summary_code

October 18, 2023

1 Kintamieji ir sąlyginiai sakiniai

1.1 Kintamieji

Programavime kintamaieji naudojami saugoti informaciją (pvz.: name, age, height, friends, is_adult, and person). Kintamasis sukuriamas tą momentą, kai jam priskiriama pirmojo reikšmė. Reikšmes galima parašius jo vardą, lygybės ženklą ir reikšmę:

[2]:
```
# Numerical variable assignments
a = 10
b = 20.5
```

Šiomis dviem programinio kodo eilutėmis, buvo sukurti du kintamieji *a* ir *b* ir buvo priskirtos reikšmės *10* ir *20,5* atitinkamai. Nuo šio momentu, šie kintamieji gali būti panaudoti skaičiuojant kitas reikšmes, modifikuojant patį kintąmąjį ar atliekant veiksmus.

Pavyzdžiui galime atlikti pagrindinius aritmetinius veiksmus, o šių veiksmų rezultatus priskirti naujiems kintamiesiems:

[3]:
```
# Basic arithmetic operations
sum_ab = a + b # sum
diff_ab = a - b # subtraction
prod_ab = a * b # multiplication
quot_ab = a / b # division
mod_ab = a % b # modulo (remainder after division)
```

[4]:
```
# Displaying results
print("Sum of a and b: " + str(sum_ab))
print("Difference when b is subtracted from a: " + str(diff_ab))
print("Product of a and b: " + str(prod_ab))
print("Quotient when a is divided by b: " + str(quot_ab))
print("Remainder when a is divided by b: " + str(mod_ab))
```

```
Sum of a and b: 30.5
Difference when b is subtracted from a: -10.5
Product of a and b: 205.0
Quotient when a is divided by b: 0.4878048780487805
Remainder when a is divided by b: 10.0
```

1.1.1 Kodo paaiškinimas

"# tekstas" - reiškia komentarą. Šis komentaras skirtas žmonėms skaitantiems kodą. Komentarai nedaro įtakos programos vykdymui.

"print(argumentas)" - *print* raktožodis reiškia funkcija, kuri atvaizduoja ekrane pateiktą reikšmę (argumentą).

Kabutėse pateikta reikšmės yra eilutės tipo reikšmė (galima tai laikyti, kaip tekstinį tipą). Funkcijoje *print* pateiktos eilutės reikšmės yra sujungiamos su "+" ženklus ir kita reikšme/kintamuoju. Pavyzdžiui *print("Sum of a and b:" + str(sum_ab))* eilutės vykdymo metu iškviečiama *print()* funkcija (ji yra kažkur *python* viduje kažkieno kito suprogramuota), ši funkcija išveda į ekraną visą argumentą (*"Sum of a and b:" + str(sum_ab)*). Šiame argumente yra reiškinys - eilutės reikšmių sudėtis. Čia randama dar viena funkcija *str()*. Ši funkcija paverčia bet kokį pateiktą kintąmjį/reiškinį į eilutės tipo reikšmę. Tai turime tokį reiškinys: eilutė+eilutė. Tai galų galiausiai paimama *sum_ab* reikšmė, ji paverčia eilute (tekstu), o ši eilutė (tekstas) prijungiamas prie *"Sum of a and b:"*.

1.1.2 Kintamojo panaudojimo pavyzdys

Kintamuosius gerai panaudoti, kai jie naudojami kelis kartus kode, pavyzdžiui turime kažkokį tai matematinį reiškinį $y = (a+b) \cdot 15/(a-b)$. Šį galima parašyti, naudojantis jau anksčiau sukurtais kintamaisiais:

[5]:
```
# Using variables in mathematical expressions
y = (a + b) * 15 / (a - b)
print("Result of the expression: " + str(y))
```

```
Result of the expression: -43.57142857142857
```

1.1.3 Kintamųjų tipai

Programavime, dažnu atveju, yra skirtingų tipų kintamieji. *Python* programavimo kalboje yra tokių tipų kintamieji: - sveikieji skaičiai (integer); - dešimtainiai/slankiojo kablelio skaičiai (float); - teksto/eilutės (string); - loginės (bool).

[6]:
```
# Assigning values to variables
name = "John"  # str: a string data type that holds textual information
age = 30  # int: an integer data type that holds whole numbers
height = 5.9  # float: a floating-point data type that holds numbers with␣
 ↪decimal points
is_raining = True # bool: a boolean data type that holds either true or either␣
 ↪false
```

Sužinoti kintąmojo tipą galima sužinoti panaudojus funkciją *type(arg)*, kuri gražina eilutės reikšmę, kuri parodo, kokio tipo yra kintamasis:

[7]:
```
print(type(name))
print(type(age))
print(type(height))
```

```
print(type(is_raining))
print(type(a))
print(type(sum_ab))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'int'>
<class 'float'>
```

Atkreipkite dėmesį kintamasis *a* yra sveikojo skaičiaus tipo (žiūrėti į priešpaskutinę eilutę), o jau atliktus operaciją (sudėtį) su kitu skaičiu (slankiojo kablelio, *angl. float*) ir priskyrus kintąmajam, šis į gavo slankiojo kablelio tipą (žiūrėti į paskutinę eilutę).

1.1.4 Kintamųjų vardų rašymo taisyklės

- Kintamojo vardas **turi** prasidėti su raide arba apatiniu brūkšniu ("-");
- Kintamjo varads **negali** prasidėti skaičiumi;
- Kintamjo varde **gali būti tik** raidžių-skaičių (alpha-numeric) ženklai ir apatiniai ženklai (A-z, 0-9, and _);
- Kintamojo varduose **paistomi** simbolių dydžiai (mažosios ir didžiosios raidės) (case-sensitive) - tai reiškia, kad "A" nėra lygu "a";
- Kintamojo vardas *negali* būti Python raktožodis.

[8]:
```
myname = "Vilius"
my_name = "Vilius"
_my_name = "Vilius"
myName = "Vilius"
MYNAME = "Vilius"
myname2 = "Vilius"
```

Nors ir *Python* turi savo apribojimu, bet programuotojų bendruomenėje yra sutartos taisyklės, kaip geriausia rašyti kintamuosius: naudokite mažąją raidę, žodį ar žodžius mažosiomis raidėmis. Kad būtų lengviau skaityti, žodžius atskirkite apatiniu brūkšniu (pagal PEP8). Toks rašymo būdas vadinasi *snake_case*.

[9]:
```
# Bad cases
fiRsTNAME = "Vilius"
FIRSTCASE = "Vilius"
firstcase = "Vilius"
f_i_r_s_t_c_a_s_e = "Vilius"

# Good case
first_name = "Vilius"
```

1.2 Sąlyginiai sakiniai

Prieš tai užrašytas kodas buvo vykdomas nuosekliai, po vieną eilutę, iš viršaus į apačią, nepraleidžiant nei vienos, išskyrus komentarus. Veiksmų eigai šakoti gali būti panaudoti sąlyginiai sakiniai. Sąlygas naudojime ir kasdieniniame gyvenime: jei bus X, tai darysiu Y arba darysiu Y, jeigu bus X. Pavyzdžiui: - jei dega raudona šviesoforo signalas: sustosiu. - jei dega geltonas šviesoforo signalas: sulėtinsiu greitį. - jei dega žalias šviesoforo signalas: važiuosiu.

1.2.1 Palyginimo operatoriai

Prieš išsiaiškinant, kaip rašomi sąlyginiai sakiniai programavime, svarbu žinoti loginius ir palyginimo operatorius, kurie dažniausiai naudojami užrašant tuos sąlyginius sakinius. Pats operatorius yra ženklas arba simbolis, nurodantis skaičiavimo, kurį reikia vykdyti reiškinyje, tipą. Loginiai operatoriai išrašyti žemiau:

Operatorius	Pavadinimas	Pavyzdys
==	Ar lygu	x == y
!=	Ar nelygu	x != y
>	Ar daugiau už	x > y
<	Ar mažiau už	x < y
>=	Daugiau už arba lygu	x >= y
<=	Mažiau už arba lygu	x <= y

Naudojantis šiuos operatorius ir du kintamuosius/reikšmes gaunama *True* arba *False* reikšmė. Taip galima patikrinti, ar du skaičiai lygūs, ar žodis ilgesnis už kitą, ar temperatūra šiandienos didesnė nei praėjusiais metais ir t.t.

