

---

# MAINĪGIE, TO DATU TIPI, PAMATDARBĪBAS DARBAM AR TIEM PROGRAMMĒŠANAS VALODĀ PYTHON

Gerda Fedotova 2PT

2024

---

---

# IEVADS

Python programmēšanas valoda ir plaši izmantota un populāra tās vienkāršības dēļ. Šajā materiālā ir aprakstīti pamati darbam ar mainīgajiem, datu tipi un pamatdarbības, ko var veikt ar šiem mainīgajiem.

---

# MAINĪGIE (1)

Mainīgie ir “kastes”, kurās tiek glabāti dati. Python mainīgie tiek izveidoti brīdī, kad tiem tiek piešķirta vērtība.

**Deklarēšana un vērtības piešķiršanas piemērs:**

```
1  a = 5
2  b = True
3  c = "Hello, world!"
```

Python programmēšanas valodā, deklarējot mainīgo, nav jāraksta tā datu tips

---

# MAINĪGIE (2)

Mainīgajam var būt īss nosaukums (x, y) vai aprakstošāks nosaukums (punktiKopa, dzimsanasGads). Python mainīgo lielumu noteikumi:

- Mainīgā nosaukumam jā sākas ar burtu vai pasvītrojuma rakstzīmi, nevar sākties ar skaitli
- Mainīgā nosaukumā var būt tikai burtciparu rakstzīmes un pasvītrojumi
- Mainīgo nosaukumi ir reģistrjutīgi (x un X ir trīs dažādi mainīgie)
- Mainīgā nosaukums nevar būt neviens no Python atslēgvārdiem.

---

# MAINĪGIE (3)

DERĪGS MAINĪGĀ NOSAUKUMS:

- `mansVards= "Gerda"`
- `mans_vards = "Gerda"`
- `_mans_vards = "Gerda"`
- `mansvards = "Gerda"`
- `MANSVARDS = "Gerda"`
- `mansvards2 = "Gerda"`

NEDERĪGS MAINĪGĀ NOSAUKUMS:

- `2mansvards = "Gerda"`
- `mans-vards = "Gerda"`
- `mans vards = "Gerda"`
- `mansVārds = "Gerda"`

---

# VAIRĀKVĀRDU MAINĪGO NOSAUKUMI

## Camel Case

Katrs vārds, izņemot pirmo, sākas ar lielo burtu:

## Piemēri:

`mansDzimsanasGads`

## Pascal Case

Katrs vārds sākas ar lielo burtu:

`MansDzimsanasGads`

## Snake Case

Katru vārdu atdala pasvītrojuma rakstzīme:

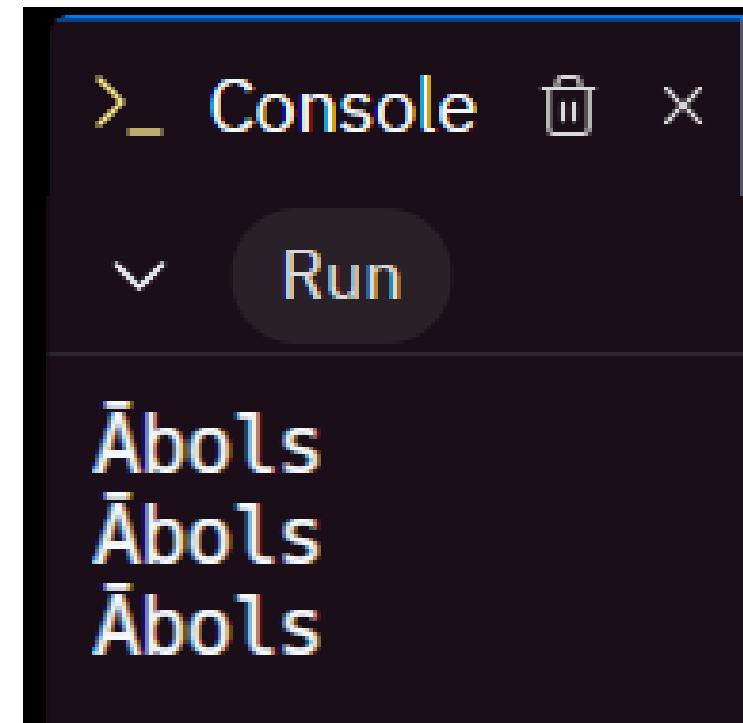
`mans_dzimsanas_gads`

---

# VIENA VĒRTĪBA VAIRĀKIEM MAINĪGAJIEM

Vienu un to pašu vērtību var piešķirt vairākiem mainīgajiem vienā rindā:

```
a = b = c = "Āboļs"  
print(a)  
print(b)  
print(c)
```

