PROGRAMIRANJE

ZBIRKA ZADATAKA

Ishod učenja 5

2023-2024.

Zadatak 1

Napišite program koji od korisnika učitava datum i vrijeme početka prvog predavanja te ispisuje termine svih 15 predavanja u hrvatskom formatu, ako znamo da se svako predavanje odvija u istom terminu svakog tjedna.

Moguće rješenje:

```
import datetime

prvo_predavanje = datetime.datetime.strptime(
    input("Upišite datum i vrijeme početka prvog predavanja u hrvatskom formatu: "),
    "%d.%m.%Y. %H:%M"
)

trenutno_predavanje = prvo_predavanje
razmak = datetime.timedelta(weeks=1)

for i in range(15):
    print("{}. predavanje će se održati: {}".format(i + 1,
trenutno_predavanje.strftime("%d.%m.%Y. %H:%M")))
    trenutno_predavanje += razmak
```

Zadatak 2

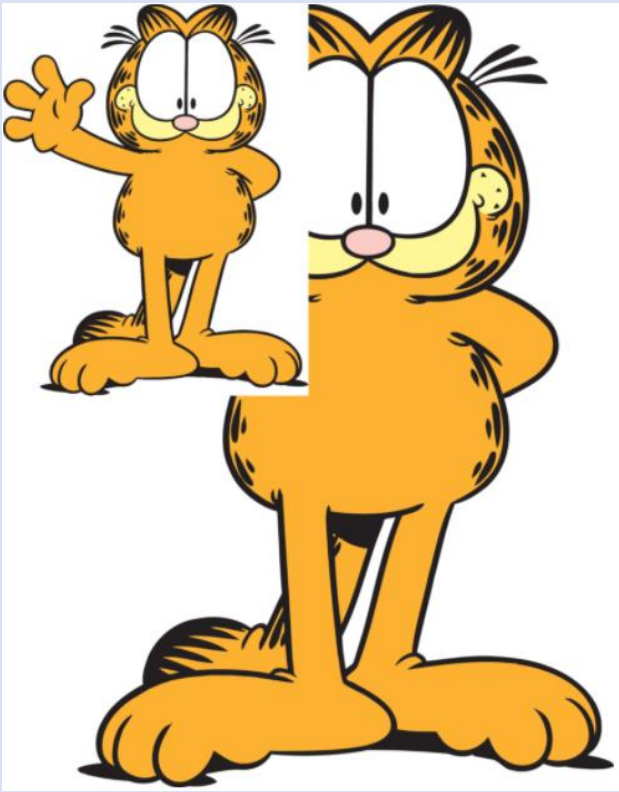
Napišite program koji omogućuje korisniku da upiše datum početka i kraja godišnjeg odmora pa mu izračunajte i ispišite koliko će radnih dana potrošiti (smatrajte da su radni dani ponedjeljak-petak).

Moguće rješenje:

```
import datetime

pocetak_go = datetime.datetime.strptime(
    input("Upišite prvi dan GO u hrvatskom formatu: "),
    "%d.%m.%Y."
)

kraj_go = datetime.datetime.strptime(
    input("Upišite zadnji dan GO u hrvatskom formatu: "),
    "%d.%m.%Y."
)

trenutni_dan = pocetak_go
razmak = datetime.timedelta(days=1)
radnih_dana = 0

while trenutni_dan <= kraj_go:
    if trenutni_dan.isoweekday() in [1, 2, 3, 4, 5]:
        radnih_dana += 1
    trenutni_dan += razmak

print("Potrošit ćete {} radnih dana".format(radnih_dana))
```

Zadatak 3

Izradite zadanu sliku koristeći original dostupan na /home/gdjambic/images/garfield.png. Manji Garfield ima 2 puta manju širinu i proporcionalno manju visinu od originala.



Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

im = PIL.Image.open("/home/gdjambic/images/garfield.png")
sirina = im.size[0]
visina = im.size[1]
omjer = sirina / visina
print(sirina, visina, omjer)

nova_sirina = int(sirina / 2)
nova_visina = int(nova_sirina / omjer)
print(nova_sirina, nova_visina)

smanjeni = im.resize(size=(nova_sirina, nova_visina))

im.paste(smanjeni, box=(0, 0, nova_sirina, nova_visina))

IPython.display.display(im)
```

Zadatak 4

Danas u 20:30 sati je utakmica koju želite gledati, ali imate i četiri stvari za obaviti prije utakmice:

- Naučiti novo gradivo iz RAUP-a (trebat će vam 2 sata za to)
- Skočiti do dućana pa namirnice za kuhanje (za to vam je dovoljno pola sata)
- Nazvati mamu (razgovor će svakako završiti za 10 minuta)
- Izribati kamenac s kade (20 minuta je dovoljno)

Napišite program koji će vam odgovoriti na pitanje: kad najkasnije morate početi obavljati ova četiri posla tako da završite točno 5 minuta prije početka utakmice? Program napišite tako da je lako dodavati nove obaveze, a da program i dalje zna izračunati kad treba početi s poslovima.

Moguće rješenje:

```
import datetime

# sve što treba obaviti ide u listu
poslovi = [
    datetime.timedelta(hours=2), # RAUP
    datetime.timedelta(minutes=30), # dućan
    datetime.timedelta(minutes=10), # poziv
    datetime.timedelta(minutes=20), # kada
]

# utakmica je danas u 20:30
danas = datetime.date.today()
utakmica = datetime.datetime(day=danas.day, month=danas.month, year=danas.year,
hour=20, minute=30)

