
Checking our Programming Environment

Predno nadaljujemo poglejmo ali naše programsko okolje deluje.

Znotraj visual studio code ustvarimo novo datoteko. Poimenujmo jo **test.py**. Vanjo vpišimo:

```
print("hello world")
```

Pritisnimo gump **play**. Če se nam v terminal izpiše `hello world` potem vse deluje pravilno.

Kalkulator

Za začetek bomo sprogramirali preprost **Calculator**, ki omogoča seštevanje, odštevanje, deljenje, množenje 2 števil.

Uprašat, **kakšni bi bili koraki, da naredimo kalkulator?** * Rabmo prikazat rezultat,...kaj še? * Rabmo znat zračunat... kaj še?

Naš program mora:

- Od uporabnika zahtevati vnos dveh števil
- Od uporabnika zahtevati naj pove katero matematično operacijo naj izvede
- Izvesti matematično operacijo
- Izpisati rezultat.

Program bi izgledal nekako sledeče:

```
# Uporabnik naj vnese prvo številko

# Uporabnik naj vnese drugo številko

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo

# Izpiši rezultat
```

Ustvarimo datoteko **simple_calculator.py**. Znotraj te skripte bomo pisali svoj kalkulator in ga sproti nadgrajevali.

Izpis rezultata

Za začetek se bomo lotili prikaz rezultata.

Print() funkcija

Da nekaj izpišemo v terminal uporabimo print( ) funkcijo.

To je **built-in funkcija**, ki pride z inštalacijo Pythona. Nekdo drug je že napisal kodo, ki lahko nekaj izpiše v terminal. Celotno "kodo je shranil" pod ime **print**. Tako da, sedaj nam ni potrebno pisati vse kode, ki izpiše nekaj v terminal ampak lahko preprosto pokličemo **print** funkcijo.

Znotraj oklepajev vpišemo vrednost, katero želimo izpisati.

```
print("Hello world")
```

Če želimo izpisati črke, jih moramo dati v navednice (")

In [1]:
```
print("Hello world")
```

```
Hello world
```

In [2]:
```
print(10)
```

```
10
```

In [3]:
```
print(-2)
```

```
-2
```

Če želimo izpisat več stvari hkrati jih ločimo z vejico.

In [4]:
```
print("Hello World", 10, 2, "Rezultat")
```

```
Hello World 10 2 Rezultat
```

> **Naloga:** S pomočjo print() funkcije izpišite `Danes je lep dan.`.

In [5]:
```
print("Danes je lep dan.")
```

```
Danes je lep dan.
```

Preprosti Kalkulator

> **Naloga:** S pomočjo print() funkcije naj se vam v terminal izpiše: Prva številka: 2 Druga številka: 3 2 + 3 = 5

In [6]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko

# Uporabnik naj vnese drugo številko

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo

# Izpiši rezultat
print("Prva številka: 2")
print("Druga številka: 3")
print("2 + 3 = 5")
```

```
Prva številka: 2
Druga številka: 3
2 + 3 = 5
```

Zaenkrat pustimo direktno napisane številke in vrednosti. Kasneje bomo te številke zamenjali z dejanskimi številkami katere bo vnesel uporabnik.

Vnos števila

Za naslednji korak bomo od uporabnika zahtevali vnos števila:

Input() funkcija

S pomočjo bult-in funkcije input( ) lahko od uporabnika zahtevamo nek vnos.

Znotraj oklepajev lahko napišemo besedilo, ki uporabniku pove, kaj se od njega pričakuje.

In [8]:
```
input("Vnesi črko: ")
```

```
Vnesi črko: a
```
Out[8]:
```
'a'
```

Sedaj bi radi, da vrednost katero uporabnik vnese, izpišemo z našim `print()` stavkom.

Težava se pojavi, ker mi lahko pridobimo vrednost od uporabnika, vendar pa te vrednosti ne shranimo nikamor. In zato je ne moremo uporabiti kasneje v programu.

V pythonu lahko vrednosti shranimo v **spremenljivke**.

Spremenljivke

Spremenljivka je kot neka beseda v katero shranimo vrednost in do te vrednosti dostopamo kasneje v kodi.

```
x = 2
```

Beri: **Vrednost 2 shrani v spremenljivko z imenom x.**

Oziroma bolj natančno: **Ovrednoti kar je na desni strani enačaja in to shrani v levo stran enačaja**

Spremenljivke nam omogočajo shranjevanje vrednosti in lepšo kontrolo nad kodo.

Vizualizacija kode

In [9]:
```
x = 2
print(x)
```

```
2
```

Vizualizacija kode

In [53]:
```
x = 10
y = 22
print(x)
print(y)
```

```
10
22
```

Da sedaj shranimo številko, ki jo uporabnik vnese:

Vizualizacija kode

In [11]:
```
x = input("Vnesi črko:")
print(x)
```

```
Vnesi črko:s
s
```

V spremenljivko lahko kadarkoli shranimo novo vrednost:

Vizualizacija kode

In [12]:
```
x = input("Vnesi črko:")
print(x)

x = 2
print(x)
```

```
Vnesi črko:d
d
2
```

Pri imenu spremenljivk moramo paziti, saj so **case-sensetive**.

Vizualizacija kode

In [54]:
```
x = 1
print(x)

X = 2
print(X)
```

```
1
2
```

Prav tako spremenljivk ne moremo poimenovati s posebnimi imeni ("keywords") katere Python že uporablja (False, None,...).

In [55]:
```
False = 1
```

```
  Cell In [55], line 1
    False = 1
    ^
SyntaxError: cannot assign to False
```

V tem primeru vidimo, da je napaka kera se je zgodila **SyntaxError**. Kar pomeni, da je prišlo do napake v sintaksi / kodi katero smo napisali.

Nato je še bolj natančno poročilo o napaki: `cannot assign to False`. Pomeni, da ne moremo v besedo `False` shraniti vrednost.

Dodatno lahko vidimo, da se je napaka zgodila v **1. vrstici.**: `Cell In [56], line 1`

Da je koda lažje berljiva, tudi po tem, ko nekdo drug bere za nami, obstaja nek skupek priporočil kako naj bo koda zapisana (PEP8).

Notri piše, da nej se spremenljivke poimenuje z uporabo **snake_case** (vse je z malimi začetnicai, besede ločimo z podčrtajem)

In [15]:
```
# Narobe
MojaSpremenljivka = 2
print(MojaSpremenljivka)
```

```
# Pravilno
moja_spremenljivka = 2
print(moja_spremenljivka)
```

```
2
2
```

> **Naloga:** Od uporabnika zahtevajte naj vnese črko. To črko shrnaite v spremenljivko z imenom **c**. Vrednost spremenljivke nato izpišite.

In [17]:
```
n = input("Vnesi črko")
print(n)
```

```
Vnesi črkoa
a
```

> **Naloga:** Od uporabnika zahtevajte naj z besedo opiše kakšen dan je. Nato izpišite `Danes je` `` `dan.`. Primer: Vnesite kakšen dan je: lep Danes je lep dan.

In [20]:
```
x = input("Vnesite kakšen dan je: ")
print("Danes je", x, "dan.")
```

```
Vnesite kakšen dan je: lep
Danes je lep dan.
```

Preprosti Kalkulator

> **Naloga:** Od uporabnika zahtevajte naj vnese dve številki. Vsako shranite v svojo spremenljivko. Spremenljivki uporabite pri končnem izpisu. (Rezultat seštevanja naj vas sedaj ne moti. Pridobili in zamenjali ga bomo v nadaljevanju.) PRIMER: Vnesi prvo številko: 4 Vnesi drugo številko: 5 Prva številka: 4 Druga številka: 5 4 + 5 = ?

In [2]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")

# Uporabnik naj vnese drugo številko
y = input("Vnesi drugo številko:")

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo

# Izpiši rezultat
print("Prva številka:", x)
print("Druga številka:", y)
print(x, "+", y, "= ?")
```

```
Vnesi prvo številko:4
Vnesi drugo številko:8
Prva številka: 4
Druga številka: 8
4 + 8 = ?
```

Vizualizacija kode

Sedaj si poglejmo, kako izvedemo matematične operacije v pythonu.

Matematična operacija

- + seštevanje
- - odštevanje
- * množenje
- / deljenje
- // celoštevilsko deljenje
- ** eksponent
- % ostanek pri deljenju

In [15]:
```
x = 9
y = 4
```

In [16]:
```
x + y
```

Out[16]:
```
13
```

In [17]:
```
# še drugačen način seštevanja
# x += y
# x
```

In [18]:
```
x - y
```

Out[18]:
```
5
```

In [19]:
```
x * y
```

Out[19]:
```
36
```

In [20]:
```
x / y
```

Out[20]:
```
2.25
```

In [21]:
```
a = 6
b = 3
a / b # Pri navadnem deljenju je rezultat vedno float. Tud če je delenje brez ostanka
```

Out[21]:
```
2.0
```

In [2]:
```
x // y # 9 / 4 = 2*4 + ostanek (ta dvojka se izpiše) - MODUL
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [2], line 1
----> 1 x // y # 9 / 4 = 2*4 + ostanek (ta dvojka se izpiše) - MODUL

NameError: name 'x' is not defined
```

In [23]:
```
x ** y # na potenco
```

Out[23]:
```
6561
```

In [24]:
```
x % y # ostanek pri deljenju
```

Out[24]:
```
1
```

Potek operacij

In [25]:
```
x = 20 + 4 * 10
x # kaj se bo izpisal? 60 al 240
```

Out[25]:
```
60
```

Vsaka operacija ima določeno pomembnost.

V izrazu se prvo izvedejo operacije z najvišjo pomembnostjo. Ko pridobimo te rezultate, se nato izvedejo naslednje najpomembnejše operacije in tako do konca.

V primeru operacij z enako pomembnostjo se le te izvajajo od leve-proti-desni.

Tabela (od najpomembnejše do najmanj)

Operacije	Opis
**	exponentiation
+x, -x, ~x	unary positive, unary negation, bitwise negation
*, /, //, %	multiplication, division, floor division, modulo
+, -	addition, subtraction
<<, >>	bit shifts
&	bitwise AND
^	bitwise XOR
|	bitwise OR
==, !=, <, <=, >, >=, is, is not	comparisons, identity
not	Boolean NOT
and	Boolean AND
or	Boolean OR

Potek operacij se lahko spremeni z uporabo oklepajev ().

Izrazi v okepajih se izvedejo pred izrazi, ki niso v oklepajih.

Nič ni narobe s pretirano uporabo oklepajev tudi, če niso potrebni. Uporaba oklepajev velja za dobro prakso, saj izboljša berljivost kode.

In [26]:
```
x = 20 + (4 * 10) # prvo se izvede oklepaj in dobimo 20 + 40 = 60
y = (20 + 4) * 10 # prvo se izvede oklepaj in dobimo 24 * 10 = 240
print(x)
print(y)
```

```
60
240
```

Vaja

Naloga: V spremenljivko **k** shranite seštevek 10 in 5. Nato to spremenljivko izpišite. V spremenljivko **l** shranite rezultat deljenja 4 in 9. Nato to spremenljivko izpišite. V spremenljivko **m** shranite rezultat množenja 3 in 7. Nato to spremenljivko izpišite. V spremenljivko **n** shranite rezultat odštevanja 3 in 5. Nato to spremenljivko izpišite. V spremenljivko **o** shranite rezultat potenciranja 3 na 4. Nato to spremenljivko izpišite. V sremenljivko **p** shranite razultat odštevanja

12 in 7. Ta rezultat nato potencirajte z dvema tretjinama. Output: 15 0.4444444444444444 21 -2 81 2.924017738212866

In [3]:
```
k = 10 + 5
print(k)

l = 4 / 9
print(l)

m = 3 * 7
print(m)

n = 3 - 5
print(n)

o = 3 ** 4
print(o)

p = (12 - 7)**(2/3)
print(p)
```

```
15
0.4444444444444444
21
-2
81
2.924017738212866
```

Uporabimo sedaj to pri našem kalkulatorju.

