# Izpit

Tekom izpita boste napisali svoj modul. Poimenujte ga **Ime_Priimek**. Ko končate z izpitom ga pošljite na gregor.balkovec@ltfe.org (mailto:gregor.balkovec@ltfe.org) in anze.glusic@ltfe.org (mailto:anze.glusic@ltfe.org) .

# Naloga 01

**Točka: / 5**

Napišite funkcijo `funkcija01` , ki kot prvi parameter sprejme premer krogle. Funkcija naj vrne izračunano prostornino krogle.

"π" najdete v built-in knjižnjici **math**.

```
import math
print(math.pi)

INPUT:
funkcija01(3)

OUTPUT:
14.1371
```

In [6]:

```
# Rešitev
import math

def funkcija01(d):
    return math.pi*d**3/6
```

In [7]:

```
# Test

print(funkcija01(3))
```

```
14.137166941154069
```

In [ ]:

```

```

# Naloga 02

**Točke: / 5**

Napišite funkcijo `funkcija02` , ki prejme list.

Vsak element v listu je tuple. Prvi element v tuplu je znamka avta. Drugi element je leto prve registracije avta. Tretji element je število prevoženih kilometrov.

Funkcija naj vrne list znamk avtomobilov, ki imajo leto prve registracije večje od 2015 in število prevoženih kilometrov manjše od 200 000.

```
INPUT:
l = [("ŠKODA OCTAVIA", 2015, 265_000), ("VOLKSWAGEN PASSAT", 2014, 199_000
),
     ("RENAULT CLIO", 2016, 211_000), ("CITROEN XSARA PICASSO", 2017, 120_0
00),
     ("BMW 318", 2016, 300_000), ("KIA CEED", 2016, 154_000)]

print(funkcija02(l))

OUTPUT:
['CITROEN XSARA PICASSO', 'KIA CEED']
```

In [12]:

```
l = [("ŠKODA OCTAVIA", 2015, 265_000), ("VOLKSWAGEN PASSAT", 2014, 199_000),
     ("RENAULT CLIO", 2016, 211_000), ("CITROEN XSARA PICASSO", 2017, 120_000),
     ("BMW 318", 2016, 300_000), ("KIA CEED", 2016, 154_000)]
```

In [14]:

```
# Rešitev
def funkcija02(l):
    result = []
    for car in l:
        if car[1] > 2015 and car[2] < 200_000:
            result.append(car[0])
    return result
```

In [15]:

```
# Test
l = [("ŠKODA OCTAVIA", 2015, 265_000), ("VOLKSWAGEN PASSAT", 2014, 199_000),
     ("RENAULT CLIO", 2016, 211_000), ("CITROEN XSARA PICASSO", 2017, 120_000),
     ("BMW 318", 2016, 300_000), ("KIA CEED", 2016, 154_000)]
print(funkcija02(l))
```

```
['CITROEN XSARA PICASSO', 'KIA CEED']
```

In [ ]:

```

```

# Naloga 03

**Točke: / 5**

Napišite funkcijo `funkcija03` , ki prejme dictionary oglasov nepremičnin v ljubljani.

Funkcija naj najde kateri oglas ima najugodnejšo ceno na kvadratni meter.

Funkcija naj nato vrne tuple 3 vrednosti:

- id oglasa (int)
- ime prodajalaca (string)
- ceno kvadratnega metra (float)

```
d = {
    "dravlje-stanovanje_6448914":{
        "cena": 311_000,
        "m2": 75.12,
        "prodajalec": "MESTO NEPREMIČNIN d.o.o.",
        "id": 6448914
    },
    "fuzine-dvigalo-balkon-stanovanje_6451032": {
        "cena": 229_900,
        "m2": 70,
        "prodajalec": "STAN nepremičnine d.o.o., Ljubljana",
        "id": 6451032
    },
    "kodeljevo-funkcionalno-stanovanje-z-vrtom-stanovanje_6437495": {
        "cena": 145_000,
        "m2": 39.1,
        "prodajalec": "ABC nepremičnine d.o.o.",
        "id": 6437495
    },
    "lj-bezigrad-stanovanje_6446295": {
        "cena": 125_000,
        "m2": 29.2,
        "prodajalec": "Mreža nepremičnin d.o.o.",
        "id": 6446295
    },
    "lj-bezigrad-stanovanje_6447284": {
        "cena": 620_000,
        "m2": 84.9,
        "prodajalec": "Mreža nepremičnin d.o.o.",
        "id": 6447284
    },
    "lj-bezigrad-ekskluzivno-petsobno-stanovanje-v-situli-stanovanje_635999
0": {
        "cena": 750_000,
        "m2": 188.9,
        "prodajalec": "ABC nepremičnine d.o.o.",
        "id": 6359990
    }
}

INPUT:
best_id, best_prodajalec, best_price_v_m2 = funkcija03(d)
print((f"Najboljša ponudba je ID:{best_id}, pri prodajalcu {best_prodajalec
} z ceno na kvadrat {best_price_v_m2:.2f} €"))

OUTPUT:
Najboljša ponudba je ID:6451032, pri prodajalcu STAN nepremičnine d.o.o., L
jubljana z ceno na kvadrat 3284.29 €
```

In [51]:

```
d = {
    "dravlje-stanovanje_6448914":{
        "cena": 311_000,
        "m2": 75.12,
        "prodajalec": "MESTO NEPREMIČNIN d.o.o.",
        "id": 6448914
    },
    "fuzine-dvigalo-balkon-stanovanje_6451032": {
        "cena": 229_900,
        "m2": 70,
        "prodajalec": "STAN nepremičnine d.o.o., Ljubljana",
        "id": 6451032
    },
    "kodeljevo-funkcionalno-stanovanje-z-vrtom-stanovanje_6437495": {
        "cena": 145_000,
        "m2": 39.1,
        "prodajalec": "ABC nepremičnine d.o.o.",
        "id": 6437495
    },
    "lj-bezigrad-stanovanje_6446295": {
        "cena": 125_000,
        "m2": 29.2,
        "prodajalec": "Mreža nepremičnin d.o.o.",
        "id": 6446295
    },
    "lj-bezigrad-stanovanje_6447284": {
        "cena": 620_000,
        "m2": 84.9,
        "prodajalec": "Mreža nepremičnin d.o.o.",
        "id": 6447284
    },
    "lj-bezigrad-ekskluzivno-petsobno-stanovanje-v-situli-stanovanje_6359990": {
        "cena": 750_000,
        "m2": 188.9,
        "prodajalec": "ABC nepremičnine d.o.o.",
        "id": 6359990
    }
}
```

In [52]:

```
# Rešitev
def funkcija03(data):
    best_prodajalec = None
    best_id = None
    best_price_v_m2 = 100000
    for key, value in data.items():
        if value["cena"]/value["m2"] < best_price_v_m2:
            best_prodajalec = value["prodajalec"]
            best_id = value["id"]
            best_price_v_m2 = value["cena"]/value["m2"]

    return best_id, best_prodajalec, best_price_v_m2
```

In [53]:

```
# Test
best_id, best_prodajalec, best_price_v_m2 = funkcija03(d)
print((f"Najboljša ponudba je ID:{best_id}, pri prodajalcu {best_prodajalec} z ceno
```

```
Najboljša ponudba je ID:6451032, pri prodajalcu STAN nepremičnine d.o.
o., Ljubljana z ceno na kvadrat 3284.29 €
```

In [ ]:

```

```

# Naloga 04

**Točke: /5**

Napišite funkcijo `funkcija04` . Funkcija naj odpre datoteko `input.txt` v kateri imamo napisan recept za peko palačink vendar pa so koraki v napačnem zaporedju.

Funkcija na korake razporedi v pravilni zaporedje in jih nato zapiše v datoteko `output.txt` .

Lahko privzamete, da bo številka koraka vedno zavzela prvi dve mesti vrstice.

```
INPUT:
03. Segrejte ponev in nanjo vlijte olje.
01. Zmešajte 0.5l mleka, 2 jajci in 1 žlico olja.
04. Na ponev zlijte 1 zajemalko mase.
06. Palačinko namažite s čokolado ali pa marmelado in kislo smetano.
05. Palačinko popecite na obeh straneh, da pridobi zlato-rjavo barvo.
02. Masi dodajte 25dkg bele moke in sol.


OUTPUT:
01. Zmešajte 0.5l mleka, 2 jajci in 1 žlico olja.
02. Masi dodajte 25dkg bele moke in sol.
03. Segrejte ponev in nanjo vlijte olje.
04. Na ponev zlijte 1 zajemalko mase.
05. Palačinko popecite na obeh straneh, da pridobi zlato-rjavo barvo.
06. Palačinko namažite s čokolado ali pa marmelado in kislo smetano.
```

In [64]:

```
# Rešitev
def funkcija04():
    with open("input.txt") as f:
        data = f.readlines()
        data = sorted(data, key=lambda x: int(x[:2]))
        #print(data)

    with open("output.txt", "w") as f:
        for line in data:
            f.write(line)
```

In [66]:

```
# Test
funkcija04()
with open("output.txt") as f:
    for line in f.readlines():
        print(line)
```

```
01. Zmešajte 0.5l mleka, 2 jajci in 1 žlico olja.

02. Masi dodajte 25dkg bele moke in sol.

03. Segrejte ponev in nanjo vlijte olje.

04. Na ponev zlijte 1 zajemalko mase.

05. Palačinko popecite na obeh straneh, da pridobi zlato-rjavo barvo.

06. Palačinko namažite s čokolado ali pa marmelado in kislo smetano.
```

In [ ]:

```

```

# Naloga 05

**Točke: /5**

Napišite razred `Volk`. Ko ustvarimo novo instanco razreda, vanj shranimo ime in njegovo hitrost.

Napišite tudi razred `Trop`. Ko ustvarimo novo instanco razreda lahko kot argument pošljemo list volkov v tropu ali pa tudi ne.

Trop naj ima spremenljivko `volkovi`, ki je list volkov, ki se nahajajo v tropu.

Trop naj ima funkcijo `dodaj_volka`, ki kot argument prejme instanco razreda Volk in le-tega doda v svoj list volkov.

Trop naj ima funkcijo `najhitrejsi_volk`, ki naj vrne ime najhitrejšega volka.

Trop naj ima funkcijo `razvrsti_po_hitrosti`, ki naj volkove razvrsti od najpočasnejšega do najhitrejšega. Nato naj funkcija vrne list samo imen volkov v pravilnem vrstnem redu.

```
rex = Volk("Rex", 51)
milo = Volk("Milo", 55)
leo = Volk("Leo", 59)

trop = Trop([rex, milo, leo])
print(trop.najhitrejsi_volk())
OUTPUT:
Leo

chip = Volk("Chip", 49)
bruce = Volk("Bruce", 62)

trop.dodaj_volka(chip)
trop.dodaj_volka(bruce)
print(trop.najhitrejsi_volk())
OUTPUT:
Bruce


print(trop.razvrsti_po_hitrosti())
OUTPUT:
['Chip', 'Rex', 'Milo', 'Leo', 'Bruce']
```

In [80]:

```
# Rešitev

class Volk:
    def __init__(self, ime, hitrost):
        self.ime = ime
        self.hitrost = hitrost

class Trop:
    def __init__(self, volkovi=[]):
        self.volkovi = volkovi

    def dodaj_volka(self, volk):
        self.volkovi.append(volk)

    def najhitrejsi_volk(self):
        best_wolf = self.volkovi[0]
        for volk in self.volkovi[1:]:
            if volk.hitrost > best_wolf.hitrost:
                best_wolf = volk
        return best_wolf.ime

    def razvrsti_po_hitrosti(self):
        self.volkovi = sorted(self.volkovi, key=lambda volk: volk.hitrost)
        return [volk.ime for volk in self.volkovi]
```

In [82]:

```
# Test
rex = Volk("Rex", 51)
milo = Volk("Milo", 55)
leo = Volk("Leo", 59)

trop = Trop([rex, milo, leo])
print(trop.najhitrejsi_volk())

chip = Volk("Chip", 49)
bruce = Volk("Bruce", 62)

trop.dodaj_volka(chip)
trop.dodaj_volka(bruce)
print(trop.najhitrejsi_volk())
print(trop.razvrsti_po_hitrosti())
```

```
Leo
Bruce
['Chip', 'Rex', 'Milo', 'Leo', 'Bruce']
```

In [ ]:

In [ ]:

In [ ]: