FLOW CONTROL STATEMENTS

Omogočajo nam kontrolo sprememb in logike programa.

If statement

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

```
<expr> je izraz ovrednoten v Boolean kontekstu.
<statement> je Python izraz (nadaljevanje naše kode), ki je pravilno zamakn
jen.
Če je <expr> True, potem se izvedejo <statement>. Če je <expr> False, potem
se  <statement> preskoči in se ne izvede.
Nato se program nadaljuje z <following_statement>
```

Indentation / Zamikanje

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

> Bloke kode se lahko poljubno globoko "nesta".

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

In [5]:

```
x = 0
y = 5

if x < y:
    print("Smo znotraj if.")
    print("End if")
print("End")
```

```
Smo znotraj if.
End if
End
```

Else

Včasih želimo, da če je nekaj res se izvede določen blok kode, če stvar ni res pa naj se izvede drug del kode.

To dosežemo z else.

```
if <expr>:
    <statement(s)>
else:
    <statement(s)>
<following_statement>
```

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

In [4]:

```
x = 100

if x < 50:
    print('(first block)')
    print('x is small')
else:
    print('(second block)')
    print('x is large')

print("End")
```

```
(second block)
x is large
End
```

Elif

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

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

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

In [5]:

```
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

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

```
<expr1> if <conditional_expr> else <expr2>

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

If <conditional_expr> is true, <expr1> is returned and <expr2> is not evalu
ated.
If <conditional_expr> is false, <expr2> is returned and <expr1> is not eval
uated.
```

In [6]:

```
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
```

The pass statements

Uporablja se kot "placeholder", da nam interpreter ne meče napak.

In [11]:

```
if True:

print("Hello") # should give IndentationError
```

```
  File "<ipython-input-11-33a91c099307>", line 3
    print("Hello") # should give IndentationError
        ^
IndentationError: expected an indented block
```

In [12]:

```
if True:
    pass
print("Hello") # should be fine now with the pass added
```

```
Hello
```

Vaja 01

Napišite program, ki bo uporabnika uprašal naj vnese neko celoštevilsko vrednost. Program naj nato izpiše ali je vrednost deljiva z 3 ali ne.

In [2]:

```
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: 1
Število ni deljivo s 3
```

Vaja 02

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.

$$T(°F) = T(°C) \times 9/5 + 32$$

$$T(°C) = (T(°F) - 32) \times 5/9$$

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.
```

While

While zanka deluje na podoben princip kot if. While izvaja blok kode, dokler je "expression" True.

```
while <expr>:
    <statement(s)>
```

In [26]:

```
lepo_vreme = True
while lepo_vreme:
  print('Vreme je lepo.')
  lepo_vreme = False
```

```
Vreme je lepo.
```

In [14]:

```
#the body should be able to change the condition’s value, because if the condition
#True at the beginning, the body might run continuously to infinity
#while True:
#    print("Neskončna zanka. Se ne ustavim.")

#ustavimo v CTRL + C
```

While zanko se lahko uporabi za ponovitev bloka kode določenega števila korakov.

In [27]:

```
i = 0
while i < 10:
    print(f'Repeated {i} times')
    i += 1
```

```
Repeated 0 times
Repeated 1 times
Repeated 2 times
Repeated 3 times
Repeated 4 times
Repeated 5 times
Repeated 6 times
Repeated 7 times
Repeated 8 times
Repeated 9 times
```

In [16]:

```
#A common use of the while loop is to do things like these:

temperature = 15

while temperature < 20:
    print('Heating...')
    temperature += 1

#Only instead of the temperature increasing continuously, we would e.g. get it from
#Remember to always have a way of exiting the loop! Otherwise it will run endlessly
```

```
Heating...
Heating...
Heating...
Heating...
Heating...
```

Obstaja tud while else.

```
while <expr>:
    <statement(s)>
else:
    <additional_statement(s)>
```

The <additional_statement(s)> specified in the else clause will be executed when the while loop terminates.
About now, you may be thinking, “How is that useful?” You could accomplish the same thing by putting those statements immediately after the while loop , without the else:

What’s the difference?

In the latter case, without the else clause, <additional_statement(s)> will be executed after the while loop terminates, no matter what.

When <additional_statement(s)> are placed in an else clause, they will be executed only if the loop terminates “by exhaustion”—that is, if the loop iterates until the controlling condition becomes false. If the loop is exited by a break statement, the else clause won’t be executed.

Vaja 01

Napišite program, ki izpiše prvih 10 sodih števil.

In [29]:

```
counter = 0
number = 1

while counter<10:
    if number%2 == 0:
        print(number)
        counter += 1
    number += 1
```

```
2
4
6
8
10
12
14
16
18
20
```

Vaja 02

Uporabnik naj vnese željeno dolžino Fibonaccijevega zaporedja. Program naj nato to zaporedje shrani v list in ga na koncu izpiše.

```
Fibonacci sequence

0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

In [1]:

```
"""
Uporabnik naj vnese željeno dolžino Fibonaccijevega
zaporedja. Program naj nato to zaporedje shrani
v list in ga na koncu izpiše.

Fibonacci sequence
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
"""
dolzina = int(input("Vnesi dolzino fibonacijevega zaporedja: "))
fibonacci = [0, 1]
if dolzina == 0:
    fibonacci = []
elif dolzina == 1:
    fibonacci = [0]
else:
    counter = 0
    while counter < dolzina - 2:
        nova_cifra = fibonacci[-1] + fibonacci[-2]
        print(nova_cifra)
        fibonacci.append(nova_cifra)
        counter += 1
print(fibonacci)
```

```
Vnesi dolzino fibonacijevega zaporedja: 10
1
2
3
5
8
13
21
34
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

For loop

Uporablja se kadar hočemo izvesti blok kode za vnaprej določeno število ponovitev.

Primer: kadar hočemo izvesti blok kode za vsak element v list-u.

```
for <var> in <iterable>:
    <statement(s)>
```

In [17]:

```
primes = [2, 3, 5, 7, 11] #itrable
for prime in primes:
    print(f'{prime} is  a prime number.')
```

```
2 is  a prime number.
3 is  a prime number.
5 is  a prime number.
7 is  a prime number.
11 is  a prime number.
```

In [18]:

```
kid_ages = (3, 7, 12)
for age in kid_ages:
    print(f'I have a {age} year old kid.')
```

```
I have a 3 year old kid.
I have a 7 year old kid.
I have a 12 year old kid.
```

Velikokrat se skupaj z for-loop uporablja funkcija range().

```
range(start, stop, step)
```

- start - Optional. An integer number specifying at which position to start. Default is 0
- stop - An integer number specifying at which position to end, excluding this number.
- step - Optional. An integer number specifying the incrementation. Default is 1

Funkcija range nam zgenerira list števil.

In [24]:

```
x = range(-5, 10, 1)
print(type(x))
print(list(x))
```

```
<class 'range'>
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [41]:

```
# Primer: Iteracija čez dictionary
pets = {
    'macka': 6,
    'pes': 12,
    'krava': 20
}

for pet, years in pets.items():
    print(f'{pet} je star/a {years} let.')
```

```
macka je star/a 6 let.
pes je star/a 12 let.
krava je star/a 20 let.
```

Nasveti

- Use the enumerate function in loops instead of creating an “index” variable

In [ ]:

```
```

Programmers coming from other languages are used to explicitly declaring a variable to track the index of a container in a loop. For example, in C++:

```
for (int i=0; i < container.size(); ++i)
{
// Do stuff
}
```

In Python, the enumerate built-in function handles this role.

In [7]:

```
moj_list = ["Anže", "Luka", "Mojca"]
index = 0
for element in moj_list:
    print (f'{index} {element}')
    index += 1
```

```
0 Anže
1 Luka
2 Mojca
```

In [6]:

```
#Idiomatic
moj_list = ["Anže", "Luka", "Mojca"]
for index, element in enumerate(moj_list):
    print (f'{index} {element}')
```

```
0 Anže
1 Luka
2 Mojca
```

Vaja 01

Iz danega dictionary izpišite vse ključe, katerih vrednost vsebuje črko **r** ali **R**.

In [81]:

```
d = {
    "mačka": "Micka",
    "pes": "Fido",
    "volk": "Rex",
    "medved": "Žan",
    "slon": "Jan",
    "žirafa": "Helga",
    "lev": "Gašper",
    "tiger": "Anže",
    "papagaj": "Črt",
    "ribica": "Elena",
    "krokodil": "Kasper",
    "zajec": "Lars",
    "kamela": "Manca"
}


for key,value in d.items():
    if "r" in value or "R" in value:
        print(f"{key}")
```

```
volk
lev
papagaj
krokodil
zajec
```