Za začetek privzemimo da bo uporabnik vedno hotel sešteti števili.

Preprosti Kalkulator

Kalkulatorju bomo sedaj dodali novo spremenljivko **rezultat**. V to spremenljivko bomo shranili vsoto dveh števil.

In [27]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")

# Uporabnik naj vnese drugo številko
y = input("Vnesi drugo številko:")

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo
rezultat = x + y

# Izpiši rezultat
print("Prva številka:", x)
print("Druga številka:", y)
print(x, "+", y, "=", rezultat)
```

```
Vnesi prvo številko:4
Vnesi drugo številko:5
Prva številka: 4
Druga številka: 5
4 + 5 = 45
```

Vidimo, da se nam pojavi problem.

Številki ni seštelo ampak se jih je preprosto združilo.

Do problema pride, ker so naše številke **napačnega data tipa**.

Data Types

V Pythonu so spremenljivke **dinamične**. To pomeni, da nam ni potrebno izrecno povedati računalniku kakšnega tipa je spremenljivka.

V pythonu poznamo več data tipov:

- `int` - celo število
- `float` - decimalno število
- `str` - niz črk / stavek
- ...

Da preverimo katerega data tipa je naša spremenljivka lahko uporabimo funkcijo type( ). Spremenljivko katere data tip želimo preveriti vpišemo znotraj oklepajev.

```
type(var)
```

In [28]:
```
x = 1
print(type(x))
```

```
<class 'int'>
```

Osnovni data tipi so:

Integer (celo število) - int

V Python3 ni maximalne velikosti integerja. Številka je lahko velika kolikor želimo. Omejeni smo samo z našim pomnilnikom.

In [29]:
```
x = 5
print(type(x))
print(x)
```

```
<class 'int'>
5
```

In [30]:
```
x = 2**100 # v Javi je omejitev men se zdi da 2^31 -1
print(type(x))
print(x)
```

```
<class 'int'>
1267650600228229401496703205376
```

Floating-point (decimalno število) - float

Float predstavlja decimalno ševilo (število s plavajočo vecijo).

Treba je pazit saj te številke niso popolnoma natančne ampak le aproksimacije (te aproksimacije se vidjo šele pri n-ti decimalki).

> Almost all platforms represent Python float values as 64-bit “double-precision” values, according to the IEEE 754 standard. In that case, the maximum value a floating-point

> number can have is approximately 1.8 × 10308. Python will indicate a number greater than that by the string inf:
>
> The closest a nonzero number can be to zero is approximately 5.0 × 10-324. Anything closer to zero than that is effectively zero:
>
> Floating point numbers are represented internally as binary (base-2) fractions. Most decimal fractions cannot be represented exactly as binary fractions, so in most cases the internal representation of a floating-point number is an approximation of the actual value. In practice, the difference between the actual value and the represented value is very small and should not usually cause significant problems.

In [32]:
```
x = 5.43
print(type(x))
print(x)
```

```
<class 'float'>
5.43
```

Complex numbers (kompleksna števila) - complex

Nam predstavlja kompleksna števila. Števila, ki so sestavljena iz realnega in imaginarnega dela.

In [33]:
```
x = 2 + 3j
print(type(x))
print(x)
```

```
<class 'complex'>
(2+3j)
```

String (stavek) - str

Stringi so zaporedja črk. Začnejo in končajo se z dvojnim (") ali enojnim (') narekovajem.

Vsebuje lahko neomejeno število črk. Edina omejitev je naš pomnilnik.

Lahko je tudi prazen stavek.

In [34]:
```
x = "Stavek" # navaden string z dvojnim narekovajem ""
print(type(x))
print(x)
```

```
<class 'str'>
Stavek
```

In [35]:
```
x = 'String' # navaden string z enojnim narekovajem ''
print(type(x))
print(x)
```

```
<class 'str'>
String
```

In [36]:
```
x = "" # prazen string
print(type(x))
print(x)
```

```
<class 'str'>
```

In [37]:
```
x = "100"
print(type(x))
print(x)
```

```
<class 'str'>
100
```

Če želimo v našem stringu uporabiti narekovaje naredimo to tako:

In [38]:
```
x = "String with (')"
y = 'String with (")'
print(x)
print(y)
```

```
String with (')
String with (")
```

Če pred črko vstavimo backslash (\) s tem zaobidemo kako Python ponavadi prebere to črko.

In [39]:
```
x = "String with (\")"
print(x) # ponavadi bi python prebral drugi " kot konec stringa
x = "String \nString"
print(x) # ponavadi bi python prebral n kot n. Ampak z \ ga ne prebere tko kot ponava
```

```
String with (")
String
String
```

Obstaja tudi možnost večvrstičnega izpisa.

In [40]:
```
print('''
To je primer večvrstičnega izpisa.
Vrstica 1
Vrstica 2 ''')
```

```
To je primer večvrstičnega izpisa.
Vrstica 1
Vrstica 2
```

Boolean (True or False) - bool

Boolean spremenljivka lahko zavzeme samo 2 vrednosti. Ali True ali False.

In [31]:
```
x = True
print(x)
print(type(x))

print("-------")

x = False
print(x)
print(type(x))
```

```
True
<class 'bool'>
-------
False
<class 'bool'>
```

Tudi, če spremenljivka sama po sebi ni True ali False, se jo še vedno lahko pretvorivmo v tip bool. Tako lahko vidimo, da so naslednje vrednosti False:

- Boolean False
- numerična vrednost 0 (0, 0.0, 0+0j...)
- Empty string
- Keyword None
- Empty object (kot je prazen list, prazna terka...)

Vse ostalo je True.

Da pretvorivmo neko spremenljivko v boolean tip, uporabimo besedo:

```
bool(spemenljivka)
```

In [32]:
```
print(bool(False)) # bool(x) pretvor vrednost x v boolean (al true al false)
print(bool(0))
print(bool(""))
print(bool(None))
print(bool([]))
print("********")
print(bool(True))
print(bool(1))
print(bool("abc"))
print(bool([1,2]))
```

```
False
False
False
False
False
********
True
True
True
True
```

Na podoben način lahko spreminjamo spremenljivke v ostale tipe:

```
int(spremenljivka),
str(spremenljivka),
complex(spremenljivka)
```

Vaja

Naloga: S funkcijo print() in type() izpišite po eno spremenljivko tipa boolean, integer, float, complex in string.

In [21]:
```
a = True
b = 2
c = 3.4
d = 1 +9j
e = "neki"

print(type(a))
print(a)
print()

print(type(b))
print(b)
print()

print(type(c))
print(c)
print()

print(type(d))
print(d)
print()

print(type(e))
```

```
print(e)
print()
```

```
<class 'bool'>
True

<class 'int'>
2

<class 'float'>
3.4

<class 'complex'>
(1+9j)

<class 'str'>
neki
```

Vaja

Naloga: V neko spremenljivko shranite poljubno float vrednost. Izpišite spremenljivko in njen tip. To spremenljivko pretvorite v boolan vrednost in to vrednost shranite v novo spremenljivko. Izpišite novo spremenljivko in njen tip.

In [23]:
```
x = 1.2
print(type(x))
print(x)
print()

y = bool(x)
print(type(y))
print(y)
print()
```

```
<class 'float'>
1.2

<class 'bool'>
True
```

Preprosti Kalkulator

In [41]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")

# Uporabnik naj vnese drugo številko
y = input("Vnesi drugo številko:")

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo
rezultat = x + y

# Izpiši rezultat
print("Prva številka:", x)
print("Druga številka:", y)
print(x, "+", y, "=", rezultat)
```

```
Vnesi prvo številko:4
Vnesi drugo številko:5
Prva številka: 4
Druga številka: 5
4 + 5 = 45
```

Problem pri našem kalkulatorju je, ker nam `input` vrne vrednost data tipa `str`.

In [24]:
```
x = input("Vnesi prvo številko: ")
print(x)
print(type(x))
```

```
Vnesi prvo številko: 3
3
<class 'str'>
```

Vrednosti data tipa `str` pa se ne seštevajo matematično ampak preprosto združijo skupaj.

In [44]:
```
x = "Hello"
y = "World"
print(x, type(x))
print(y, type(y))
print(x+y)
```

```
Hello <class 'str'>
World <class 'str'>
HelloWorld
```

In [45]:
```
x = input("Vnesi prvo številko:")
print(x, type(x))

y = input("Vnesi drugo številko:")
print(y, type(y))

rezultat = x + y
print(rezultat, type(rezultat))
```

```
Vnesi prvo številko:4
4 <class 'str'>
Vnesi drugo številko:5
5 <class 'str'>
45 <class 'str'>
```

Da bi jih lahko normalno seštevali jih želimo spremeniti v tip **int** - `<class int>`.

Da spremenimo spremenljivko iz string v integer naredimo tako:

In [47]:
```
x = "3"
print(type(x), x)

x = int("3")
print(type(x), x)
```

```
<class 'str'> 3
<class 'int'> 3
```

In [6]:
```
x = input("Vnesi številko: ")
print(type(x), x)

x = int(input("Vnesi številko: "))
print(type(x), x)
```

```
Vnesi številko: 4
<class 'str'> 4
Vnesi številko: 4
<class 'int'> 4
```

In sedaj lahko normalno uporabimo matematične funkcije.

Vaja

Naloga: Uporabnika vprašajte naj vnese svojo starost v letih. Vrednost pretvorite v mesece in to izpišite.

In [56]:
```
age_str = input("Vnesi koliko let si sat: ")
age_int = int(age_str)

months = age_int*12
print(f"Star si {months} mesecev.")
```

```
Vnesi koliko let si sat: 12
Star si 144 mesecev.
```

Vizualizacija kode

Kodo lahko optimiziramo tako, da ne uporabimo dveh spremenljivko (`age_str` in `age_int`), ampak samo eno spremenljivko `age` .

In [57]:
```
age = input("Vnesi koliko let si sat: ")
age = int(age)

months = age*12
print(f"Star si {months} mesecev.")
```

```
Vnesi koliko let si sat: 12
Star si 144 mesecev.
```

Vizualizacija kode

Dodatno lahko kodo polepšamo tako, da obe vrstici združimo.

In [48]:
```
age = int(input("Vnesi koliko let si sat: "))

months = age*12
print(f"Star si {months} mesecev.")
```

```
Vnesi koliko let si sat: 20
Star si 240 mesecev.
```

Vizualizacija kode

Preprosti Kalkulator

Naloga: Kalkulatorju sedaj dodate seštevanje dveh števil, katere je vnesel uporabnik.

In [7]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede
```

```
# Izvedi to operacijo
rezultat = x + y

# Izpiši rezultat
print("Prva številka:", x)
print("Druga številka:", y)
print(x, "+", y, "=", rezultat)
```

```
Vnesi prvo številko:4
Vnesi drugo številko:5
Prva številka: 4
Druga številka: 5
4 + 5 = 9
```

Vizualizacija kode

Sedaj bomo še malo polepšali izpis številk in rezultata.

Dynamic string creating and string formatting

[vir](https://realpython.com/python-f-strings/)

Znotraj pythona lahko stringe dinamično ustvarjamo.

Namesto, da jih dobesedno napišemo, lahko uporabimo vrednosti katere smo shranili v spremenljivke.

V kalkulatorjo želimo, da se nam končna vrstica spreminja glede na to kaj je uporabnik vnesel.

S prihodom Python3.6 verzije se stringe dinamično ustvarja s pomočjo f-string

```
f'Besedilo {spremenljivka1:format1}, besedilo
naprej{spremenljivka2:format2}, besedilo naprej....'
```

Dokumentacija f-string

In [25]:
```
ime = "Anže"
starost = 10

moj_izpis = f'{ime} je {starost} let star'
print(moj_izpis)
print(type(moj_izpis))
```

