koja kombinacija slova, brojeva i podvlaka. Ime varijable može biti proizvoljne duljine.

- Često je korisno imenovati varijablu uz pomoć fraze. U tom se slučaju preporuča korištenje stila imenovanja pod nazivom „mala slova s podvlakom“ (eng. „lower with under“).
- Radi se o načinu imenovanja kod kojeg se ime varijable piše malim tiskanim slovima, a riječi fraze koja čini to ime se povezuju podvlakom.
- Na primjer, dobra imena varijabli koja su u stvari fraze su: doba_dana, radijus_kruga.

- Programski jezik sadrži **ključne riječi** – rezervirana imena
  - ključne riječi ne mogu biti imena varijabli

```
int float bool str
False class fi nally is return
None continue for lambda try
True def from nonlocal while
and del global not with
as elif if or yield
assert else import pass
break except in raise
```

Pridruživanje vrijednosti

```
varijabla = izraz
```

- Znak = nazivamo znakom pridruživanja
  - ne smije se poistovjetiti sa znakom jednakosti u matematici gdje on označava da je ono što stoji lijevo od znaka jednakosti jednako onome desno od znaka jednakosti.
  - jednakost x = x + 1 u matematici nema nikakvog smisla dok je programskim jezicima potpuno smislena konstrukcija

- Pridruživanje se obavlja tako da se najprije izračuna vrijednost izraza s desne strane znaka pridruživanja i nakon toga se ta vrijednost pridruži varijabli koja se nalazi s lijeve strane znaka pridruživanja.

Deklaracija i inicijalizacija

- **Deklaracija** određuje ime i tip varijable.
- Nužna je kod nekih programskih jezika, ali je nepotrebna u Pythonu.
- **Inicijalizacija** varijable je prvo pridruživanje vrijednosti varijabli.
- Deklaracija (ili inicijalizacija kod programskih jezika koji ne zahtijevaju deklaraciju) rezervira memorijski prostor, tj. pridružuje memorijsku lokaciju varijabli.
- Međutim, vrijednost varijable je definirana tek kod pridruživanje vrijednosti toj varijabli.

- Brojčane varijable se inicijaliziraju na vrijednost 0
- String varijable se inicijaliziraju na vrijednost “” (**prazan string**)

- Varijabli se mogu pridruživati različite vrijednosti tijekom izvršavanja programa – otuda i ime varijabla, odnosno, promjenjiva veličina.
- Kada se varijabla pojavljuje u programu (osim na lijevoj strani pridruživanja), ono što se koristi za izvršavanje naredbi je vrijednost varijable, a ne njeno ime.

Konstante

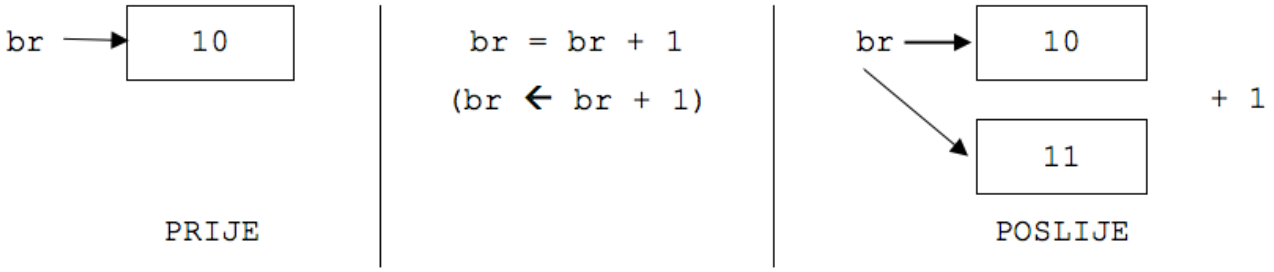
- **Konstante** (eng. constant) su posebne vrste varijabli kojima se vrijednost pridruži samo jednom tijekom cijelog izvršavanja programa, tj. to su varijable s konstantnom vrijednošću
- Konstanta je varijabla kojoj se vrijednost ne mijenja tijekom izvođenja programa.
- Na primjer, matematička konstanta pi=3.14
- Pošto smo pojam konstantne vrijednosti već koristili prilikom definiranja literala, moramo naglasiti da literali i konstante nisu isti pojmovi. Naime, u prethodnom izrazu, pi je varijabla, a 3.14 je brojčani literal.

- Kada varijabli pridružimo vrijednost izračunatog izraza ona će postati istog tipa kao i desna strana izraza.

- Tijekom uporabe varijabla može sadržavati vrijednosti različitih tipova – dakle, ona može mijenjati svoj tip

- varijable su reference na objekte

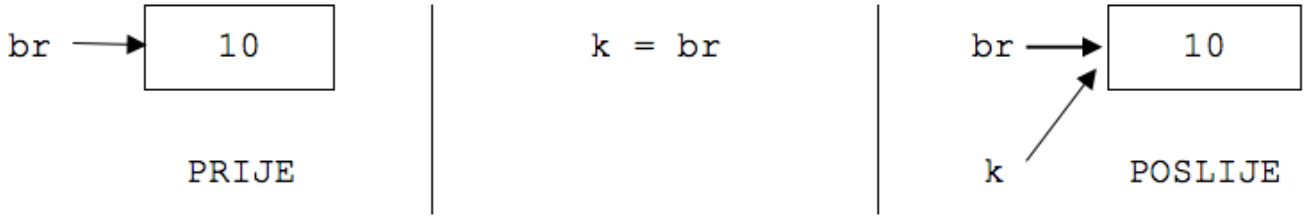
```
>>> br = 10
```



- što ako varijabli pridružimo vrijednost druge varijable?

