1MPR10_Simona_Bļinova sb24037

1.uzdevums

Programma, kas realizē eksāmena simulāciju.

Kods:

```
import random
import string

studenti = set()
biletes = set()

for i in range(1, 57):
  studenti.add(i)

for i in range(65, 91):
  biletes.add(chr(i))

#print(studenti)
#print(biletes)

studentu_atbildes = []

for i in range(10):
  while True:
    st = random.randint(1, 56)
    b = random.choice(string.ascii_uppercase)
    #print(st)
    #print(chr(b))
    if b in biletes and st in studenti:
      studenti.remove(st)
      biletes.remove(b)
      break

  #print(studenti)
  #print(biletes)
  print(f'Students: {st}, biļete: {b}')

  studentu_atbildes.append((st, b))

#print(studentu_atbildes)
```

```
while len(studenti) > 0:
    #print(studentu_atbildes)
    a = studentu_atbildes[0]
    studentu_atbildes.pop(0)

    biletes.add(a[1])

    while True:
        st = random.randint(1, 56)
        b = random.choice(string.ascii_uppercase)
        #print(st)
        #print(chr(b))
        if b in biletes and st in studenti:
            studenti.remove(st)
            biletes.remove(b)
            break

    print(f'Students: {st}, biļete: {b}')
    studentu_atbildes.append((st, b))
```

Testa piemērs(1)

```
Students: 56, biļete: M
Students: 46, biļete: G
Students: 35, biļete: O
Students: 33, biļete: N
Students: 1, biļete: D
Students: 31, biļete: L
Students: 22, biļete: T
Students: 3, biļete: X
Students: 28, biļete: R
Students: 8, biļete: A
Students: 18, biļete: F
Students: 23, biļete: J
Students: 7, biļete: B
Students: 53, biļete: N
Students: 2, biļete: H
Students: 52, biļete: O
Students: 5, biļete: T
Students: 36, biļete: Q
Students: 10, biļete: X
Students: 44, biļete: U
Students: 55, biļete: D
Students: 30, biļete: S
Students: 14, biļete: B
Students: 40, biļete: E
Students: 26, biļete: K
Students: 12, biļete: R
Students: 29, biļete: A
Students: 48, biļete: N
```

Testa piemērs(2)

```
Students: 21, biļete: D
Students: 13, biļete: I
Students: 49, biļete: P
Students: 9, biļete: L
Students: 18, biļete: R
Students: 43, biļete: O
Students: 40, biļete: F
Students: 15, biļete: W
Students: 30, biļete: A
Students: 23, biļete: B
Students: 5, biļete: V
Students: 1, biļete: S
Students: 39, biļete: Q
Students: 25, biļete: C
Students: 56, biļete: P
Students: 19, biļete: G
Students: 35, biļete: Y
Students: 16, biļete: X
Students: 36, biļete: L
Students: 48, biļete: O
Students: 50, biļete: U
Students: 31, biļete: F
Students: 32, biļete: S
Students: 24, biļete: V
Students: 52, biļete: Q
Students: 10, biļete: T
Students: 34, biļete: G
```

Testa piemērs(3)

```
Students: 9, biļete: G
Students: 34, biļete: H
Students: 43, biļete: B
Students: 14, biļete: L
Students: 42, biļete: S
Students: 2, biļete: E
Students: 1, biļete: N
Students: 12, biļete: M
Students: 46, biļete: Z
Students: 49, biļete: I
Students: 37, biļete: P
Students: 38, biļete: H
Students: 10, biļete: D
Students: 5, biļete: A
Students: 41, biļete: T
Students: 16, biļete: K
Students: 36, biļete: U
Students: 7, biļete: G
Students: 11, biļete: C
Students: 40, biļete: M
Students: 47, biļete: X
Students: 18, biļete: E
Students: 52, biļete: H
Students: 54, biļete: V
Students: 53, biļete: R
Students: 27, biļete: A
Students: 8, biļete: U
Students: 3, biļete: D
Students: 25, biļete: K
Students: 45, biļete: M
Students: 44, biļete: S
```

2.uzdevums

Programma, kas izveido klasi Taisnsturis un izvada objekta perimetru un laukumu.

Kods:

```
class Taisnsturis():
  '''
  Klase 'Taisnsturis' pārstāv vienkāršu taisnstūra modeli ar pamatinformāciju:
  garums, platums

  Šī klase nodrošina metodes laukuma, perimetra aprēķiniem un informācijas izvadei.

  Lauki:
    garums (int): Taisntūra garums
    platums (int): Taisnstūra platums

  Metodes:
    perimetrs()
    laukums()
    __str__()
  '''

  def __init__(self, garums, platums):
    '''
    Konstruktors (inicializators).
    Piešķir sākotnējās vērtības laukiem.

    Argumenti:
      garums (int): Taisntūra garums
      platums (int): Taisnstūra platums
    '''

    self.garums = garums
    self.platums = platums

  def perimetrs(self):
    '''
    Metode.
    Aprēķina objekta perimetru.
    '''
```

```
        return f'Perimetrs: {(self.garums + self.platums) * 2}'

    def laukums(self):
        '''
        Metode.
        Aprēķina objekta laukumu.
        '''

        return f'Laukums: {self.garums * self.platums}'

    def __str__(self):
        '''
        Metode.
        Izvada informāciju par objektu.
        '''

        return f'Taisnstūris {self.garums} x {self.platums}'


def galvena_programma():
    x = int(input('Ievaidet taisnstūra garumu --> '))
    y = int(input('Ievaidet taisnstūra platumu --> '))
    z = Taisnsturis(x, y)
    print(z)
    print(z.perimetrs())
    print(z.laukums())


if __name__ == '__main__':
    galvena_programma()
    #mans_taisnsturis = Taisnsturis(4, 4)
    ##mans_taisnsturis2 = Taisnsturis(6, 2)

    #print(mans_taisnsturis)

    #print(mans_taisnsturis.perimetrs())
    #print(mans_taisnsturis.laukums())
```

Testa piemērs(1)

```
Ievaidet taisnstūra garumu --> 3
Ievaidet taisnstūra platumu --> 4
Taisnstūris 3 x 4
Perimetrs: 14
Laukums: 12
```

Testa piemērs(2)

```
Ievaidet taisnstūra garumu --> 1
Ievaidet taisnstūra platumu --> 6
Taisnstūris 1 x 6
Perimetrs: 14
Laukums: 6
```

Testa piemērs(3)

```
Ievaidet taisnstūra garumu --> 7
Ievaidet taisnstūra platumu --> 3
Taisnstūris 7 x 3
Perimetrs: 20
Laukums: 21
```

3.uzdevums

Programma, kas izveido klasi Trijsturis ar noteiktam metodēm.

Kods:

```
import math

class Trijsturis:
    '''
    Klase 'Trijsturis' pārstāv vienkāršu trijstūra modeli ar pamatinformāciju:
    mala1, mala2, mala3

    Šī klase nodrošina metodes laukuma, perimetra, apvilktas un ievilktas riņķa līnijas radiusa aprēķiniem un trijstūra malu garumu izvadi.

    Lauki:
        mala1 (int): 1. trijstūra mala
        mala2 (int): 2. trijstūra mala
        mala3 (int): 3. trijstūra mala

    Metodes:
        perimetrs()
        laukums()
        ievilktas_rinka_linijas_radiuss()
        apvilktas_rinka_linijas_radiuss()
```

```
    __str__()
'''

def __init__(self, mala1, mala2, mala3):
    '''
    Konstruktors (inicializators).
    Piešķir sākotnējās vērtības laukiem.

    Argumenti:
        mala1 (int): 1. trijstūra mala
        mala2 (int): 2. trijstūra mala
        mala3 (int): 3. trijstūra mala
    '''

    self.mala1 = mala1
    self.mala2 = mala2
    self.mala3 = mala3

def perimetrs(self):
    '''
    Metode.
    Aprēķina objekta perimetru.
    '''

    return self.mala1 + self.mala2 + self.mala3

def laukums(self):
    '''
    Metode.
    Aprēķina objekta laukumu.
    '''

    p = self.perimetrs()
    #print(p)
    pusp = p / 2
    return round(math.sqrt(pusp * (pusp - self.mala1) * (pusp - self.mala2) * (pusp - self.mala3)), 2)

def ievilktas_rinka_linijas_radiuss(self):
    '''
    Metode.
    Aprēķina objekta ievilktas rinka linijas radiusu.
```

```
        '''

        s = self.laukums()
        p = self.perimetrs()
        return round(2 * s / p, 2)

    def apvilktas_rinka_linijas_radiuss(self):
        '''
        Metode.
        Aprēķina objekta apvilktas rinka linijas radiusu.
        '''

        s = self.laukums()
        return round(self.mala1 * self.mala2 * self.mala3 / 4 / s, 2)

    def __str__(self):
        '''
        Metode.
        Izvada visu sākotnējo informāciju par objektu.
        '''
        return f'Trijstūris {self.mala1}x{self.mala2}x{self.mala3}'


def galvena_programma():
    m1 = int(input('Ievaidet trijstūra 1. malu --> '))
    m2 = int(input('Ievaidet trijstūra 2. malu --> '))
    m3 = int(input('Ievadiet trijstūra 3. malu --> '))

    if m1+m2>m3 and m1+m3>m2 and m2+m3>m2:
        trijsturis = Trijsturis(m1, m2, m3)
        print(f'Perimetrs: {trijsturis.perimetrs()}')
        print(f'Laukums: {trijsturis.laukums()}')
        print(f'Ievilktas riņķa līnijas laukums: {trijsturis.ievilktas_rinka_linijas_radiuss()}')
        print(f'Apvilktas riņķa līnijas laukums: {trijsturis.apvilktas_rinka_linijas_radiuss()}')
    else:
        print('Tadas malas trijstūrim nav iespējamas!')
        exit()


if __name__ == '__main__':
    galvena_programma()
```

Testa piemērs(1)

```
Ievaidet trijstūra 1. malu --> 13
Ievaidet trijstūra 2. malu --> 14
Ievadiet trijstūra 3. malu --> 15
Perimetrs: 42
Laukums: 84.0
Ievilktas riņķa līnijas laukums: 4.0
Apvilktas riņķa līnijas laukums: 8.12
```

Testa piemērs(2)

```
Ievaidet trijstūra 1. malu --> 2
Ievaidet trijstūra 2. malu --> 5
Ievadiet trijstūra 3. malu --> 4
Perimetrs: 11
Laukums: 3.8
Ievilktas riņķa līnijas laukums: 0.69
Apvilktas riņķa līnijas laukums: 2.63
```

Testa piemērs(3)

```
Ievaidet trijstūra 1. malu --> 7
Ievaidet trijstūra 2. malu --> 9
Ievadiet trijstūra 3. malu --> 11
Perimetrs: 27
Laukums: 31.42
Ievilktas riņķa līnijas laukums: 2.33
Apvilktas riņķa līnijas laukums: 5.51
```