```
Anže je 10 let star
<class 'str'>
```

Sedaj smo dinamično ustvarili nek string.

Vrednosti katere dinamično vstavljamo v naš string lahko tudi izpišemo na specifični način. Decimalne vrednosti lahko izpišemo na 5 decimalk natančno, številke lahko izpišemo z predznakom ali brez, vrednosti lahko centriramo, itd.

In [26]:
```
ime = "Anže"
starost = 10

moj_izpis = f'{ime} je {starost:.5f} let star'
```

```
print(moj_izpis)
print(type(moj_izpis))
```

```
Anže je 10.00000 let star
<class 'str'>
```

In [27]:
```
ime = "Anže"
starost = 10

moj_izpis = f'{ime} je {starost:e} let star'
print(moj_izpis)
print(type(moj_izpis))
```

```
Anže je 1.000000e+01 let star
<class 'str'>
```

In [28]:
```
ime = "Anže"
starost = 10

moj_izpis = f'{ime:^10} je {starost} let star'
print(moj_izpis)
print(type(moj_izpis))
```

```
   Anže    je 10 let star
<class 'str'>
```

Še primer kako lahko oblikujemo naš format izpisa.

In [29]:
```
ime = "Anže"
starost = 10

moj_izpis = f'{ime:-^10} je {starost:*>10.3f} let star, oziroma {starost*12:e} mesece
print(moj_izpis)
# {ime:-^10} ime -> spremenljivka, "-" -> znak s katerim zapolni mesta, "^" -> naj bo
# {starost:*>10.3f} starost -> ime spremenljivke, "*" -> znak s katerim zapolni mesta
# {starost*12:e} starost*12 -> spremenljivka ki jo želimo izpisat, "e" -> naj bo stva
```

```
---Anže--- je ****10.000 let star, oziroma 1.200000e+02 mesecev.
```

Pred tem, s prihdom Python2.6, se je uporabljalo

```
str.format()
```

In [30]:
```
ime = "Anže"
starost = 10

moj_izpis = "Živjo {}. Star si {} let.".format(ime, starost)
print(moj_izpis)
```

```
Živjo Anže. Star si 10 let.
```

.format() je počasnejši od f' ' stavka

Še pred tem se je uporabljalo

```
%-formating
```

In [31]:
```
name = "Eric"
age = 74

moj_izpis = "Hello, %s. You are %s." % (name, age)
print(moj_izpis)
```

```
Hello, Eric. You are 74.
```

Ta način je najpočasnejši. Pri veliki količini spremenljivk hitro postane nepregleden.

Vaja

Naloga: Podana imate dva podatka o zemlji. Uporabite ju, da dobite sledeč izpis: radij = 6371.0 # km age = 4_543_000_000 # years Specifikacije zemlje: Povprečni radij: 6371.00 km. Starost zemlje: 4.54e+09 let.

In [45]:
```
radij = 6371.0 # km
age = 4_543_000_000 # years

print(f"Specifikacije zemlje:")
print(f"Povprečni radij: {radij:.2f} km.")
print(f"Starost zemlje: {age:.2e} let.")
```

```
Specifikacije zemlje:
Povprečni radij: 6371.00 km.
Starost zemlje: 4.54e+09 let.
```

Preprost Kalkulator

Naloga: Konče zadeve kalkulatorja izpišite na sledeč način s pomočjo **f-strings**: Prva številka: 57 Druga številka: 4 57 + 4 = 61.00

In [33]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede

# Izvedi to operacijo
rezultat = x + y

# Izpiši rezultat
print(f"Prva številka:\t {x}")
print(f"Druga številka:\t {y}")
print(f"{x} + {y} = {rezultat:.2f}")
```

```
Vnesi prvo številko:57
Vnesi drugo številko:4
Prva številka:   57
Druga številka:  4
57 + 4 = 61.00
```

Sedaj gremo na zadnji korak.

Ta korak bomo izvedli tako, da bomo od uporabnika zahtevali naj vnese znak operacije katero hoče, da se izvede. Možni znaki so:

- **+** - seštevanje
- **-** - odštevanje
- ***** - množenje
- **** - deljenje

Nato bomo preverili kateri znak je uporabnik vnesel:

- če je vnesel znak **+** bomo števili **sešteli**,
- če je vnesel znak **-** bomo števili **odšteli**,
- če je vnesel znak ***** bomo števili **zmnožili**,
- če je vnesel znak **** bomo števili **deljili**

To logiko odločanja bomo dodali s pomočjo `IF` stavka.

If statement

```
if <expr>:
    <if-block statement>
    <if-block statement>
    <if-block statement>
    ...
<following_statement>
```

`<expr>` je nek izraz, ki nam vrne `bool` vrednost (ali `True`, ali `False`).

- Če je `<expr>` enak `True`, potem se bo izvedel **if blok kode**. To so vse vrstice, ki so zamaknjene desno. Ko so vse te vrstice izvede, se program nadaljuje z izvajanje `<following_statement>` in naprej.
- Če je `<expr>` enak `False`, potem se **if blok kode** preskoči in se program nadaljuje z `<following_statement>`.

Vizualizacija kode

In [58]:
```
print("Začetek programa")
if True:
    print("Znotraj if")
print("Nadaljevanje programa")
```

```
Začetek programa
Znotraj if
Nadaljevanje programa
```

In [59]:
```
print("Začetek programa")
if False:
    print("Znotraj if")
print("Nadaljevanje programa")
```

```
Začetek programa
Nadaljevanje programa
```

Indentation / Zamikanje

Pri Pythonu se zamikanje (indentation) uporablja za definiranje blokov kode. Vse vrstice z istim zamikom se smatrajo kot isti blok kode.

Zamikanje je določeno z tabulatorjem ali presledki. Ni važno točno število, važno je, da je skozi kodo enako.