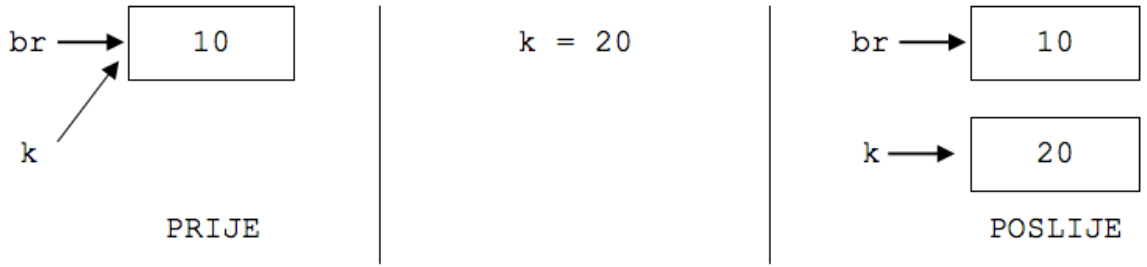
→ dijeljene reference

```
>>> br = 10
>>> k = br
```



- što ako nakon toga varijabli a dodijelimo novu vrijednost ?

```
>>> br = 10
>>> k = br
>>> k =20
```



- stvoren je novi objekt, s kojim se povezuje varijabla k
- ne utječe se na varijablu br – ona i dalje pokazuje na isti objekt
- varijabla k može pokazivat na objekt drugačijeg tipa

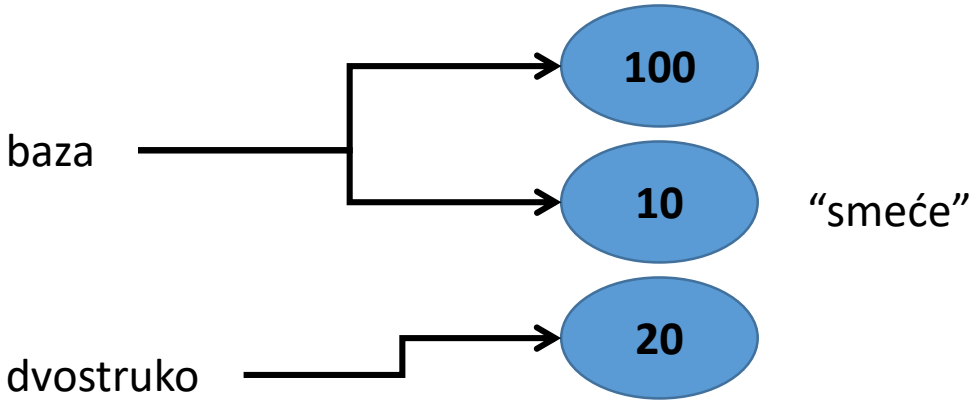
→ tip je svojstvo objekta, a ne imena (varijable)

- kada se varijabla poveže s novim objektom, Python provjerava referencira li “napušteni” objekt neka druga varijabla (ili neki drugi objekt) i preuzima objekte bez reference

  → *garbage collection*

- Primjer: kako se i kada mijenjaju sadržaji na koje pokazuju varijable

```
>>> baza = 10
>>> dvostruko = 2 * baza
>>> dvostruko
20
>>> baza = 100
>>> dvostruko
20
```



- Ulančano pridruživanje
  - kada većem broju različitih varijabli želimo pridružiti istu vrijednost
    - a = b = c = d = 5

- Višestruko pridruživanje

```
varijabla_1, varijabla_2, ... = izraz_1, izraz_2, ...
```

- S desne strane znaka pridruživanja mora se nalaziti onoliko izraza odvojenih zarezima koliko je s lijeve strane imena varijabli

Složeni operator pridruživanja

- izraz oblika

```
izraz_1 op= izraz_2
```

- gdje je op jedna od operacija +, -, *, /, %, ekvivalentan je s

```
izraz_1 = izraz_1 op (izraz_2)
```

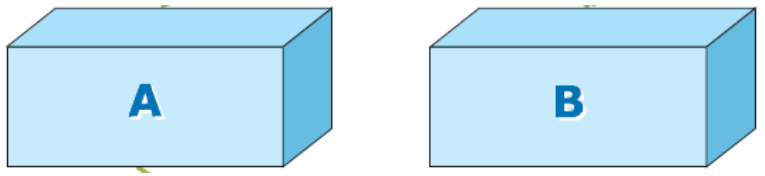
Skraćeni zapis	Ekvivalentno značenje
`a += 2`	`a = a + 2`
`a -= 2`	`a = a - 2`
`a /= 2`	`a = a / 2`
`i *= j + 1`	`i = i * (j + 1)`
`a //= 2`	`a = a // 2`
`a %= 2`	`a = a % 2`
`a **= 2`	`a = a ** 2`

- Najčešće korišteni skraćeni zapis je **inkrement**, na primjer a += 1

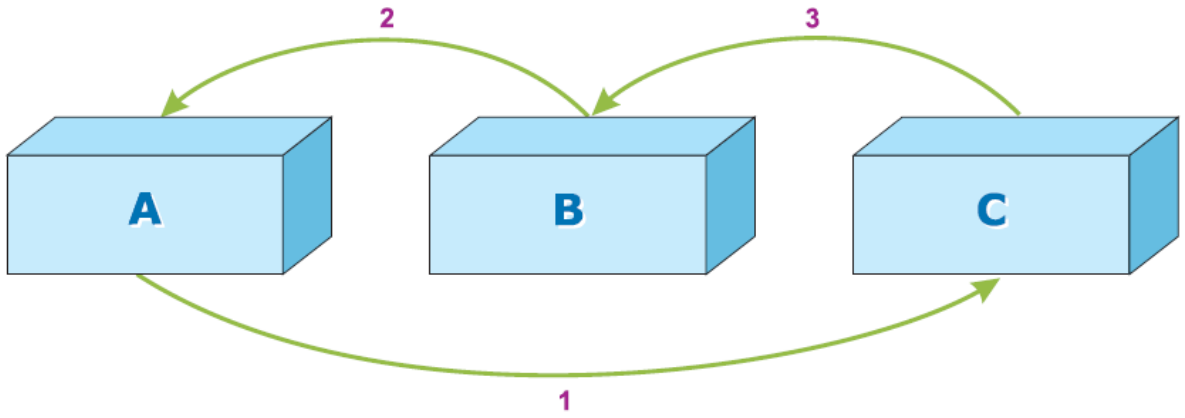
Zamjena vrijednosti varijabli

- Pretpostavimo da imamo 2 akvarija.
  - U prvom akvariju (nazovimo ga A) nalaze se ribice roda skalari, dok se u drugom akvariju (B) nalaze pirane.
  - Mi želimo skalare premjestiti u akvarij B dok pirane želimo premjestiti u akvarij A.
  - Ukoliko prvo stavimo pirane u akvarij A pirane će pojesti skalare.
  - Ukoliko prvo skalare stavimo u akvarij B ponovo će pirane pojesti skalare.

```
>>> a = skalari
>>> b = pirane
>>> a = b
>>> b = a
```



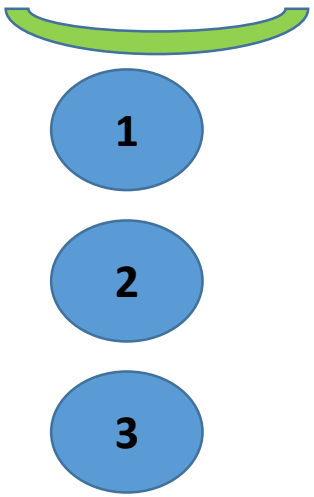
- Rješenje je uvođenje trećeg akvarija (nazovimo ga C)
  - premjestimo skalare u akvarij C
  - premjestimo pirane u akvarij A
  -




```
>>> a = skalari
>>> b = pirane
>>> c = a
>>> a = b
>>> b = c
```

```
>>> a = skalari
>>> b = pirane
>>> a, b = b, a →Python
```

```
>>> x = 1
>>> y = 2
>>> z = 3
>>> x, y, z = z, x, y
```



temp1

Varijable i izrazi

- Ako pridijelite izraz varijabli, izraz se izračuna i vrijednost varijable postaje izračunata vrijednost.

```
x = 500 + (10 * 7)
print (x)
```

```
570
```

- Također se *varijable* mogu koristiti kao izrazi.

```
br,vr = 50,2
udaljenost = br * vr
print (udaljenost)
```

```
100
```

- Isto tako se mogu kombinirati varijable s brojevima u izrazima.

```
X = 100
Y = X * 7
print (Y)
```

```
700
```

- PRINT ispisuje izračunatu vrijednost izraza.

```
print (512 + 478)
```

```
990
```

- Ako zatvorite izraz s navodnicima, onda on postaje string i neće se izračunati. Na primjer:

```
print ("512 + 478")
```

```
512 + 478
```

Varijable i stringovi

- String varijabla

```
X = "Pozdrav svima"
print (X)
```

```
Pozdrav svima
```

- String se može dodati na kraj postojeće string varijable.

```
X = "Pozdrav"
X = X + "svima"
print (X)
```

```
Pozdravsvima
```

- Možete također dodavati string varijable jedna drugoj.

```
a,b,c ="S1","S2","S3"
d = a + b + c
print (d)
```

```
S1S2S3
```


python™

Zadaci

- Obavezno pročitati do sljedećih vježbi i predavanja:

  “Rješavanje problema programiranjem u Pythonu”

  stranice: 15 – 60, 167-170