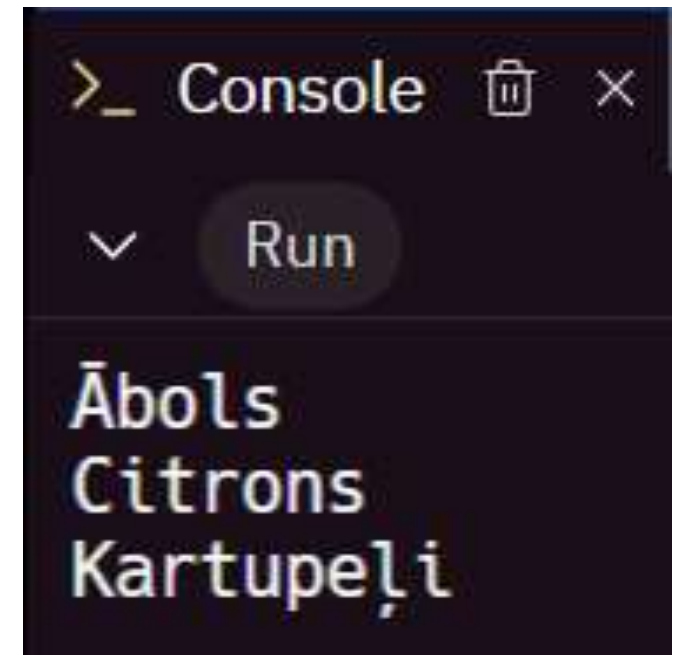


---

# DAUDZAS VĒRTĪBAS VAIRĀKIEM MAINĪGAJIEM

Python ļauj piešķirt vērtības vairākiem  
mainīgajiem vienā rindā:

```
1 a, b, c = "Ābols", "Citrons", "Kartupeļi"  
2 print(a)  
3 print(b)  
4 print(c)
```





---

# DATU TIPI

**Teksta tips:** str

**Set tips:** set, frozenset

**Ciparu tips:** int, float, complex

**Būla tips:** bool

**Secības tips:** list, tuple, range

**None tips:** NoneType

**Binārie tips:** bytes, bytearray, memoryview

**Kartēšanas tips:** dict

Tālāk pastāstīšu sīkāk par populārākajiem datu tiptiem

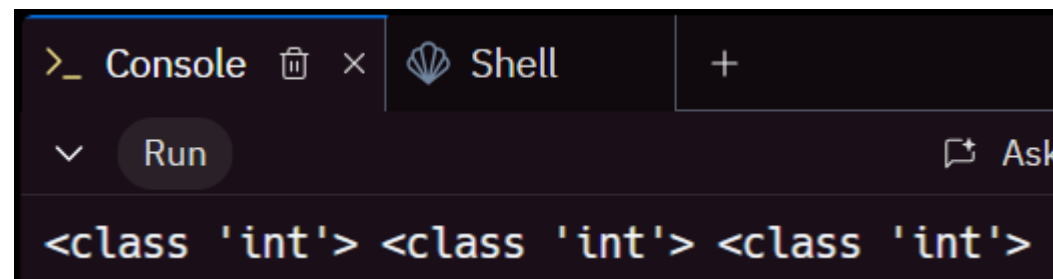
---

---

# INT – VESELS SKAITLIS

Veseli skaitļi ir nulle, pozitīvi un negatīvi skaitļi bez decimāldaļas.

```
1 x, y, z = -1, 0, 1
2 print(type(x), type(y), type(z))
```



The screenshot shows a Python console window with a dark theme. The title bar includes 'Console', a trash icon, a close icon, and a 'Shell' tab. Below the title bar, there is a 'Run' button and an 'Ask' button. The output of the code is displayed as three instances of '<class 'int'>' separated by spaces.

```
>_ Console [trash] [close] Shell +
  Run Ask
<class 'int'> <class 'int'> <class 'int'>
```

Lietošanas piemērs:

```
#Programma lūdz lietotājam ievadīt
#viņa dzimšanas gadu un saglabā šo
#veselo skaitli mainīgajā gads
gads = int(input("Ievadi savu dzimšanas gadu: "))
```