# želim završiti sve poslove 5 minuta prije utakmice
granica = utakmica - datetime.timedelta(minutes=5)

# sad oduzmem trajanje svih poslova kako bih dobio vrijeme kad trebam početi
obavljati poslove
for p in poslovi:
    granica -= p

# ispišem kad trebam početi obavljati poslove
print("Poslove treba početi obavljati najkasnije danas u
{}".format(granica.strftime("%H:%M")))
```

Zadatak 5

Napišite program koji ispisuje sve prijestupne godine od 2000-te do 2100-te (uključujući i njih). Koristite činjenicu da u prijestupnoj godini veljača ima 29 dana.

Moguće rješenje:

```
import datetime

start = datetime.date(day=1, month=1, year=2000)
end = datetime.date(day=31, month=12, year=2100)
trenutno = start
za_dodati = datetime.timedelta(days=1)

prijestupne_godine = []

while trenutno <= end:
    if trenutno.month == 2 and trenutno.day == 29:
        prijestupne_godine.append(trenutno.year)
    trenutno += za_dodati

print("Prijestupne godine su {}".format(prijestupne_godine))
```

Zadatak 6

Napišite program koji od korisnika učitava mjesec, a zatim ispisuje kalendar sa svim danima u mjesecu, kao u prikazu:

```
Upišite mjesec: 2
PON    UTO    SRI    ČET    PET    SUB    NED
                     1      2      3      4
5      6      7      8      9      10     11
12     13     14     15     16     17     18
19     20     21     22     23     24     25
26     27     28     29
```

Moguće rješenje:

```
import datetime

mjesec =int(input("Upišite mjesec: "))

start = datetime.date(day=1, month=mjesec, year=2024)
end = datetime.date(day=1, month=mjesec + 1, year=2024)
end -= datetime.timedelta(days=1)
trenutno = start

# ispišemo zaglavlje
print("PON\tUTO\tSRI\tČET\tPET\tSUB\tNED")

# ostavimo prazno dane u tjednu prije početka mjeseca
prvi_dan_mjeseca = start.isoweekday()
for i in range(prvi_dan_mjeseca - 1):
    print("\t", end="")

while trenutno <= end:
    print("{}\t".format(trenutno.day), end="")

    # ako sam upravo ispisao nedjelju, ide novi red
    if trenutno.isoweekday() == 7:
        print()

    trenutno += datetime.timedelta(days=1)
```

Zadatak 7

Definirajte klasu Datoteka koja omogućuje čuvanje podataka o nazivu i o vremenu kad je datoteka kreirana. Kreirajte nekoliko objekata tipa Datoteka. Nakon toga, omogućite korisniku da odabere opciju želi li:

- Prikazati datoteke koje su kreirane danas.
- Prikazati datoteke koje su kreirane u zadnjih *n* dana (učitajte *n* od korisnika).

Moguće rješenje:

```
import datetime

class Datoteka:
    def __init__(self, naziv, vrijeme_kreiranja):
        self.naziv = naziv
        self.vrijeme_kreiranja = vrijeme_kreiranja

fs = [
    Datoteka("plaće.xlsx", datetime.datetime(day=19, month=12, year=2023, hour=12,
minute=22, second=11)),
    Datoteka("predavanje11.pptx", datetime.datetime(day=21, month=12, year=2023,
hour=19, minute=10, second=40)),
    Datoteka("pas_i_ja.jpeg", datetime.datetime(day=26, month=12, year=2023, hour=9,
minute=32, second=55)),
    Datoteka("završni rad.docx", datetime.datetime(day=10, month=1, year=2024,
hour=22, minute=52, second=2)),
    Datoteka("brojevi.txt", datetime.datetime(day=12, month=1, year=2024, hour=0,
minute=0, second=58)),
]

def prikazi(od):
    # dobili smo date, a treba nam datetime pa ga sad napravimo
    pocetni_datum_vrijeme = datetime.datetime(day=od.day, month=od.month,
year=od.year)

    for dat in fs:
        if dat.vrijeme_kreiranja >= pocetni_datum_vrijeme:
            print("Pronađena je datoteka: {} (kreirana {})".format(
                dat.naziv, dat.vrijeme_kreiranja.strftime("%d.%m.%Y %H:%M")))

print("A = prikaži datoteke koje su kreirane danas")
print("B = prikaži datoteke koje su kreirane u zadnjih n dana")
odabir = input("> ")

if odabir == "A":
    prikazi(datetime.date.today())
elif odabir == "B":
    n = int(input("Koliko dana: "))
    prikazi(datetime.date.today() - datetime.timedelta(days=n))
```

Zadatak 8

Napišite program koji od korisnika učitava dva datuma u ISO formatu, a zatim ispisuje četvrtaka ima između njih (uključujući i te datume).

Moguće rješenje:

```
import datetime

pocetak = datetime.datetime.strptime(
    input("Upišite prvi datum u ISO formatu: "), "%Y-%m-%d"
)
kraj = datetime.datetime.strptime(
    input("Upišite zadnji datum u ISO formatu: "), "%Y-%m-%d"
)

trenutni = pocetak
cetvrtaka = 0

while trenutni <= kraj:
    if trenutni.isoweekday() == 4:
        cetvrtaka += 1
    trenutni += datetime.timedelta(days=1)

print("Između unesenih datuma ima {} četvrtaka".format(cetvrtaka))
```

Zadatak 9

Napišite program koji od korisnika učitava datum, a zatim ispisuje cijeli tjedan u kojemu se nalazi taj datum, uz naglašavanje upisano datuma pomoću zvjezdice. Primjer ispisa:

