Vaja 01

Napišite program, ki bo uporabnika uprašal naj vnese neko celoštevilsko vrednost. Program naj nato izpiše ali je vrednost deljiva z 3 in z 7, ali ne.

In [3]:

```
x = int(input("Vnesi celoštevilsko vrednost: "))
if x%3 == 0 and x%7==0:
    print("Število je deljivo s 3 in s7")
else:
    print("Število ni deljivo s 3 ali s 7")
```

```
Vnesi celoštevilsko vrednost: 21
Število je deljivo s 3 in s7
```

In [ ]:

```

```

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.

In [1]:

```
counter = 1
while counter<5:
    print("Counter: ", counter)
    if counter == 3: break # comment or uncomment to see how it affects the program
    counter+=1
else:
    print("Else statement")

print("Nadaljevanje programa")
```

```
Counter:  1
Counter:  2
Counter:  3
Nadaljevanje programa
```

In [ ]:

```

```

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]:

```
x = int(input("Dolžina Fibonnacijevega zaporedja: "))
fibonacci = [0, 1]
counter = 2

while counter < x: # while len(fibonacci) < x  bi tud šlo
    fibonacci.append(fibonacci[-1] + fibonacci[-2])
    counter += 1
print(fibonacci)
```

```
Dolžina Fibonnacijevega zaporedja: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
```

In [ ]:

```

```

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

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
```

Break

Break keyword terminira najbolj notranjo zanko v kateri se nahaja.

In [44]:

```
avti = ["ok", "ok", "ok", "slab", "ok"]

for avto in avti:
    if avto == "slab":
        print("Avto je zanič.")
        break
    print("Avto je ok.")
    print("Naslednji korak zanke")
print("End")
```

```
Avto je ok.
Naslednji korak zanke
Avto je ok.
Naslednji korak zanke
Avto je ok.
Naslednji korak zanke
Avto je zanič.
End
```

In [ ]:

```

```

Continue

Continue keyword izpusti kodo, ki se more še izvesti, in skoči na naslednjo iteracijo zanke .

In [45]:

```
avti = ["ok", "ok", "ok", "slab", "ok"]

for avto in avti:
    if avto == "slab":
        print("Avto je zanič.")
        continue #continue #lah pokažeš še primer k je stvar zakomentirana
    print("Avto je ok.")
    print("Naslednji korak zanke")
print("End")
```

```
Avto je ok.
Naslednji korak zanke
Avto je ok.
Naslednji korak zanke
Avto je ok.
Naslednji korak zanke
Avto je zanič.
Avto je ok.
Naslednji korak zanke
End
```

In [ ]:

```

```

Vaja 01

Iz danega dictionary izpišite vse ključe, katerih vrednost vsebuje črko **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
```

Vaja 02

Poiščite vsa praštevila med 2 in 30.

In [76]:

```
for num in range(2,51):
    prime = True
    for i in range(2,num):
        #print(f"{num} / {i}. Ostanek je {num%i}")
        if (num%i==0):
            prime = False
            break
    if prime:
        print(f"{num} JE praštevilo!")
    #else:
        #print(f"{num} NI praštevilo.")
```

```
2 JE praštevilo!
3 JE praštevilo!
5 JE praštevilo!
7 JE praštevilo!
11 JE praštevilo!
13 JE praštevilo!
17 JE praštevilo!
19 JE praštevilo!
23 JE praštevilo!
29 JE praštevilo!
31 JE praštevilo!
37 JE praštevilo!
41 JE praštevilo!
43 JE praštevilo!
47 JE praštevilo!
```

In [ ]:

```
```

Funkcije

Funkcija je blok kode, ki izvede specifično operacijo in jo lahko večkrat uporabimo.

> Za primer, če v programu večkrat uporabniku rečemo, naj vnese celo število med 1 in 20. Od njega zahtevamo vnos s pomočjo **input** in nato to spremenimo v celo število z uporabo **int**. Nato preverimo ali je število v pravilnem rangu. To zaporedje kode v programu večkrat ponovimo.

> Če se sedaj odločimo, da naj uporabnik vnese celo število v rangu med 1 in 100, moramo popraviti vsako vrstico posebej, kar hitro lahko privede do napake.

> Za lažje pisanje programa lahko to zaporedje kode shranimo v funkcijo. Če sedaj spremenimo rang, le-tega popravimo samo enkrat, znotraj naše funkcije.

Funkcije nam omogočajo uporabo tuje kode brez globjega razumevanja kako le-ta deluje. Z njihovo pomočjo lahko zelo kompleksne probleme razbijemo na majhne in bolj obvladljive komponente.

Defining a Function

Funkcijo definiramo z uporabo def keyword kateri sledi ime funkcije in oglati oklepaji (). Zaključi se jo z " : ".

Blok kode, katero želimo, da naša funkcija izvede zapišemo z ustreznim zamikom.

```
def ime_funkcije():
    # Naš blok kode katero želimo izvesti
    x = input("...")
    y = int(x) + 5
    ...
```

Po priporočilih se imena funkcije piše na snake_case način (vse male črke, med besedami podčrtaj _)

Funkcijo nato uporabimo tako, da jo pokličemo po imenu in dodamo zraven ().

```
ime_funkcije() # Klic naše funkcije
```

In [28]:

```
def hello():
    print("Hello, World!")

print("Začetek programa")
hello()
print("Nadaljevanje programa")
#pokažemo, da moremo funkcijo klicat po definiciji.
#pazt, če to kažeš v jupyter notebooku, k tm se shranjo stvari v ozadju
```

```
Začetek programa
Hello, World!
Nadaljevanje programa
```

Funkcije je v kodi potrebno ustvariti, še predno jo kličemo.

In [30]:

```
print("Začetek programa")
hello2()
print("Nadaljevanje programa")

def hello2():
    print("Hello, World!")
```

```
Začetek programa

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
 last)
<ipython-input-30-d0c6cd4e0154> in <module>
      1 print("Začetek programa")
----> 2 hello2()
      3 print("Nadaljevanje programa")
      4
      5 def hello2():

NameError: name 'hello2' is not defined
```

In [ ]:

```
```

Naloga:

Napišite funkcijo, ki od uporabnika zahteva naj vnese svojo EMŠO število.

Funkcija naj nato izpiše koliko let je uporabnik star.

EMŠO ima 14 številk XXXXyyyXXXXXXX. 5.,6.,7. številka predstavljajo letnico rojstva (999 -> 1999 leto rojstva).

Primeri:

```
Input:
Vnesi emšo: 0102999500111

Output:
Star si 22 let

Input:
Vnesi emšo: 0104986505555

Output:
Star si 35 let
```

In [1]:

```
# Rešitev
def fun():
    emšo = input("Vnesi emšo: ")
    if emšo[4] == "0":
        letnica = int(emšo[4:7]) + 2000
    else:
        letnica = int(emšo[4:7]) + 1000
    print(f"Star si {2022-letnica} let")

fun()
```

```
Vnesi emšo: 0102999500111
Star si 23 let
```

In [ ]:

```
```

In [ ]:

```
```