TABELE (MESTO V TABELI SE ZAČNE Z 0!)
Metode:
append()...doda element na konec tabele
clear()...izbriše vse elemente v tabeli
copy()...vrne kopijo tabele
count()...vrne število teh el. V tab
extend()...doda vse elemente iz druge tabele
index()...vrne indeks prvega elementa neke spremenljivke
insert()...doda element v tabeli v specifičen indeks (pozicijo)
pop()...odstrani elemente na specifični lokaciji
remove()...odstrani element s specifično vrednostjo
sorted(tab)... urejena tabela
sum()...vrne vsoto elementov tabele
max()...vrne največji element
min()...vrne najmanjši element

RANDOM:
import random
random()...vrne med 0 in 1 (float)
randint(a, b)...vrne med a in b (samo cela števila) (int)
uniform(a, b)...vrne med a in b (tudi float)
shuffle() ... premeša elemente

==========

input(" ")... uporabnik VPIŠE
print()...izpišemo
return...vrnemo (uporabimo v funkcijah)
print("Blablabla", "neki", "jp", sep = "4", end = "!!!")-----> izpiše: Blablabla4neki4jp!!!
sep = "4"...presledek (v tem primeru je število 4 namesto presledka)
end = "!!!"...s čim se string konča; privzeto je '\n'

'naredi niz iz elemntov v tabeli'.join(tab) ----------

import math;
math.sin, math.cos, math.tan, math.sqrt, math.pi
a / b...navadno deljenje
a // b...celoštevilsko deljenje (12 // 10 = 1)
a % b...ostanek pri deljenju (5 % 3 = 2)
a * b...množenje
a ** b = a ^ b...a na b (a * a * a...) ^ b-krat

DELO Z OS:
os.listdir(mapa): □ seznam datotek in map v imeniku/mapi
os.getcwd() [pove trenutni delovni imenik]
os.chdir(d) [spremenimo delovni imenik]
os.mkdir(d) [ustvari imenik]
os.rmdir(d) [zbriši imenik]
os.rename(kaj, kam) [preimenuj imenik]
os.path.exists(f)
os.path.basename(f) --- datoteka
os.path.dirname(f) --- imenik
os.path.splitext(f) --- dobimo par (osnova, podaljšek)
os.path.split(f) --- par (imenik, dat)
os.path.splitdrive(f) --- par (enota, ostalo)
os.path.isfile(dat) … True/False: ali je dat »navadna« datoteka
os.path.isdir(dat)…True/False: ali je dat imenik
os.path.getsize(dat)… velikost datoteke
os.path.split(dat)…razdeli ime datoteke
os.path.getmtime(dat)…čas zadnje spremembe datoteke
os.rmdir(pot) □ odstrani (prazen) imenik/mapo
Pri imenih datotek lahko uporabljamo obliko r'L:\Primer\naloge1.txt' namesto 'L:\\Primer\\naloge1.tx

SLOVARJI:
POMEMBNO!!! – vrstnega reda elementov v slovarju načeloma ne poznamo
slovar = {ključ1: vrednost1, ključ2: vrednost2...} ali dict()
len(slovar) # dolžina (število "parov")
slovar.keys() # skoraj seznam ključev
slovar.values() # skoraj seznam vrednosti
slovar.items() # skoraj seznam parov (ključ, vrednost)
list(slovar.keys())# seznam ključev
slovar.update(sl2) # doda pare (ključ, vrednost) iz sl2 v slovar
slovar.get(ključ, privzetaVrednost) … Z get dobimo slovar[ključ], če je ključ v slovarju, oziroma privzetaVrednost drugače.
slovar.copy() # kopiramo slovar
slovar.clear() # zbrišemo slovar
slovar.items() → [(kljuc1, vrednost1), (ključ2, vrednost2), ()...]
for ključ, vrednost in slovar.items()

MNOŽICE:
POMEMBNO!!! – vrstnega reda elementov v množici načeloma ne poznamo
Prazna mnozica = set()
mnozica.add() → doda element v množico
mnozica.difference() → vrne množico, ki vsebuje razliko dveh ali več množic
mnA.difference(mnB) ≡ mnA – mnB..razlika
mnozica.discard() → odstrani specifični element NE VRNE NAPAKE, ČE ELEMENTA NI V MNOŽICI
mnozica.intersection() → vrne množico, ki je presek dveh množic
mnA.intersection(mnB) ≡ mnA ∩ mnB…presek
mnozica.remove() → odstrani specifičen element VRNE NAPAKO, ČE ELEMENTA NI V MNOŽICI
množica.union() → vrne množico ki vsebuje unijo množic
set([6, 42, 1, 3, 1, 1, 6]) – izpise: {1, 42, 3, 6}, -set(range(5)) –izpise: {0, 1, 2, 3, 4}
NE! set ({{2}, {3, 4}})(pari,nabori niso el mnozic..!)

DATOTEKE:
Branje → datoteka = open('bla.txt', (encoding='utf8'))
Lahko: for vstica in open('datoteka')
Pisanje → datoteka = open('bla.txt', 'w') VEDNO NAREDI NOVO DATOTEKO
Pisanje → datoteka = open('bla.txt', 'a') ČE NE OBSTAJA DATOTEKA, JO NAREDI, SICER ZAČNE PISAT NA KONEC
Metode:
datoteka.read() → prebere celotno datoteko in vrne njeno vsebino kot niz (neobvezen argument je lahko število znakov)
datoteka.readline() → prebere vrstico v kateri se nahajamo
datoteka.readlines() → prebere vse vrstice in vrne seznam, kjer so elementi vrstice datoteke
datoteka.close() → datoteko zapremo
datoteka.write('niz') → v datoteko zapiše podan niz. NE GRE V NOVO VRSTO
print('niz', file=datoteka) → v datoteko zapiše podan niz in gre v novo vrstico
for vstica in open('datoteka'):
print(vrstica)___!!TUKAJ BO IZPISALO VMES PRAZNE VRSTICE!!!

IMPORT
- import dat
- Python prebere datoteko dat.py s kodo in jo izvede. Ne izvede vedno vsega!

- Če je datoteka s kodo glavni program, potem __name__ dobi vrednost __main__
- Če pa je datoteka s kodo "uvožena" v drugo (z import), potem pa __name__ postane enak imenu modula

Če želimo, da python izvede neko kodo, le če delamo v določeni datoteki, potem ta del ločimo z:

If __name__ == '__main__'.

Importamo lahko tudi z:

from dat import *(s tem omogočimo, da funkcije iz dat ni treba klicati kot dat.funkcija()…)

NAPAKE:

```
try:
    # nekaj kar poskušamo izpeljati
except: (Lahko prestrežemo točno določene napake(da poimenujemo to vrsto; npr except ValueError: …))
    # če je python sprožil izjemo jo tukaj prestrežemo in mu povemo kaj storimo.
    # lahko sprožimo napako z raise Exception.
finally:  # se izvede v vsakem primeru
```

DRUGE FUNKCIJE in METODE:

isinstance(x, pod. tip) → vrne ali je x vrsta nekega podatkovnega tipa (int, str, list, tuple, dict, set, ...)
map(funkcija, tab) → na vseh elementih tabele izvede dano funkcijo
zip(tab1, tab2) → združi elemente dveh tabel v tabelo parov (parov je toliko kot je dolga krajša tabela)
round(st, koliko_mest) → zaokroži dano število
strip() → odstrani presledke na vsaki strani niza
split() → vrne tabelo nizov
eval(niz) - izračuna »vrednost« niza (recimo da je nek račun zapisan v nizu)
round(število, na koliko mest) … zaokroži nam število
import time
time.time() … vrne trenutni čas (1590440115.3778644)
time.sleep(čas v sekundah ko python spi)
break … prekinemo zanko
pass … nič se ne stori

ŽELVJA GRAFIKA

forward() | fd()
backward() | bk() | back()
right() | rt()
left() | lt()
goto() | setpos() | setposition()
setx()
sety()
circle()
undo()
speed()

pendown() | pd() | down()
penup() | pu() | up()
pensize() | width()
pen()
isdown()

filling()
begin_fill()
end_fill()
write()

OOP

```
Class ImeRazreda:
        def __init__(self,…):
```

Magične metode:

__str__ (uporabimo z str(objekt) ali pa z print(objekt))
__add__(self,drugi)… drugi je ne desni strani
__radd__
__mul__
__rmul__
__repr__

__abs____len__
__neg__
__sub__
__rsub__
__truediv__ (/)
__floordiv__ (//)

REKURZIJA

Fakulteta(n), ki izračuna fakulteto števila n. Argument n je nenegativno celo število.

```
def fakulteta(n):

    '''Rekurzivni izračun fakultete'''

    if n == 0:

        return 1

    return n * fakulteta(n - 1)
```