```
Upišite datum u hrvatskom formatu: 12.01.2024.
PON, 08.01.2024.
UTO, 09.01.2024.
SRI, 10.01.2024.
ČET, 11.01.2024.
PET, 12.01.2024.*
SUB, 13.01.2024.
NED, 14.01.2024.
```

Moguće rješenje:

```
import datetime

def ispisi_dan(broj):
    if broj == 1:
        print("PON, ", end="")
    elif broj == 2:
        print("UTO, ", end="")
    elif broj == 3:
        print("SRI, ", end="")
    elif broj == 4:
        print("ČET, ", end="")
    elif broj == 5:
        print("PET, ", end="")
    elif broj == 6:
        print("SUB, ", end="")
    else:
        print("NED, ", end="")

datum = datetime.datetime.strptime(
    input("Upišite datum u hrvatskom formatu: "),
    "%d.%m.%Y."
)

# pronađemo ponedjeljak
ponedjeljak = datum - datetime.timedelta(days=(datum.isoweekday() - 1))
nedjelja = ponedjeljak + datetime.timedelta(days=6)
trenutno = ponedjeljak

while trenutno <= nedjelja:
    ispisi_dan(trenutno.isoweekday())

    print(trenutno.strftime("%d.%m.%Y."), end="")
    if trenutno == datum:
        print("*", end="")

    print()
    trenutno += datetime.timedelta(days=1)
```

Zadatak 10

Sigurnosna politika firme kaže da se pune sigurnosne kopije produkcijske baze podataka izrađuju u ponoć s nedjelje na ponedjeljak i u ponoć sa srijede na četvrtak, a da se inkrementalne sigurnosne kopije baze izrađuju točno dva dana nakon svake pune kopije. Napišite program koji korisniku ispisuje plan izrade sigurnosnih kopija za 30 dana počevši od sutra. Primjer rada programa:

```
13.01.2024. - Inkrementalna kopija
14.01.2024. - ništa
15.01.2024. - Puna kopija u ponoć s nedjelje na ponedjeljak
16.01.2024. - ništa
17.01.2024. - Inkrementalna kopija
18.01.2024. - Puna kopija u ponoć sa srijede na četvrtak
19.01.2024. - ništa
20.01.2024. - Inkrementalna kopija
21.01.2024. - ništa
22.01.2024. - Puna kopija u ponoć s nedjelje na ponedjeljak
23.01.2024. - ništa
24.01.2024. - Inkrementalna kopija
25.01.2024. - Puna kopija u ponoć sa srijede na četvrtak
26.01.2024. - ništa
27.01.2024. - Inkrementalna kopija
28.01.2024. - ništa
29.01.2024. - Puna kopija u ponoć s nedjelje na ponedjeljak
30.01.2024. - ništa
31.01.2024. - Inkrementalna kopija
01.02.2024. - Puna kopija u ponoć sa srijede na četvrtak
02.02.2024. - ništa
03.02.2024. - Inkrementalna kopija
04.02.2024. - ništa
05.02.2024. - Puna kopija u ponoć s nedjelje na ponedjeljak
06.02.2024. - ništa
07.02.2024. - Inkrementalna kopija
08.02.2024. - Puna kopija u ponoć sa srijede na četvrtak
09.02.2024. - ništa
10.02.2024. - Inkrementalna kopija
11.02.2024. - ništa
```

Moguće rješenje:

```
import datetime

trenutno = datetime.date.today() + datetime.timedelta(days=1)
kraj = trenutno + datetime.timedelta(days=29)

while trenutno <= kraj:
    print(trenutno.strftime("%d.%m.%Y. - "), end="")
    if trenutno.isoweekday() == 1:
        print("Puna kopija u ponoć s nedjelje na ponedjeljak")
    elif trenutno.isoweekday() == 4:
        print("Puna kopija u ponoć sa srijede na četvrtak")
    elif trenutno.isoweekday() in [3, 6]:
        print("Inkrementalna kopija")
    else:
        print("ništa")
```

```
    trenutno += datetime.timedelta(days=1)
```

Zadatak 11

Napišite program koji od korisnika učitava koliko epizoda serije želi pogledati, koliko traje jedna epizoda, te kad počinje gledati seriju. Nakon toga, ispišite korisniku kad će završiti s gledanjem serije, uzevši u obzir da između dvije epizode stavite 10 minuta pauze.

Moguće rješenje:

```
import datetime

epizoda = int(input("Koliko epizoda želite pogledati: "))
trajanje = int(input("Koliko minuta traje svaka epizoda: "))
pocetak = datetime.datetime.strptime(
    input("Kad počinjete gledati seriju: "),
    "%d.%m.%Y. %H:%M"
)

kraj = pocetak

for i in range(epizoda):
    # dodamo trajanje jedne epizode
    kraj += datetime.timedelta(minutes=trajanje)