---

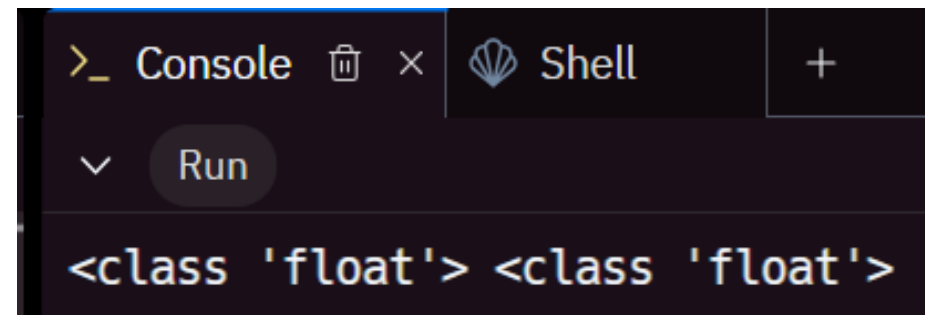
# FLOAT – REĀLS SKAITLIS

Reāli skaitļi ir pozitīvs vai negatīvs skaitlis, kas satur decimāldaļu.

```
1 x, y = -1.789, 5.6
2 print(type(x), type(y))
```

Lietošanas piemērs:

```
1 #Programma pieprasa garumu un
2 #saglabā to kā reālu skaitli
3 garums = float(input("Ievadi garumu: "))
```



The screenshot shows a code editor interface with a 'Console' tab and a 'Shell' icon. Below the 'Run' button, the output of the first code block is displayed: `<class 'float'> <class 'float'>`.

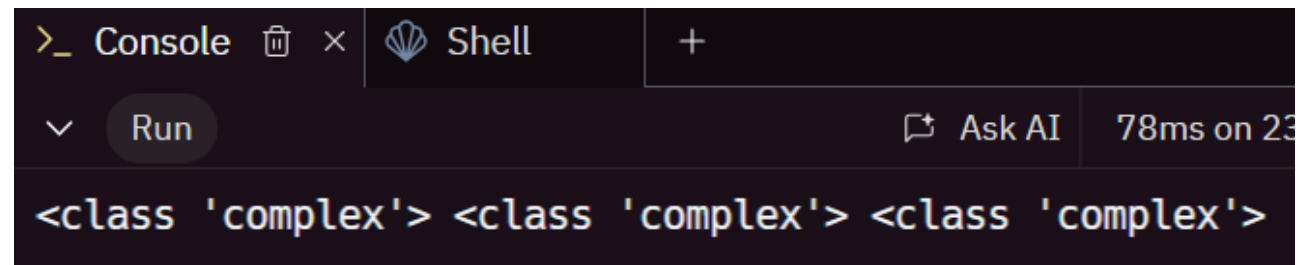
! Svarīgi atcerēties, ka izmantot komatu nedrīkst, tikai punktu!

---

# COMPLEX – SALIKTS SKAITLIS

Salikti skaitļi ir rakstīti ar “j” kā iedomāto daļu.

```
1 x, y, z = 3+5j, 5j, -5j
2 print(type(x), type(y), type(z))
```



```
>_ Console × Shell +
  Run Ask AI 78ms on 23
<class 'complex'> <class 'complex'> <class 'complex'>
```

Lietošanas piemērs:

Kompleksos skaitļus izmanto  
fizikā un matemātikā

---

# OPERATORI

+	saskaitīšana	$Z = X + Y$	$Z = 5 + 2$	$Z = 7$
-	atņemšana	$Z = X - Y$	$Z = 5 - 2$	$Z = 3$
*	reizinājums	$Z = X * Y$	$Z = 5 * 2$	$Z = 10$
/	dalīšana	$Z = X / Y$	$Z = 5 / 2$	$Z = 2.5$
%	atlikums	$Z = X \% Y$	$Z = 5 \% 2$	$Z = 1$
**	celt pakāpē	$Z = X ** Y$	$Z = 5 ** 2$	$Z = 25$
//	sadalīt bez atlikuma	$Z = X // Y$	$Z = 5 // 2$	$Z = 2$

---

# BOOL

Boļeāni attēlo vienu no divām vērtībām: True vai False.

```
1 print(2 > 1)
2 print(2 == 1)
3 print(2 < 1)
```

```
▼ Run
True
False
False
```

! Svarīgi atcerēties,  
ka Python valodā  
True un False obligāti  
rakstāma ar lielo burtu!

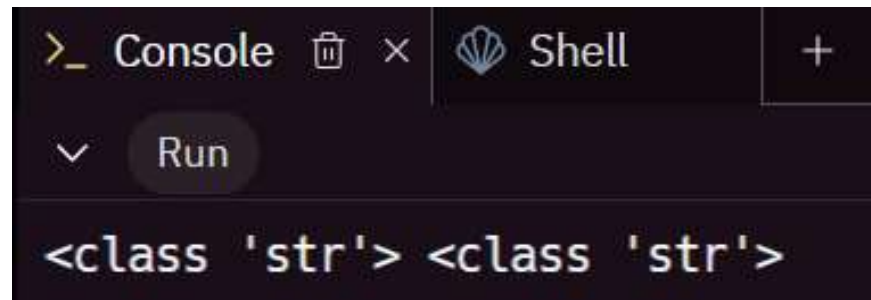
True tiek izmantots, ja darbība ir patiesa, False, kad nav.

---

# STR – TEKSTA VIRKNE

String ir teksts, kas Python ir vai nu vienkāršā, vai dubultpēdīnās.

```
1 x, y = "Sveiki", 'Labdien'
2 print(type(x), type(y))
```

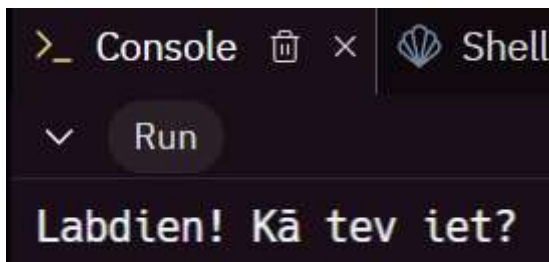


The screenshot shows a Python console window with a dark background. The title bar includes a prompt character, the word "Console", a trash icon, a close button, a shell icon, and the word "Shell". Below the title bar is a "Run" button. The output of the code is displayed as two separate lines: "<class 'str'>" followed by "<class 'str'>".

```
>_ Console [trash] [x] [shell] Shell +
  Run
<class 'str'> <class 'str'>
```

Lietošanas piemērs:

```
1 x = "Labdien!"
2 y = 'Kā tev iet?'
3 print(x, y)
```



The screenshot shows a Python console window with a dark background. The title bar includes a prompt character, the word "Console", a trash icon, a close button, a shell icon, and the word "Shell". Below the title bar is a "Run" button. The output of the code is displayed as a single line: "Labdien! Kā tev iet?".

```
>_ Console [trash] [x] [shell] Shell
  Run
Labdien! Kā tev iet?
```

!

Vienā mainīgajā nevar kombinēt dažādas pēdīņas (x = 'Sveiki')

---

# LIST – SARAKSTS

Saraksti tiek izmantoti, lai glabātu vairākus vienumus vienā mainīgajā.

```
1 saraksts = [1, 2, 3, 4]
2 print(type(saraksts))
```

Run  
<class 'list'>

Vienā sarakstā var būt dažādu tipu dati.

Un piemērs, kā vienumus var izņemt no saraksta.

```
1 saraksts = ["Gerda", 2006, True, 9.8]
2 print(saraksts[1], saraksts[-1])
```

Run  
2006 9.8



---

# TURPLE – KORTEŽS

- Kortežs ir sakārtota un nemainīga kolekcija.
- Turple pieļauj vērtību dublikātus.

```
1 kortezs = ("Gerda", 2006, True, 9.8)
2 print(type(kortezs))
```

▼ Run  
<class 'tuple'>

```
1 tuple = ("apple",)
2 tuple2 = ("apple")
3 print(type(tuple), type(tuple2))
```

▼ Run  
<class 'tuple'> <class 'str'>

Ja kortežam jābūt tikai vienai vērtībai, tad pēc tā jābūt komatam, citādi tas ir str

---

# SET – KOPA

- Kopa ir nesakārtota, nemainīga, neindeksēta kolekcija.
- Kopām nevar būt divi vienumi ar vienādu vērtību.
- Vērtības True un 1 vai False un 0 kopās tiek uzskatītas par vienu un to pašu vērtību, un tās tiek uzskatītas par dublikātiem.

```
1 kopa = {1, 2, 3, 4}
2 print(type(kopa))
```

Run

<class 'set'>

---

# DICT – VĀRDNĪCA

- Vārdnīcas tiek izmantotas, lai saglabātu datu atslēgas:vērtības pāros.
- Vārdnīca ir sakārtots, maināms un nepieļauj dublikātus.

```
1 x = {"vards": "gerda", "gads": 2004}
2 print(type(x))
```

Run

<class 'dict'>

---

# METODES

`len()` - atgriež elementu skaitu objektā.