In [4]:
```
if True:
    print("Znotraj if. Vrstica 1")
    print("Znotraj if. Vrstica 2")
    print("Znotraj if. Vrstica 3")
print("Nadaljevanje programa")
```

```
Znotraj if. Vrstica 1
Znotraj if. Vrstica 2
Znotraj if. Vrstica 3
Nadaljevanje programa
```

In [5]:
```
if True:
 print("Znotraj if. Vrstica 1")
 print("Znotraj if. Vrstica 2")
 print("Znotraj if. Vrstica 3")
print("Nadaljevanje programa")
```

```
Znotraj if. Vrstica 1
Znotraj if. Vrstica 2
Znotraj if. Vrstica 3
Nadaljevanje programa
```

In [6]:
```
if True:
 print("Znotraj if. Vrstica 1")
 print("Znotraj if. Vrstica 2")
  print("Znotraj if. Vrstica 3")
print("Nadaljevanje programa")
```

```
  Cell In [6], line 4
    print("Znotraj if. Vrstica 3")
    ^
IndentationError: unexpected indent
```

Vidimo, da dobimo napako `IndentationError`. Dodatno nam python pove, kjer on misli, da je prišlo do napake: `Cell In [6], line 4`.

Trenutno je naša `True / False` vrednost nespremenljiva. Mi pa želimo odločitve delati glede na to kaj je uporabnik vnesel.

Mi želimo primerjat vrednost katero je vnesel uporabnik, z neko drugo vrednostjo.

Primerjalne operacije

S pomočjo primerjalnih operacij python primerja dve vrednosti in nam nato vrne `bool` vrednost (`True` oziroma `False`)

- `a < b` -> Python nam vrne vrednost `True`, če je `a` manjši od `b`. Če je `a` večji od `b` nam vrne `False`.
- `a > b` -> Python nam vrne vrednost `True`, če je `a` večji od `b`. Če je `a` manjši od `b` nam vrne `False`.
- `a <= b` -> Python nam vrne vrednost `True`, če je `a` manjši ali enak `b`. Če je `a` večji od `b` nam vrne `False`.
- `a >= b` -> Python nam vrne vrednost `True`, če je `a` večji ali enak `b`. Če je `a` manjši od `b` nam vrne `False`.
- `a == b` -> Python nam vrne vrednost `True`, če je vrednost spremenljivke `a` enaka vrednosti spremenljivke `b`. Če `a` ni enak `b` nam vrne `False`.
- `a != b` -> Python nam vrne vrednost `True`, če vrednost spremenljivke `a` ni enaka vrednosti spremenljivke `b`. Če je `a` enak `b` nam vrne `False`.

In [14]:
```
x = 5 < 10
print(type(x), x)
```

```
<class 'bool'> True
```

In [15]:
```
x = 10 > 5
print(type(x), x)
```

```
<class 'bool'> True
```

In [16]:
```
x = 3 <= 2
print(type(x), x)
```

```
<class 'bool'> False
```

In [17]:
```
x = 5 >= 5
print(type(x), x)
```

```
<class 'bool'> True
```

In [20]:
```
x = 5 == 4
print(type(x), x)
```

```
<class 'bool'> False
```

In [23]:
```
# treba pazit pri primerjanju float vrednosti, ker na prvi decimalki je stvar še enak
x = 1.1000 + 2.2000
y = 3.3000
z = x == y
print("X in Y gledana na 3 decimalke natancno:")
print(f' x: {x:.3} \n y: {y:.3}')
print(type(z), z)

print()
print("X in Y gledana na 50 decimalk nacantno:")
print(f' x: {x:.50} \n y: {y:.50}')
```

```
X in Y gledana na 3 decimalke natancno:
 x: 3.3
 y: 3.3
<class 'bool'> False

X in Y gledana na 50 decimalk nacantno:
 x: 3.30000000000000026645352591003756970167160034179690
 y: 3.29999999999999982236431605997495353221893310546880
```

In [24]:
```
x = 4 != 4
print(type(x), x)
```

```
<class 'bool'> False
```

Se pravi lahko primerjamo dve spremenljivki na sledeč nain:

Vizualizacija kode

In [60]:
```
temperatura = 10
meja = 15
expr = temperatura < meja
print(expr)

if expr:
    print("Vklopi ogrevanje")

print("Nadaljevanje programa")
```

```
True
Vklopi ogrevanje
Nadaljevanje programa
```

Našo kodo lahko polepšamo tako, da združimo vrstice:

In [61]:
```
temperatura = 10
meja = 15

if temperatura < meja:
    print("Vklopi ogrevanje")

print("Nadaljevanje programa")
```

```
Vklopi ogrevanje
Nadaljevanje programa
```

Vaja

Naloga: Uporabnika vprašajte za dve decimalni vrednosti. Če je prva vrednost večja od druge, izpišite `Prvo število je večje.`. V nasprotnem primeru ne naredite ničesar.

In [50]:
```
x = float(input("first: "))
y = float(input("second: "))

if x > y:
    print("Prvo število je večje.")
```

```
first: 12
second: 12.3
```

Else and elif

Včasih želimo, da če je nek pogoj izpolnjen, se izvede dolčen del kode. V primeru, da pogoj ni izpolnjen pa naj se zgodi drug del kode.

To dosežemo z `else`.

```
if <expr>:
    <if-blok statement(s)>
else:
    <else-blok statement(s)>

<following statement>
```

Če je `<expr>` `True` se izvede blok direktno pod njem, če pa je `<expr>` `False` se ta blok kode preskoči in se izvede blok pod `else`.

Vizualizacija kode - Za `soncno_vreme`. probamo in `True` in `False`.

In [62]:
```
soncno_vreme = True

if soncno_vreme:
    print('Odpri okna')
else:
    print('Zapri okna')

print("Nadaljevanje programa")
```

```
Odpri okna
Nadaljevanje programa
```

Če želimo še večjo razvejanost naših možnosti lahko uporabimo `elif` (else if).

```
if <expr>:
    <if-blok statement(s)>
elif <expr>:
    <elif-blok statement(s)>
elif <expr>:
    <elif-blok statement(s)>
else:
    <else-blok statement(s)>
```

Python preveri vsak `<expr>` posebej. Pri ta prvem, ki bo `True`, bo izvedel njegov blok kode. Če ni nobeden `True` se bo izvedel `else` blok kode.

Vizualizacija kode - Probamo mal menjat vrednost `x`.