    # ako nismo upravo pogledali zadnju epizodu, dodamo pauzu
    # (iza zadnje epizide ne ide pauza)
    if i < epizoda - 1:
        kraj += datetime.timedelta(minutes=10)

print("S gledanjem ćete završiti: {}".format(kraj.strftime("%d.%m.%Y. %H:%M")))
```

Zadatak 12

Napišite program koji prikazuje sliku /home/gdjambic/images/cvv.jpg, ali tako da CVV zamrljate da se ne može pročitati (pazite da zamrljate samo brojeve, crveni obrub okolo mora ostati jasno vidljiv:

Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

im = PIL.Image.open("/home/gdjambic/images/cvv.jpg")

cvv_lokacija = (815, 257, 885, 295)
cvv_slika = im.crop(box=cvv_lokacija)
cvv_slika = cvv_slika.filter(PIL.ImageFilter.GaussianBlur(radius=6))

im.paste(cvv_slika, box=cvv_lokacija)

IPython.display.display(im)
```

Zadatak 13

Napišite program koji koristi /home/gdjambic/images/snoopy.jpg i /home/gdjambic/images/nature.jpg te izrađuje i prikazuje sliku koja izgleda ovako (Snoopy ostaje u svojoj originalnoj veličini):

Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

snoopy = PIL.Image.open("/home/gdjambic/images/snoopy.jpg")
nature = PIL.Image.open("/home/gdjambic/images/nature.jpg")

pos_x = 1000
pos_y = 520
nature.paste(snoopy, box=(pos_x, pos_y, pos_x + snoopy.size[0], pos_y +
snoopy.size[1]))

snoopy = snoopy.transpose(PIL.Image.FLIP_LEFT_RIGHT)

pos_x = 520
pos_y = 450
nature.paste(snoopy, box=(pos_x, pos_y, pos_x + snoopy.size[0], pos_y +
snoopy.size[1]))

IPython.display.display(nature)
```

Zadatak 14

Napišite program koji koristi /home/gdjambic/images/snoopy.jpg i /home/gdjambic/images/nature.jpg te izrađuje i prikazuje sliku koja izgleda ovako (najbliži Snoopy je u 150% originalne veličine, srednji ostaje u originalnoj veličini, a najudaljeniji je na 50% originalne veličine):

Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

snoopy = PIL.Image.open("/home/gdjambic/images/snoopy.jpg")
nature = PIL.Image.open("/home/gdjambic/images/nature.jpg")

# najbliži
snoopy_najblizi = snoopy.resize(size=(
    int(snoopy.size[0] * 1.5),
    int(snoopy.size[1] * 1.5)
))

pos_x = 500
pos_y = 800
nature.paste(snoopy_najblizi, box=(pos_x, pos_y, pos_x + snoopy_najblizi.size[0],
pos_y + snoopy_najblizi.size[1]))

# srednji
pos_x = 700
pos_y = 550
nature.paste(snoopy, box=(pos_x, pos_y, pos_x + snoopy.size[0], pos_y +
snoopy.size[1]))
```

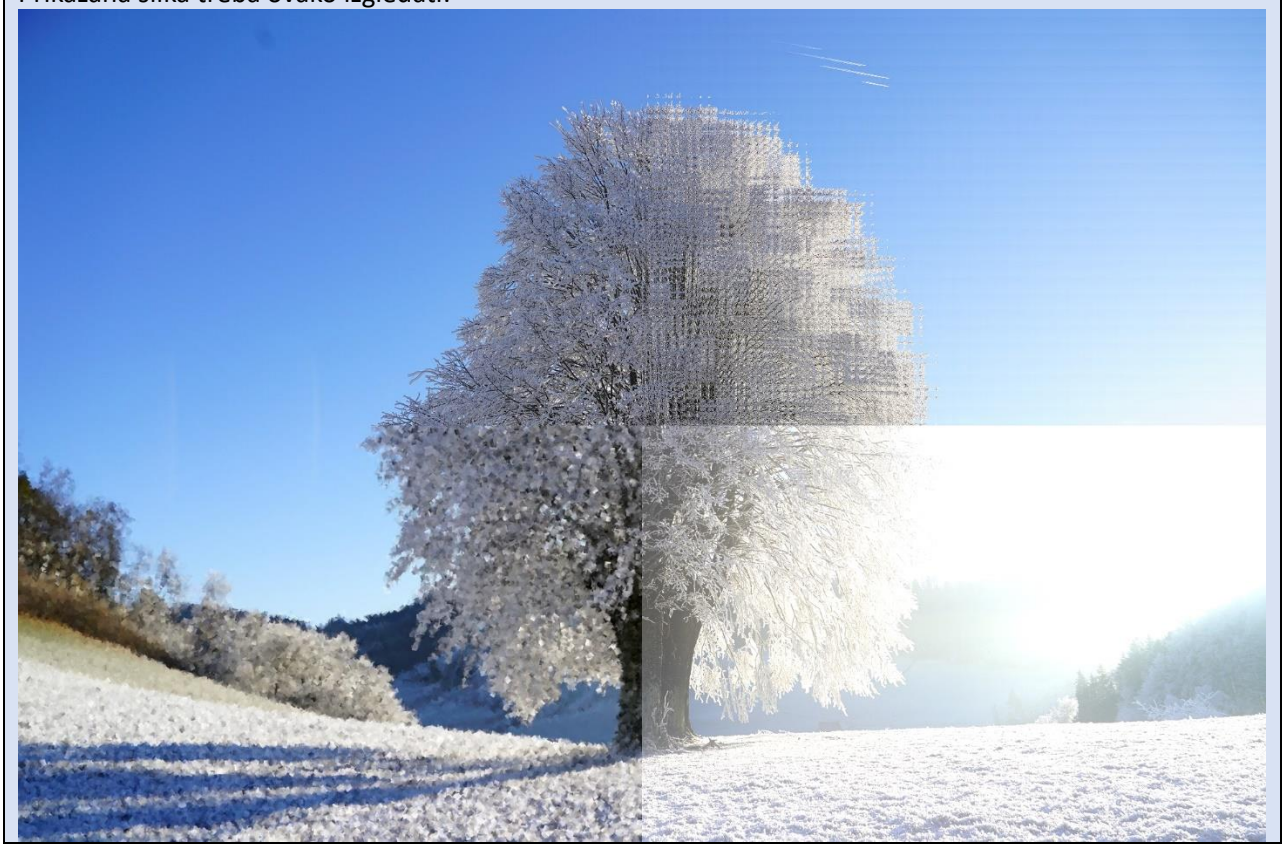
```
# najdalji
snoopy_najdalji = snoopy.resize(size=(
    int(snoopy.size[0] * 0.5),
    int(snoopy.size[1] * 0.5)
))

pos_x = 1000
pos_y = 450
nature.paste(snoopy_najdalji, box=(pos_x, pos_y, pos_x + snoopy_najdalji.size[0],
pos_y + snoopy_najdalji.size[1]))

IPython.display.display(nature)
```