## **str**

- `upper()` - atgriež rindas kopiju augšējā reģistrā.
- `lower()` - atgriež virknes kopiju apakšējā reģistrā.
- `capitalize()` - atgriež rindas ar lielo pirmo burtu kopiju.
- `title()` - atgriež rindas kopiju ar katra vārda lielo pirmo burtu.

## **list**

- `append(elem)` - pievieno elementu saraksta beigām.
- `insert(i, elem)` - ievieto vienumu norādītajā pozīcijā.
- `pop ([i])` - noņem un atgriež indeksu. Ja indekss nav norādīts, tiek paņemts pēdējais elements.
- `clear()` - no saraksta dzēš visus vienumus.

---

# METODES

`len()` - atgriež elementu skaitu objektā.

## **dict**

- `keys()` - atgriež visu vārdnīcas atslēgu attēlojumu.
- `values()` - atgriež visu vārdnīcas vērtību attēlojumu.
- `items()` - tiek atgriezts visu vārdnīcas pāru (atslēga, vērtība) attēlojums.

## **set**

- `add(elem)` - pievieno elementu.
- `remove(elem)` - noņem elementu.

## **turtle**

- `count(value)` - tiek atgriezts vērtības gadījumu skaits kortežā.
- `index(value, [start, [stop]])`: tiek atgriezts pirmais vērtības indekss kortežā.

---

# KONKRĒTĀ DATU TIPA IESTATĪŠANA

Ja vēlaties norādīt datu tipu, varat izmantot šādas konstruktora funkcijas:

- `x = str("Hello World")`
- `x = int(20)`
- `x = float(20.5)`
- `x = complex(1j)`
- `x = list(("apple", "banana"))`
- `x = tuple(("apple", "banana"))`
- `x = range(6)`
- `x = dict(name="John", age=36)`
- `x = set(("apple", "banana"))`
- `x = frozenset(("apple", "banana"))`
- `x = bool(5)`
- `x = bytes(5)`
- `x = bytearray(5)`
- `x = memoryview(bytes(5))`

---

# DATU TIPĀ IEGŪŠANA

Jebkura mainīga datu tipu var iegūt, izmantojot funkcijas:

- `type()`
- `isinstance()`

## Piemēri:

```
1 x = 5
2 y = isinstance(x, int)
3 print(y)
```

Konsole: True

```
1 x = 5
2 y = type(x)
3 print(y)
```

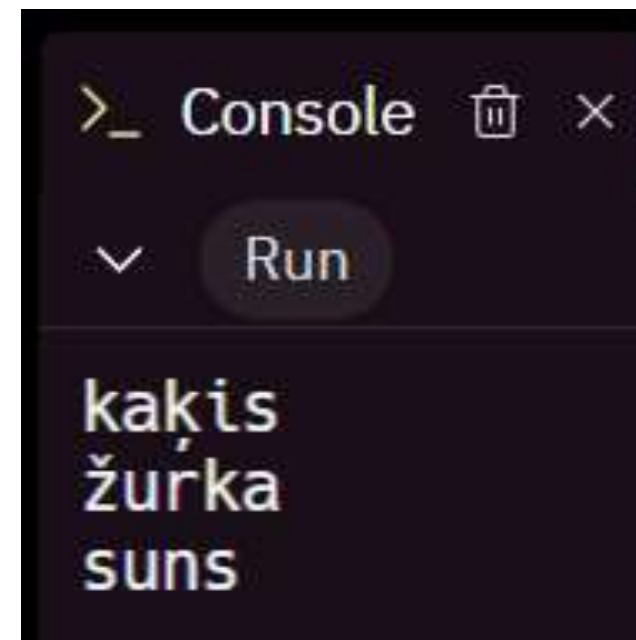
Konsole: <class 'int'>

---

# ATPAKOŠANA

Python ļauj izgūt vērtības no saraksta  
uz mainīgajiem.

```
dzīvnieki = ["kaķis", "žurka", "suns"]  
x, y, z = dzīvnieki  
print(x)  
print(y)  
print(z)
```





---

# AVOTI