In [29]:
```
x = 20
if x > 100:
    print('x je večje od 100')
elif x > 50:
    print('x večje od 50 in manjše od 100')
elif x > 30:
    print('x večje od 30 in manjše od 50')
elif x > 10:
    print('x večje od 10 in manjše od 30')
else:
    print("x manjše od 10")

print("End")
```

```
x večje od 10 in manjše od 30
End
```

One-line if statement

Vir: https://realpython.com/python-conditional-statements/

Obstaja način zapisa if stavka v eni vrstici ampak se ta način odsvetuje, ker napravi kodo nepregledno.

```
z = 1 + x if x > y else y + 2
```

In [30]:
```
x = 8
z = 1 + x if x > 10 else x**2
print(z)

x = 20
z = 1 + x if x > 10 else x**2
print(z)
```

```
64
21
```

Vaja

Naloga: Napišite program, ki bo uporabnika vprašal naj vnese neko celoštevilsko vrednost. Program naj nato izpiše ali je vrednost deljiva s 3 in ne.

In [27]:
```
x = int(input("Vnesi celoštevilsko vrednost: "))
if x%3 == 0:
    print("Število je deljivo s 3")
```

```
else:
    print("Število ni deljivo s 3")
```

```
Vnesi celoštevilsko vrednost: 3
Število je deljivo s 3
```

Vaja

Naloga: Napišite program, ki bo pretvoril stopinje Celzija v Fahrenheit ali obratno. Uporabnik naj vnese številko. Nato naj vnese v katerih enotah nam je podal vrednost (**C** ali **F**). Glede na vnešeno črko naj vaš program uporabi pravilno formulo za pretvorbo. ``` T(°F) = T(°C) × 9/5 + 32 T(°C) = (T(°F) - 32) x 5/9 ``` Če uporabnik ni vnesel **C** ali **F** naj program izpiše *Prišlo je do napake.* Primer: ``` Vnesi vrednost: 12 Vnesi enoto: C ``` Rešitev: ``` 12 stopinj celzija je enako 53.6 fahrenheit. ```

In [24]:
```
stopinje = float(input("Vnesi vrednost: "))
enota = input("V katerih enotah je podana vrednost? [C/F]: ")

if enota == "C":
    fahrenheit = stopinje*9/5 + 32
    print(f"{stopinje} {enota} je enako {fahrenheit} fahrenheit.")
elif enota == "F":
    celsius = (stopinje - 32)*5/9
    print(f"{stopinje} {enota} je enako {celsius} celsius.")
else:
    print("Prišlo je do napake")
```

```
Vnesi vrednost: -50
V katerih enotah je podana vrednost? [C/F]: C
-50.0 C je enako -58.0 fahrenheit.
```

Vizualizacija kode

Preprost kalkulator

Naloga: Od uporabnika zahtevajte naj vnese enega izmed štirih znakov: `+, -, *` ali `/`. Če je uporabnik vnesel: + naj se števili seštejeta - naj se števili odštejeta * naj se števili zmnožita / naj se števili zdelita

Vizualizacija kode

In [35]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede
operacija = input("Katero operacijo želimo izvesti? [+, -, *, /]: ")

# Izvedi to operacijo
if operacija == "+":
    rezultat = x + y
```

```
elif operacija == "-":
    rezultat = x - y
elif operacija == "*":
    rezultat = x * y
elif operacija == "/":
    rezultat = x / y


# Izpiši rezultat
print(f"Prva številka: {x}")
print(f"Druga številka: {y}")
print(f"{x} {operacija} {y} = {rezultat}")
```

```
Vnesi prvo številko:4
Vnesi drugo številko:3
Katero operacijo želimo izvesti? [+, -, *, /]: *
Prva številka: 4
Druga številka: 3
4 * 3 = 12
```

Problem se pojavi, če za drugo številko vnesemo `0` in števili želimo deljiti (`/`).

In [36]:
```
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede
operacija = input("Katero operacijo želimo izvesti? [+, -, *, /]: ")

# Izvedi to operacijo
if operacija == "+":
    rezultat = x + y
elif operacija == "-":
    rezultat = x - y
elif operacija == "*":
    rezultat = x * y
elif operacija == "/":
    rezultat = x / y


# Izpiši rezultat
print(f"Prva številka: {x}")
print(f"Druga številka: {y}")
print(f"{x} {operacija} {y} = {rezultat}")
```

```
Vnesi prvo številko:4
Vnesi drugo številko:0
Katero operacijo želimo izvesti? [+, -, *, /]: /
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In [36], line 21
     19     rezultat = x * y
     20 elif operacija == "/":
---> 21     rezultat = x / y
     24 # Izpiši rezultat
     25 print(f"Prva številka: {x}")

ZeroDivisionError: division by zero
```

Da se znebimo te napake lahko dodamo še en `if` .

Preprost Kalkulator

Naloga: Dodajte še en `if` stavek, ki v primeru deljenja preveri, da ni druga številka enaka 0. * Če je druga številka enaka 0, naj se izpiše **"Operacija ni mogoča."**, v `rezultat` pa naj se shrani string **"Ne obstaja"** * Če druga številka ni enaka 0, naj se števili zdeljita. Primer: ```python Vnesi prvo številko:4 Vnesi drugo številko:5 Katero operacijo želimo izvesti? [+, -, *, /]: / Prva številka: 4 Druga številka: 5 4 / 5 = 0.8 Vnesi prvo številko:2 Vnesi drugo številko:0 Katero operacijo želimo izvesti? [+, -, *, /]: / Operacija ni mogoča. Prva številka: 2 Druga številka: 0 2 / 0 = Ne obstaja ```

Vizualizacija kode

- Primer 1: `x = 3`, `y = 0`, `operacija = /`
- Primer 2: `x = 3`, `y = 0`, `operacija = *`
- Primer 3: `x = 3`, `y = 2`, `operacija = /`
- Primer 4: `x = 3`, `y = 2`, `operacija = *`

In [40]:
```python
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede
operacija = input("Katero operacijo želimo izvesti? [+, -, *, /]: ")