Zadatak 15

Napišite program koji prikazuje sliku /home/gdjambic/images/tree.png, ali ovako:

- Gornja desna četvrtina dolazi sa slike /home/gdjambic/images/tree_glass.png
- Donja lijeva četvrtina dolazi sa slike /home/gdjambic/images/tree_cubism.png
- Donja desna četvrtina dolazi sa slike /home/gdjambic/images/tree_bloom.png

Prikazana slika treba ovako izgledati:



Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

tree = PIL.Image.open("/home/gdjambic/images/tree.png")

pola_sirine = int(tree.size[0] / 2)
pola_visine = int(tree.size[1] / 2)

# odrežemo gornju desnu četvrtinu tree_glass.png i zalijepimo je
tree_glass = PIL.Image.open("/home/gdjambic/images/tree_glass.png")
box = (pola_sirine, 0, 2 * pola_sirine, pola_visine)
tree_glass_dio = tree_glass.crop(box=box)
tree.paste(tree_glass_dio, box=box)

# odrežemo donju lijevu četvrtinu tree_cubism.png i zalijepimo je
tree_cubism = PIL.Image.open("/home/gdjambic/images/tree_cubism.png")
box = (0, pola_visine, pola_sirine, 2 * pola_visine)
```

```
tree_cubism_dio = tree_cubism.crop(box=box)
tree.paste(tree_cubism_dio, box=box)

# odrežemo donju desnu četvrtinu tree_bloom.png i zalijepimo je
tree_bloom = PIL.Image.open("/home/gdjambic/images/tree_bloom.png")
box = (pola_sirine, pola_visine, 2 * pola_sirine, 2 * pola_visine)
tree_bloom_dio = tree_bloom.crop(box=box)
tree.paste(tree_bloom_dio, box=box)

IPython.display.display(tree)
```

Zadatak 16

Napišite program koji prikazuje sliku /home/gdjambic/images/tree.png, ali tako da svaki drugi stupac širine 50 piksela prikažete sa slike/home/gdjambic/images/tree_bloom.png. Prikazana slika treba ovako izgledati:

Moguće rješenje:

```
import PIL.Image
import IPython.display

tree = PIL.Image.open("/home/gdjambic/images/tree.png")
tree_bloom = PIL.Image.open("/home/gdjambic/images/tree_bloom.png")

sirina_slike = tree.size[0]
visina_slike = tree.size[1]
sirina_odsjecka = 50
visina_odsjecka = visina_slike
trenutni_x = 50

while trenutni_x + sirina_odsjecka <= sirina_slike: # pazimo da ne odemo izvan slike
    box_odsjecka = (trenutni_x, 0, trenutni_x + sirina_odsjecka, visina_slike)
    odsjecak = tree_bloom.crop(box=box_odsjecka)
    tree.paste(odsjecak, box=box_odsjecka)

    trenutni_x += 100

IPython.display.display(tree)
```

Zadatak 17

Napišite program koji prikazuje sliku /home/gdjambic/images/tree.png, ali tako da je izrežete na stupce širine 100 piksela i posložite ih u slučajni redoslijed. Dobivena slika može ovako izgledati:

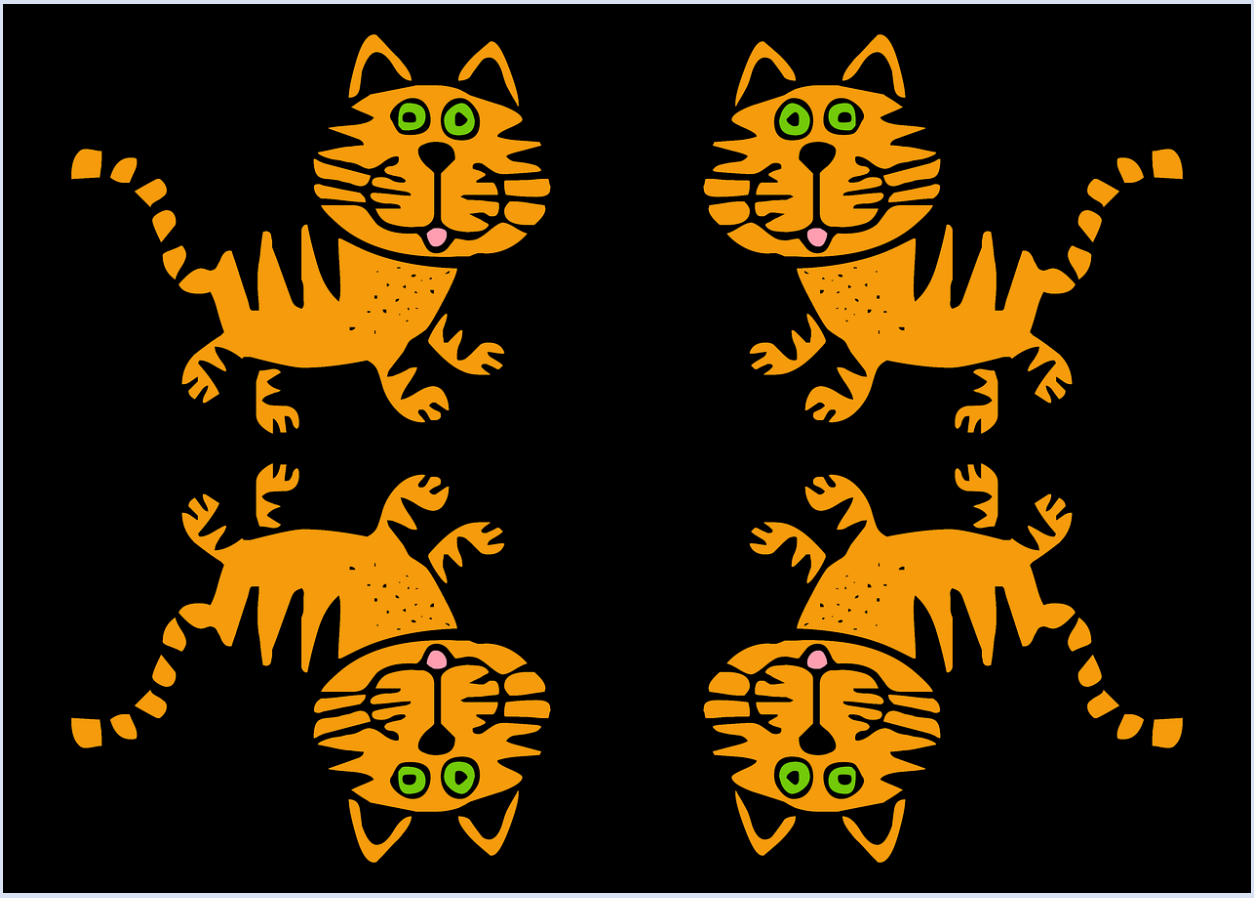
Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display
import random

tree = PIL.Image.open("/home/gdjambic/images/tree.png")

sirina_slike = tree.size[0]
visina_slike = tree.size[1]
sirina_odsjecka = 100
visina_odsjecka = visina_slike
trenutni_x = 0

# prvo izrežemo sliku na stupce
stupci = []
while trenutni_x + sirina_odsjecka <= sirina_slike: # pazimo da ne odemo izvan slike
    box_odsjecka = (trenutni_x, 0, trenutni_x + sirina_odsjecka, visina_slike)
    odsjecak = tree_bloom.crop(box=box_odsjecka)
    stupci.append(odsjecak)

    trenutni_x += sirina_odsjecka
```

```
# sad lijepimo jedan po jedan slučajno odabrani stupac i mičemo ga iz liste
kolicina = len(stupci)
trenutni_x = 0

for i in range(kolicina):
    # odaberemo slučajni stupac
    slucajni_indeks = random.randint(0, len(stupci) - 1)
    slucajni_stupac = stupci[slucajni_indeks]

    # zalijepiom ga na sliku
    tree.paste(slucajni_stupac, box=(trenutni_x, 0, trenutni_x + sirina_odsjecka,
visina_odsjecka))

    # pa ga uklonimo iz liste jer smo ga upravo iskoristili
    del stupci[slucajni_indeks]

    trenutni_x += sirina_odsjecka

IPython.display.display(tree)
```

Zadatak 18

Koristeći slike /home/gdjambic/images/empty.jpg i /home/gdjambic/images/cat.png, kreirajte i prikažite ovakvu sliku (dimenzije slika su tako podešene da na empty.jpg savršeno stanu četiri slike cat.png):



Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display
import random

empty = PIL.Image.open("/home/gdjambic/images/empty.jpg")
sirina = empty.size[0]
visina = empty.size[1]
pola_sirine = int(sirina / 2)
pola_visine = int(visina / 2)

cat_right = PIL.Image.open("/home/gdjambic/images/cat.png")
cat_left = cat_right.transpose(PIL.Image.FLIP_LEFT_RIGHT)
cat_right_tp = cat_right.transpose(PIL.Image.FLIP_TOP_BOTTOM)
cat_left_tp = cat_left.transpose(PIL.Image.FLIP_TOP_BOTTOM)

empty.paste(cat_right, box=(0, 0, pola_sirine, pola_visine))
empty.paste(cat_left, box=(pola_sirine, 0, sirina, pola_visine))
empty.paste(cat_right_tp, box=(0, pola_visine, pola_sirine, visina))
```

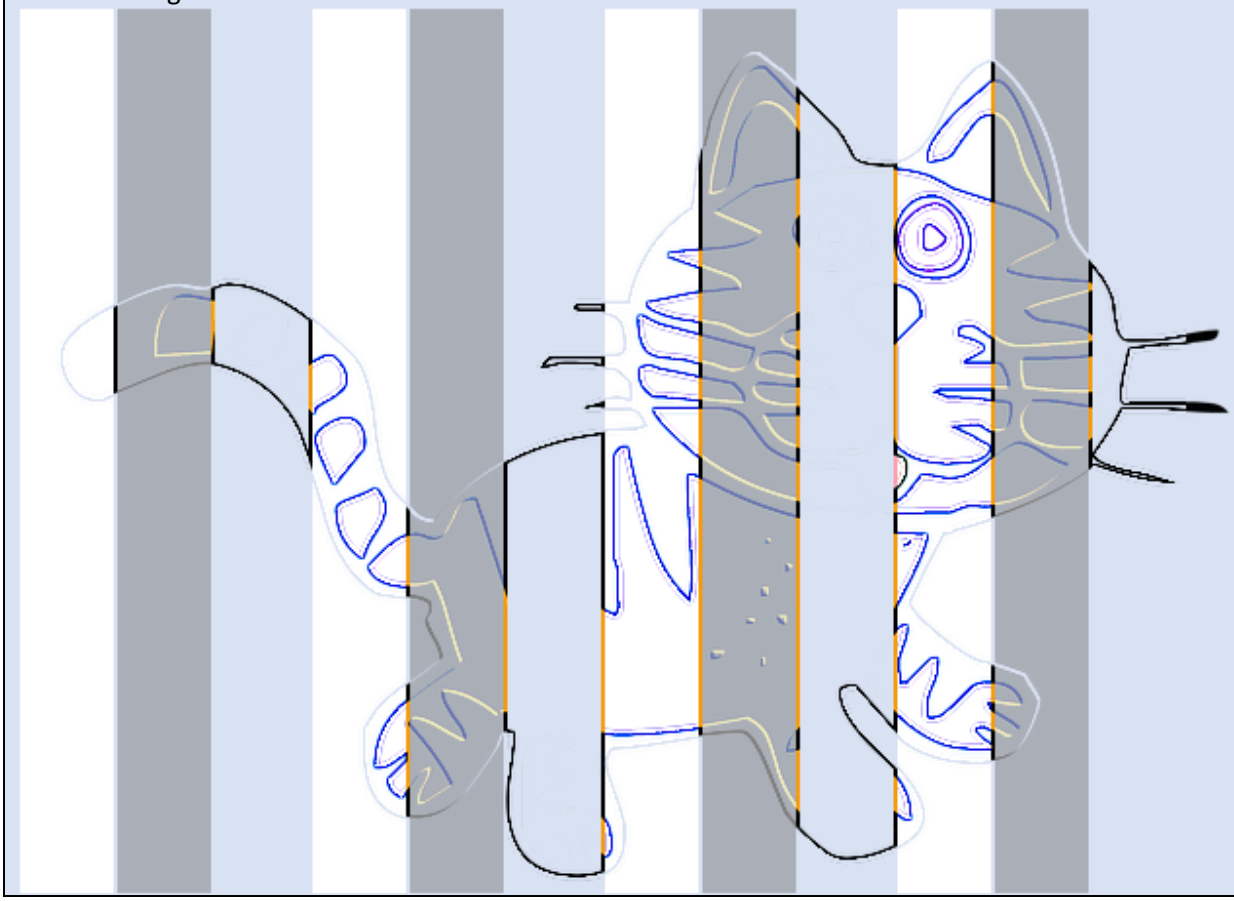
```
empty.paste(cat_left_tp, box=(pola_sirine, pola_visine, sirina, visina))

IPython.display.display(empty)
```

Zadatak 19

Napišite program koji prikazuje sliku /home/gdjambic/images/cat.png na sljedeći način:

- Prvom stupcu širine 50 piksela treba istaknuti konture
- U drugom stupcu širine 50 piksela treba prikazati reljef
- Trećem stupcu širine 50 piksela treba istaknuti rubove
- Nakon toga, ponavljajte isti redoslijed za sve preostale stupce

Slika treba izgledati ovako:



Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

def daj_sljedeci_filter(filter):
    if filter == 1:
        return PIL.ImageFilter.CONTOUR
    elif filter == 2:
        return PIL.ImageFilter.EMBOSS
    elif filter == 3:
        return PIL.ImageFilter.FIND_EDGES

cat = PIL.Image.open("/home/gdjambic/images/cat.png")
```

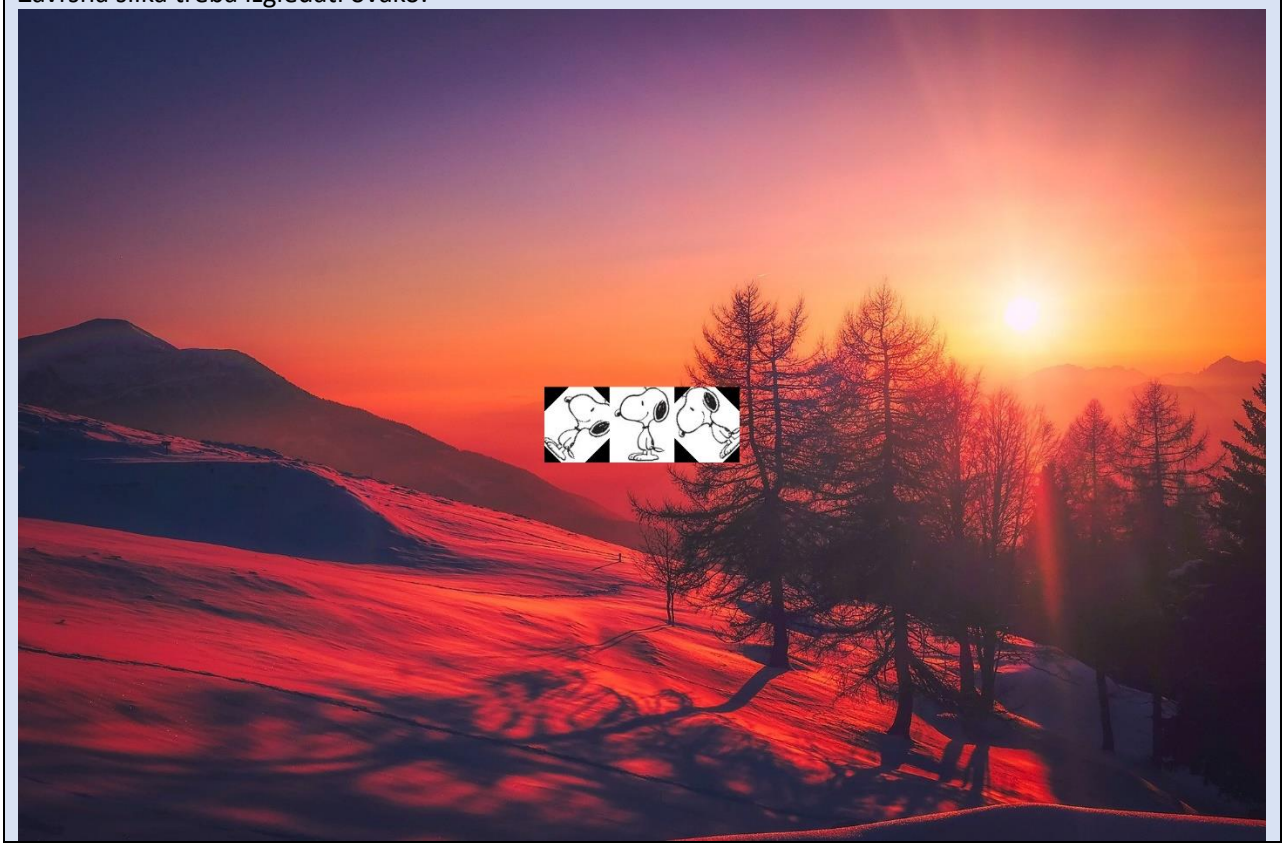
```
sirina_slike = cat.size[0]
visina_slike = cat.size[1]
sirina_odsjecka = 50
visina_odsjecka = visina_slike
trenutni_x = 0
broj_filtera = 1


while trenutni_x + sirina_odsjecka <= sirina_slike: # pazimo da ne odemo izvan slike
    # odrežemo stupac
    box_odsjecka = (trenutni_x, 0, trenutni_x + sirina_odsjecka, visina_slike)
    odsjecak = cat.crop(box=box_odsjecka)

    # primijenimo sljedeći filter
    odsjecak = odsjecak.filter(daj_sljedeci_filter(broj_filtera))
    cat.paste(odsjecak, box=box_odsjecka)

    trenutni_x += sirina_odsjecka
    broj_filtera += 1
    if broj_filtera == 4:
        broj_filtera = 1

IPython.display.display(cat)
```

Zadatak 20

Napišite program koji prikazuje sliku /home/gdjambic/images/snow_morning.jpg sa dodane tri slike /home/gdjambic/images/snoopy.jpg na sljedeći način:

- Originalna slika Snoopyja se treba staviti točno u sredinu snježne slike.
- Slika Snoopyja rotirana za 45 stupnjeva u smjeru kazaljke na satu treba biti odmah lijevo od središnje.
- Slika Snoopyja rotirana za 45 stupnjeva u smjeru suprotnom kazaljke na satu treba biti odmah desno od središnje.

Završna slika treba izgledati ovako:



Moguće rješenje:

```
import PIL.Image
import PIL.ImageFilter
import IPython.display

snoopy = PIL.Image.open("/home/gdjambic/images/snoopy.jpg")
snoopy_sirina = snoopy.size[0]
snoopy_visina = snoopy.size[1]
snoopy_pola_sirine = int(snoopy.size[0] / 2)
snoopy_pola_visine = int(snoopy.size[1] / 2)

snow_morning = PIL.Image.open("/home/gdjambic/images/snow_morning.jpg")
sredina_x = int(snow_morning.size[0] / 2)
sredina_y = int(snow_morning.size[1] / 2)

# originalni snoopy u sredini slike
```

```
snow_morning.paste(snoopy, box=(
    sredina_x - snoopy_pola_sirine,
    sredina_y - snoopy_pola_visine,
    sredina_x + snoopy_pola_sirine,
    sredina_y + snoopy_pola_visine
))

# rotirani snoopy lijevo od sredine slike
snoopy_rot1 = snoopy.rotate(-45)
snow_morning.paste(snoopy_rot1, box=(
    sredina_x - 3 * snoopy_pola_sirine,
    sredina_y - snoopy_pola_visine,
    sredina_x - snoopy_pola_sirine,
    sredina_y + snoopy_pola_visine
))

# rotirani snoopy desno od sredine slike
snoopy_rot2 = snoopy.rotate(45)
snow_morning.paste(snoopy_rot2, box=(
    sredina_x + snoopy_pola_sirine,
    sredina_y - snoopy_pola_visine,
    sredina_x + 3 * snoopy_pola_sirine,
    sredina_y + snoopy_pola_visine
))

IPython.display.display(snow_morning)
```