[10]:
```
# a and b variables are defined above
# a is defined as 10
# b is defined as 20.5
# Comparison operations
is_equal = a == b
is_not_equal = a != b
is_greater = a > b
is_less = a < b
is_greater_equal = a >= b
is_less_equal = a <= b

# Displaying comparison results
print("Is a equal to b? " + str(is_equal))
print("Is a not equal to b? " + str(is_not_equal))
print("Is a greater than b? " + str(is_greater))
print("Is a less than b? " + str(is_less))
print("Is a greater than or equal to b? " + str(is_greater_equal))
print("Is a less than or equal to b? " + str(is_less_equal))
```

```
Is a equal to b? False
Is a not equal to b? True
```

```
Is a greater than b? False
Is a less than b? True
Is a greater than or equal to b? False
Is a less than or equal to b? True
```

1.2.2 Loginiai operatoriai

Kad būtų galima jungti šių operacijų rezultatus, pavyzdžiui, ar temperatūra šiandienos didesnė nei praėjusiais metais ir šiandien lijo, bet nelijo vakar, tai naudojami loginiai operatoriai:

Operatorius	Pavadinimas	Pavyzdys
and	Grąžina True jeigu abu jei abu teiginiai (operandai) yra teisingi.	$x < 5$ and $x < 10$
or	Grąžina True jeigu vienas iš teiginių (operandų) yra teisingas.	$x < 5$ or $x < 4$
not	Apverčia rezultatą, grąžina False, jeigu rezultatas true	$x > y$

Operacijų su šiais operandais rezultatais taip pat yra loginis (*True* arba *False*).

[11]:
```
# Defining new variables
x = 15
y = 10

# a and b variables are defined above
# a is defined as 10
# b is defined as 20.5
# Logical and comparison operations
is_equal = a == b and X == y
is_ab_equal_or_xy_not_equal = a == b and X != y
is_ab_not_equal_and_xy_not_equal = a != b and x != y

# Displaying comparison results
print("Is a equal to b AND is x equal to y? " + str(is_equal))
print("Is a a equal to b OR is x not equal to y? " +␣
 ↪str(is_ab_equal_or_xy_not_equal))
print("Is a a not equal to b AND is x not equal to y? " +␣
 ↪str(is_ab_not_equal_and_xy_not_equal))
```

```
Is a equal to b AND is x equal to y? False
Is a a equal to b OR is x not equal to y? False
Is a a not equal to b AND is x not equal to y? True
```

1.2.3 Sąlyginių sakinių sintaksė

Sąlyginiai sakiniai (kai norime atlikti kodą pagal tai ar sąlyga yra tiesa) užrašomi naudojant raktažodį *if*. Pavyzdys:

[12]:
```
# a and b is defined earlier
if a > b:
    print("a is greater than b")
if a < b:
    print("a is lesser than b")
```

```
a is lesser than b
```

Kaip matote, po sąlygos raktažodžio *if* rašomas reiškinys, kurio rezultatas yra *True* arba *False*. Šie sakiniai dažniausiai galima būti skaitomi paprastai, kaip įprastas tekstas. Šiuo atveju galima skaityti taip:

1. jeigu a yra daugiau už b, tai atspausdint “a is greater than b”;
2. jeigu a yra mažiau už b, tai atspausdink “a is lesser than b”.

Svarbu žinoti, kad po sąlygos visos eilutės atitrauktos nuo dešinės pusės (žiūrėkite į kodą), bus įvykdytos. Pavyzdžiui:

[13]:
```
# a and b is defined earlier
if a > b:
    print("1. a is greater than b")
    print("2. a is greater than b")
    print("3. a is greater than b")
    print("4. a is greater than b")
    print("5. a is greater than b")
    print("6. a is greater than b")
    print("7. a is greater than b")
    print("8. a is greater than b")
    print("9. a is greater than b")
if a < b:
    print("1. a is lesser than b")
    print("2. a is lesser than b")
    print("3. a is lesser than b")
    print("4. a is lesser than b")
    print("5. a is lesser than b")
    print("6. a is lesser than b")
    print("7. a is lesser than b")
```

```
1. a is lesser than b
2. a is lesser than b
3. a is lesser than b
4. a is lesser than b
5. a is lesser than b
6. a is lesser than b
7. a is lesser than b
```

Taip galima įterpti net ir kitus sąlygos blokus:

[14]:
```
if a < b:
    print("a is lesser than b")
    if b > a:
        print("b is greater than b")
```

```
a is lesser than b
b is greater than b
```

Norint praplėsti sąlygos sakinius, kad būtų alternatyvi sąlygą (*kitu atveju jeigu*), programavimo kalbos, įskaitant ir *Python*, turi raktažodį *elif* (kitose programavimo kalbose nurodoma panašiai - *else if*):

[17]:
```
if a > b:
    print("a is greater than b")
elif a < b:
    print("a is less than b")
else:
    print("a is equal to b")
```

```
a is less than b
```

Taip pat viršuj esančiame kode matome ir kitą raktažodį *else*, kuris veikia taip: jeigu nei viena *if* ir/ar *elif* sąlyga (kad ir kiek jų būtų) yra netiesa (reiškinys yra lygus *False*), tai vykdomos visos kodo eilutės kurios yra po *else* raktažodžiu.

1.2.4 Sąlyginiai sakiniai ir loginiai operatoriai

Sąlygas galima jungti naudojant loginius operatorius, jų kiekis neribojamas, bet reikia laikytis kodo skaitomumo:

[16]:
```
z = 20
# x and y is defined earlier, above
# Using 'and' logical operator
if x > y and x < z:
    print("x is greater than y AND x is less than z.")

# Using 'or' logical operator
if x > y or x > z:
    print("x is greater than y OR x is greater than z.")

# Using 'not' logical operator
is_x_greater_than_y = x > y
if not is_x_greater_than_y:
    print("x is NOT greater than y.")
else:
    print("x IS greater than y.")

# Combining logical operators
if x > y and (not x > z):
```

```
    print("x is greater than y AND x is NOT greater than z.")

# Using 'not' with 'or'
if not (x > y or x > z):
    print("Neither condition is true.")
else:
    print("At least one condition is true.")

# Checking multiple conditions
if x > y and x > z and x > 0:
    print("x is the largest and positive number.")
elif x > y or (z > x and z > y):
    print("Either x is greater than y OR z is the greatest amongst x and y.")
else:
    print("None of the above conditions are true.")
```

```
x is greater than y AND x is less than z.
x is greater than y OR x is greater than z.
x IS greater than y.
x is greater than y AND x is NOT greater than z.
At least one condition is true.
Either x is greater than y OR z is the greatest amongst x and y.
```

2 Išspręstos užduotys

2.1 01 užduotys | skaičiai

1. suskaičiuoti **a**, kai $a = \frac{5 \cdot 10^5 - 6 \cdot (-699)^2}{25}$

   Tokiai užduočiai išspręsti reikia žinoti aritmetinius operatorius. Vienas iš variantų juos sužinoti suvesti į paieškos variklį (pvz.: google.com) raktažodžius "python arithmetic variable". Pavyzdžiui išsirinkome Python Arithmetic Operators. Taip pat, reikia žinoti, kad kintamieji sukuriami parašius jo vardą ir priskyrus jam reikšmę su lygybės ženklu. Užduoties sprendimas:

[24]:
```
a = (5*10**5-6*(-699)**2)/25
print(a)
```

```
-97264.24
```

Reiktų nepamiršti skaičiuojant apskliausti reikalingose vietose reiškinį, kad būtų teisingai suskaičiuotas (pavyzdžiui turinti trupmeną). Aritmetiniai veiksmai atliekami tokia pat kaip ir matematikoje.

2. Ištraukti kvadratinę šaknį iš 25; 99.

   Šiai užduočiai užtenka sužinoti, kaip traukiama skaičiaus kvadratinė šaknis *Python*'e. Žiūrėtame mūsų puslapyje tokio aritmetinio operatoriaus nėra. Tai pabandžius suvesti į paieškos variklį "python square root", randame (arba How do I calculate square root in Python?, kad reikia importuoti *math* modulį (apie modulius sužinosime ateityje) ir panaudoti math.sqrt(a)

funkciją, kur a - skaičius, arba pasinaudojus matematika - pakėlus trupmena. Matematikoje, kiekvieną šaknį galime pakeisti trupmena: $\sqrt{a} = a^{\frac{1}{2}}$ arba bendru atveju $\sqrt[m]{a^n} = a^{\frac{n}{m}}$.

[25]:
```
# First number with math.sqrt
import math
print(math.sqrt(25))

# Second number with fraction
print(99**(1/2))
```

```
5.0
9.9498743710662
```

3. Suskaičiuoti **c**, kai $c = \sqrt[5]{13111}$.

   Šiai užduočiai spręsti, vėl panaudosime laipsnių savybę. Šį kartą jau rezultatą priskirsime kintamąjam:

[26]:
```
c = 13111**(1/5)
# check result
print(c)
```

```
6.660811808551842
```

4. Suskaičiuoti **b**, kai $b = \frac{b^6}{2,5}$

Naujų aritmetinių veiksmų nėra, kelti ir dalinti išsiaiškinome jau anksčiau. Dabar reikės vietoj skaičiaus panaudoti kintamąjį, kurį jau sukūrėme praeitame žingsnyje:

[27]:
```
b = (c**6)/2.5
# check variable b
print(b)
```

```
34931.9614487693
```

5. Išvesti žinutę "a is greater than or equal to b", jeigu a daugiau arba lygu už b, kitu atveju "a is lesser than b";

   Šiame žingnsnyje matome, kad veiksmus reikia atlikti esant tik tam tikrom sąlygom. Tokiu atveju naudosime *if*, *elif*, *else* sakinius:

[37]:
```
if a >= b:
    print("a is greater than or equal to b")
else:
    print("a is lesser than b")
```

```
a is lesser than b
```

Iš užduoties žodis "jeigu" programavime užrašomas su ***if***, "kitu atveju" užrašomas ***else***, "daugiau arba lygu" su $>=$.

6. Išvesti žinutę “a is lesser than c”, kai **a** mažiau **c**, kitu atveju jeigu **a** yra lygu **c**, tai išvesti “a is equal to c”, kitu atveju “a is greater than c”;

   Turime analogišką situaciją, kaip praeitame žingsnyje, tik čia prisideda dar vienas sąlygos sakinys (“kitu atveju jeigu”), kurį užrašysime su ***elif***:

[39]:
```
if a < b:
    print("a is lesser than c")
elif a == c:
    print("a is equal to c")
else:
    print("a is greater than c")
```

```
a is lesser than c
```

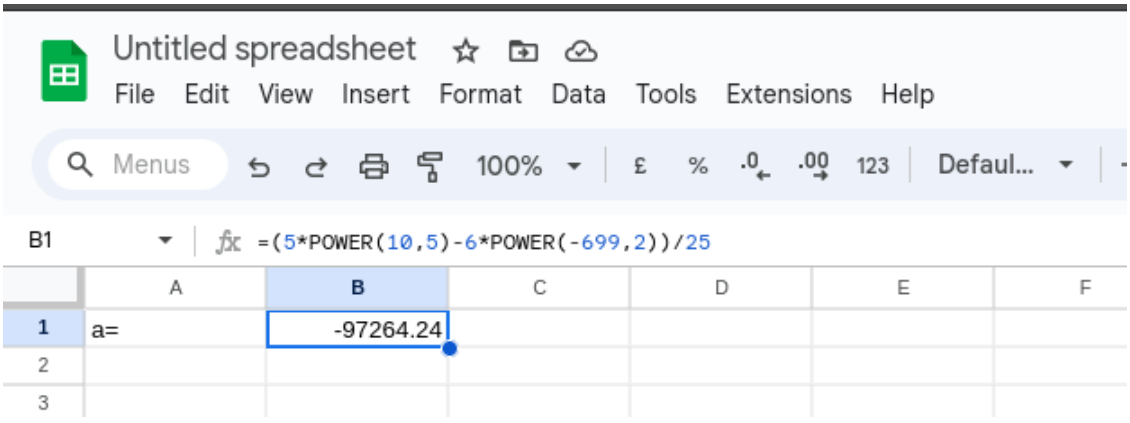
Matome, kad pirmos sąlygos reikšmei esant *True* - “**a** mažiau **b**” yra tiesa (žiūrėti į pirmą salygą po *if*), įvykdomi visi sakiniai po ta sąlyga ir kodas toliau nebevykdomas.

2.2 02 Užduotys | skaičiai | alternatyva | skaičiuoklė

Žodis skaičuoklė gali skambėti keistai, angliškai tai yra *spreadsheet*. Prieš tai buvusioms užduotis išspręsti panaudosime skaičiuoklės programą. Programą galime pasirinkti iš keletos alternatyvų: *Microsoft Excel*, *Google Sheets*, *LibreOffice Calc*. Ši užduotis bus sprendžiama su skaičiuokle tam, kad būtų parodyta langeliai, kaip analogija kintamiesiems. Naudosiu *Google Sheets*, bet lygiai tas pats veiks ir su kitomis programomis.

1. suskaičiuoti **a**, kai $a = \frac{5 \cdot 10^5 - 6 \cdot (-699)^2}{25}$

   Kad langeliui būtų priskirta reikšmė, reikia pradžioje rašyti “=” (panašiai, kaip su *python*, tik dar reikia prieš lygybę parašyti kintamojo vardą). Tada kelimas laipsniu aprašomas su *POWER(a,b)* funkcija, visi kiti aritmentiniai operatoriai yra tokie pat. Į **B1** langelį įrašiau “=(5*POWER(10,5)-6*POWER(-699,2))/25” ir gaunu:



2. Ištraukti kvadratinę šaknį iš 25; 99.

   Turime analogišką situaciją, kaip ir su *Python*, šaknies traukimo operatoriaus nėra. Yra funkcija *sqrt(num)* arba galima pasinaudoti matematikos magija pakėlus skaičių trupmena.

B2 | fx =SQRT(25)

	A	B
1	a=	-97264.24
2	25 šaknis su sqrt	5
3	99 šaknis pakėlus trupmena	9.949874371

B3 | fx =POWER(99, 1/2)

	A	B
1	a=	-97264.24
2	25 šaknis su sqrt	5
3	99 šaknis pakėlus trupmena	9.949874371

3. Suskaičiuoti **c**, kai $c = \sqrt[5]{13111}$.

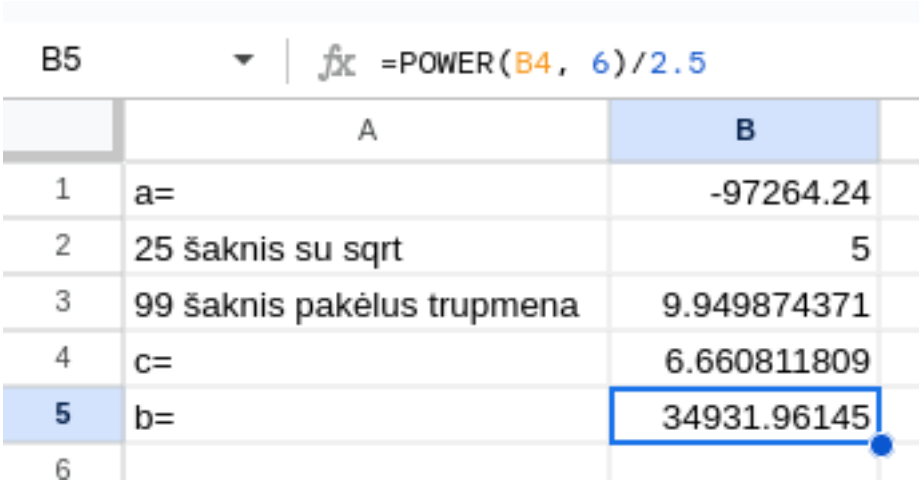
   Kadangi n-tojo laipsnio šaknies ištraukti neina su specializuota funkcija (kaip *sqrt(num)*), tai kelsime trupmeniniu laipsniu:

B4 | fx =POWER(13111,1/5)

	A	B
1	a=	-97264.24
2	25 šaknis su sqrt	5
3	99 šaknis pakėlus trupmena	9.949874371
4	c=	6.660811809

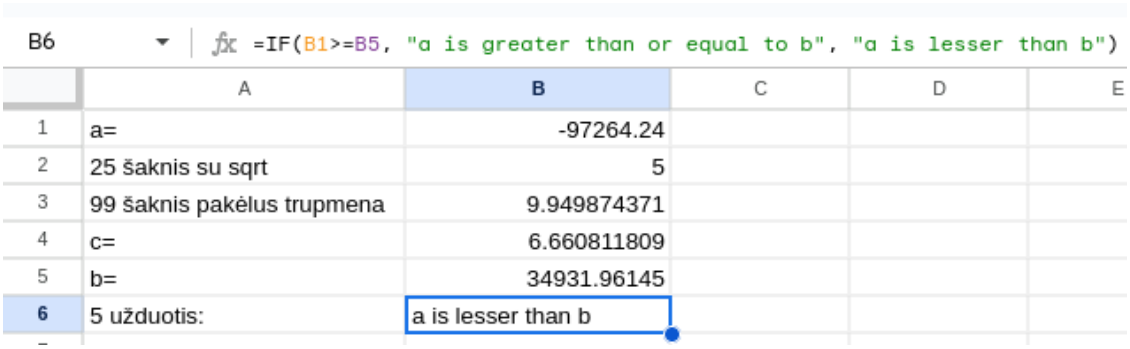
4. Suskaičiuoti **b**, kai $b = \frac{b^6}{2,5}$

   Python kalboje, kaip skaitiklį, įrašėme kintamąjį, o skaičiuoklėje įrašyme langelio koordinates:



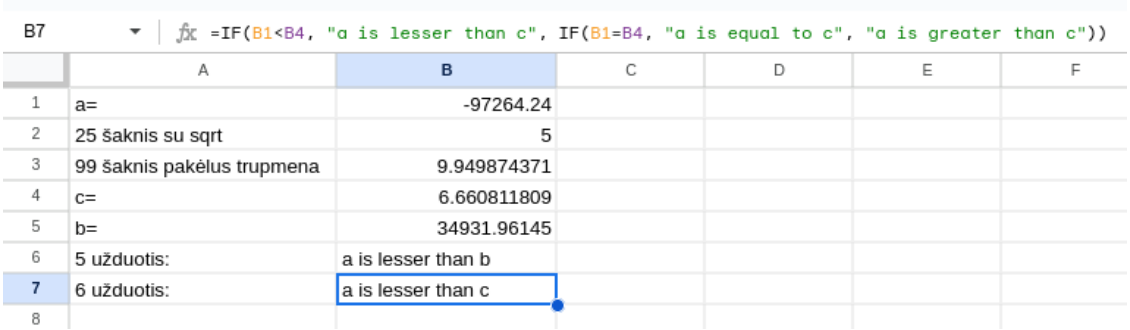
5. Išvesti žinutę “a is greater than or equal to b”, jeigu a daugiau arba lygu už b, kitu atveju “a is lesser than b”;

   Šią žinutę pavaizduosime langelyje su *IF* sintakse pasirinke atitinkamus langelius (*python*’e rašėme kintamuosius):



6. Išvesti žinutę “a is lesser than c”, kai **a** mažiau **c**, kitu atveju jeigu **a** yra lygu **c**, tai išvesti “a is equal to c”, kitu atveju “a is greater than c”;

   Kadangi skaičiuoklėse nėra *elif*, tai įrašysime vietoje *IF* trečiojo argumento kitą *IF*, taip sukursime atitikmenį *elif* programavime:

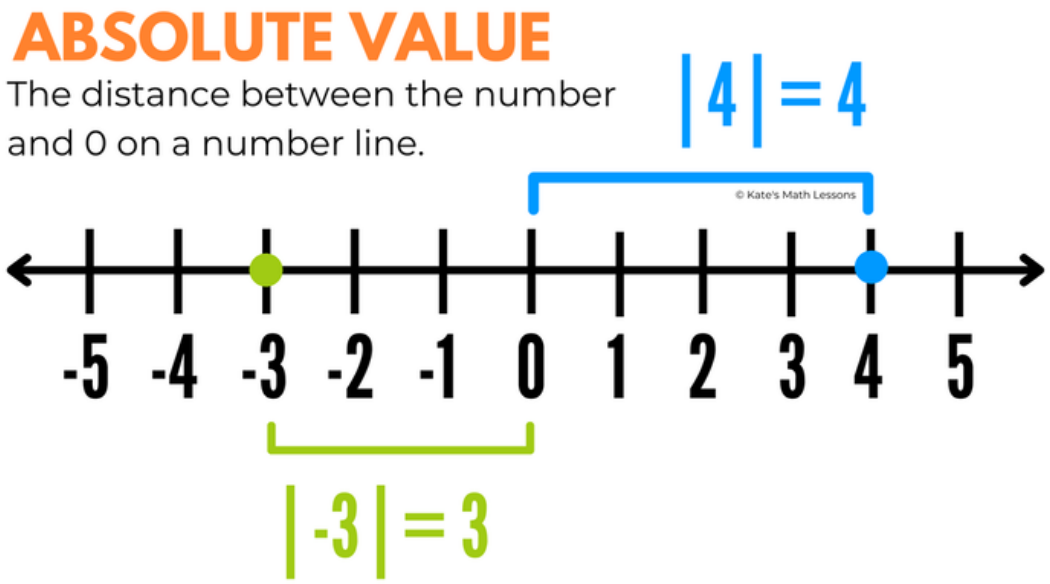


   Reikia atkreipti dėmesį, kai rašome “ar lygu”, skirtingai nei programavime, skaičiuoklė tai užsirašo vienu lygybės ženklu “=” (programavime “ar lygu” užsirašo dviems lygybės ženklais

"==").

2.3 02 Užduotys | modulis

Modulis yra skaičiaus vertė be skaičių lydinčio ženklo. Tarkim, modulis skaičių 4 ir -4 yra vienoda ir lygi 4. Modulis dar kitaip vadinamas absoliučia skaičiaus vertė. Dar galima skaičiaus modulį įsivaizduoti, kaip skaičiaus atstumą iki nulio:



Matematikoje tai yra funkcija dalimis (angl. **piecewise function**):

$$|x| = f(x) = \begin{cases} x & \text{jeigu } x \geq 0 \\ -x & \text{jeigu } x < 0 \end{cases}$$

Ši funkcija reiškia, kad jeigu modulyje esantis skaičius yra teigiamas, modulio rezultatas yra lygus tam skaičiui (žiūrėti pirmą formulės eilutę), jeigu modulyje esantis skaičius neigiamas ($x > 0$) tai rezultate prie skaičiaus esančio modulyje pridedamas minusas (arba kitaip padarome jį neigiamu). Kadangi tas skaičius jau ir taip neigiamas, tai *du minusai -> pliusas*.

Aiškinant modulį, keletą buvo pavartotas žodis **jeigu**, tai reiškia sąlygą, o tai reiškia, kad galima tai užrašyti programavimo kalba.

Pagal užduotį sprendimą reikės pasitikrinti su 3 skaičias: -10, 0, 100. Tai galima sukurti tris kintamuosius:

[7]:
```
x1 = -10
x2 = 0
x3 = 100

# check variables
print("x1:")
```

```
print(x1)
print("--------")

print("x2:")
print(x2)
print("--------")

print("x3:")
print(x3)
print("--------")
```

```
x1:
-10
--------
x2:
0
--------
x3:
100
--------
```

Dabar jau galima perrašyti modulio funkciją kiekvienam skaičiui. Ateityje bus išmokta naudotis ciklais ir funkcijomis, su kuriais kodo perrašymo skaičius mažės, o perpanaudojimas didės.

[17]:
```
print("|x1| = ")
if x1 >= 0:
    mod_x1 = x1
# In this case "else" can also be used instead of elif, it will work the same
elif x1 < 0:
    mod_x1 = -x1
print(mod_x1)

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

print("|x2| = ")
if x2 >= 0:
    mod_x2 = x2
elif x2 < 0:
    mod_x2 = -x2

print(mod_x2)

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

print("|x3| = ")
if x3 >= 0:
    mod_x3 = x3
elif x3 < 0:
```

```
    mod_x3 = x3

print(mod_x3)
```

```
|x1| =
10
--------
|x2| =
0
--------
|x3| =
100
```

Kaip matome, pagal rezultatus, kodas parašytas teisingai.

Tokios, dažnai naudojamas, funkcijos būna sukurtos programavimo kalboje ir jos vadinasi integruotomis (*angl. built-in functions*) arba standartinės bibliotekos funkcijomis (*angl. standard library functions*). Kad surastume modulio funkciją *Python* kalbome, galima paieškoti informacijos suvedus "absolute value python" (nors anglų kalbjoje šie žodžiai skiriasi, bet kitur dar galima surasti "modulus", bet nesumaišykite su "modulo", kas yra visiškai kitas dalykas). Randame, kad absoliuti skaičiaus reikšmė gali būti surasta su:

- syntax: abs(number)
- number: Integer, floating-point number, complex number.
- return: Returns the absolute value.

Toks parašymas reiškia, kad reikia naudoti funkciją *abs*, į kurią (naudojant skliaustus) įrašyme kažkokį tai skaičių, tas skaičius (žiūrėti "number") gali būti sveikasis, slankiojo kablelio, kompleksinis. Funkcijos reikšmė arba reikšmė, kurią gražina funkcija bus tos pateikto skaičiaus modulis (arba absoliuti reikšmė). Išbandome šią funkciją su *x1*, *x2*, *x3* skaičiais:

[19]:
```
# Syntax: abs(number)
# number: Integer, floating-point number, complex number.
# Return: Returns the absolute value.
print("|x1| = ")
print(abs(x1))

print("--------")
print("|x2| = ")
print(abs(x2))

print("--------")
print("|x3| = ")
print(abs(x3))
```

```
|x1| =
10
--------
|x2| =
0
```

```
--------
|x3| =
100
```

Gautos reikšmės tokios pat, sumažintas eilučių skaičius, taip galima savo kodo rašymą padaryti efektyvesniu.

3 Daugiau informacijos

- https://realpython.com/python-variables/
- https://www.w3schools.com/python/gloss_python_creating_variables.asp
- https://realpython.com/python-conditional-statements/
- https://www.geeksforgeeks.org/python-if-else/
- https://www.geeksforgeeks.org/abs-in-python/