# Izvedi to operacijo
if operacija == "+":
    rezultat = x + y
elif operacija == "-":
    rezultat = x - y
elif operacija == "*":
    rezultat = x * y
elif operacija == "/":
    if y == 0:
        print("Operacija ni mogoča.")
        rezultat = "Ne obstaja"
    else:
        rezultat = x / y


# Izpiši rezultat
print(f"Prva številka: {x}")
print(f"Druga številka: {y}")
print(f"{x} {operacija} {y} = {rezultat}")
```

```
Vnesi prvo številko:2
Vnesi drugo številko:0
Katero operacijo želimo izvesti? [+, -, *, /]: /
Operacija ni mogoča.
Prva številka: 2
Druga številka: 0
2 / 0 = Ne obstaja
```

Kodo lahko polepšamo tako, da združimo oba `if` stavka. To lahko dosežemo tako, da združimo oba `if <expresions>` v enega.

Se pravi, `bool` vrednosti katere dobimo od `if <expr>` bomo združili med seboj.

`Bool` vrednosti lahko združujemo z **logičnimi operacijami**.

Logične operacije

Poznamo različne logične operacije. Vrednosti, katere vrnejo lahko predstavimo v tabeli:

NOT - Negacija

`not` operacija obrne `bool` vrednost.

A	NOT
False	True
True	False

In [41]:
```
x = False
y = not x
print(y) # obrne vrednost. Če je vrednost True jo obrne v False, če je False jo obrne
```

```
True
```

Vizualizacija kode

In [65]:
```
ogrevanje = False
expr = not ogrevanje
print(type(expr), expr)

if expr:
    print("Vključi ogrevanje")
print("Nadaljevanje programa")
```

```
<class 'bool'> True
Vključi ogrevanje
Nadaljevanje programa
```

OR

Če je vsaj ena od vrednosti enaka `True`, bo izhod `or` operacije enah `True`.

A	B	OR
False	False	False
False	True	True
True	False	True
True	True	True

In [43]:
```
x = True
y = False
z = x or y
print(z) # če je ena izmed vrednosti True, bo izraz True
```

```
True
```

Vizualizacija kode

In [44]:
```
dezuje = False
night_time = True
expr = dezuje or night_time
print(type(expr), expr)
```

```
if expr:
    print("Zapri okno")
print("Nadaljevanje programa")
```

```
Zapri okno
Nadaljevanje programa
```

AND

Če je vsaj ena od vredonsti enaka `False`, bo izhod `False`.

A	B	AND
False	False	False
False	True	False
True	False	Fasle
True	True	True

In [45]:
```
x = True
y = False
z = x and y
print(z) # če je ena izmed vrednosti False, bo izraz False
```

```
False
```

Vizualizacija kode

In [66]:
```
temperatura = 11
ogrevanje_vkljuceno = False
expr1 = temperatura < 15
expr2 = not ogrevanje_vkljuceno
expr = expr1 and expr2
print(type(expr), expr)

if expr:
    print("Vključi ogrevanje")
print("Nadaljevanje programa")
```

```
<class 'bool'> True
Vključi ogrevanje
Nadaljevanje programa
```

Vaje

Naloga: Uporabnika vprašajte za 3 celoštevilske vrednosti in jih izpišite s pomočjo print() in type(). V eni vrstici preverite ali je druga vrednost enaka prvi in ali je tretja vrednost manjša ali enaka prvi. 1. št.: 2 2. št.: 4 3. št.: 5 Tip: , Vrednost: 2 Tip: , Vrednost: 4 Tip: , Vrednost: 5 False

In [52]:
```
a = int(input("1. št.: "))
b = int(input("2. št.: "))
c = int(input("3. št.: "))

print(f"Tip: {type(a)}, Vrednost: {a}")
print(f"Tip: {type(b)}, Vrednost: {b}")
print(f"Tip: {type(c)}, Vrednost: {c}")

print((b == a) and (c <= a))
```

```
1. št.: 2
2. št.: 4
3. št.: 5
Tip: <class 'int'>, Vrednost: 2
Tip: <class 'int'>, Vrednost: 4
Tip: <class 'int'>, Vrednost: 5
False
```

In [ ]:

Preprosti Kalkulator

Naloga: Združite oba `if` stavka tako, da, če je druga številka enaka 0 in, če je znak enak `/`, naj se izpiše **"Operacija ni dovoljena."** in rezultat naj bo enak **"Ne obstaja."** Primer: ```python Vnesi prvo številko:3 Vnesi drugo številko:4 Katero operacijo želimo izvesti? [+, -, *, /]: / Prva številka: 3 Druga številka: 4 3 / 4 = 0.75 Vnesi prvo številko:6 Vnesi drugo številko:0 Katero operacijo želimo izvesti? [+, -, *, /]: / Operacija ni mogoča. Prva številka: 6 Druga številka: 0 6 / 0 = Ne obstaja Vnesi prvo številko:4 Vnesi drugo številko:0 Katero operacijo želimo izvesti? [+, -, *, /]: + Prva številka: 4 Druga številka: 0 4 + 0 = 4 ```

Vizualizacija kode

In [55]:
```python
# Preprosti Kalkulator

# Uporabnik naj vnese prvo številko
x = input("Vnesi prvo številko:")
x = int(x)

# Uporabnik naj vnese drugo številko
y = int(input("Vnesi drugo številko:"))

# Uporabnik naj pove katera matematična operacija naj se izvede
operacija = input("Katero operacijo želimo izvesti? [+, -, *, /]: ")

# Izvedi to operacijo
if operacija == "+":
    rezultat = x + y
elif operacija == "-":
    rezultat = x - y
elif operacija == "*":
    rezultat = x * y
elif operacija == "/" and not(y == 0):
    rezultat = x / y
else:
    print("Operacija ni mogoča.")
    rezultat = "Ne obstaja"



# Izpiši rezultat
print(f"Prva številka: {x}")
print(f"Druga številka: {y}")
print(f"{x} {operacija} {y} = {rezultat}")
```

```
Vnesi prvo številko:4
Vnesi drugo številko:0
Katero operacijo želimo izvesti? [+, -, *, /]: +
Prva številka: 4
Druga številka: 0
4 + 0 = 4
```

In [ ]: