(Demo) Tentamen Programmeringsteknik I (Python) Datum
Lärare: Johan Öfverstedt

Skrivtid: 08:00 - 13:00

Tänk på följande:

- Skriv läsligt. Använd inte rödpenna.
- Skriv bara på framsidan av varje papper.
- Lägg uppgifterna i ordning. Skriv uppgiftsnummer (gäller B-delen) och din kod överst i högra hörnet på alla papper.
- Fyll i försättsbladet ordentligt.
- Såvida inget annat anges, både får och ska man bygga på lösningar från föregående uppgifter även om dessa inte har lösts.
- På B-delen är det tillåtet att införa hjälpfunktioner/hjälpmetoder och hjälpklasser. Uttrycket skriv en funktion/metod betyder alltså inte att lösningen inte får struktureras med hjälp av flera metoder.
- Du behöver inte skriva import-satser för att använda funktioner och klasser från standardbiblioteken.
- Alla uppgifter gäller programmeringsspråket Python och programkod skall skrivas i korrekt Python. Koden ska vara läslig med lämpliga variabelnamn, korta men beskrivande namn på funktioner, klasser och metoder.

  Observera att betyget påverkas negativt av
  - icke-privata variabler och onödiga variabler,
  - dålig läslighet,
  - upprepning av identisk kod,
  - underlåtenhet att utnyttja given/egen skriven kod.

Skrivningen består av två delar. Lösningarna till uppgifterna på A-delen ska skrivas in i de tomma rutorna och den delen ska lämnas in. Rutorna är tilltagna i storlek så att de ska rymma svaren. En stor ruta betyder inte att svaret måste vara stort! Lösningarna till uppgifterna på B-delen skrivs på lösa papper. För att bli godkänd (betyg 3) krävs att minst ca 75% av A-delen är i stort sett rätt löst. För betyget 4 krävs dessutom att minst hälften av uppgifterna på B-delen och betyg 5 att alla uppgifterna på B-delen är i stort sett rätt lösta. Vid bedömning av betyg 4 och 5 tas också hänsyn tillkvalitén på lösningarna i A-delen. Observera att B-delen inte rättas om inte A-delen är godkänd.

Lycka till!

Del A

A.1.

A)
`a = [(3, 'Johan'), (5, 'Erika'), (1, 'Gabriel')]`
`b = sorted(a)`
`c = [v for k, v in b]`
`print(c)`
	Vad händer?
	1) [Johan, Erika, Gabriel] skrivs ut
	2) [Gabriel, Erika, Johan] skrivs ut
	3) [Gabriel, Johan, Erika] skrivs ut
	4) [1, 3, 5] skrivs ut

B)
`def increment(x):`
`  x = x + 1`
`  return x`
`x = 1`
`y = increment(x)`
`print(x, y)`
	Vad händer?
	1) 2 2 skrivs ut
	2) 2 1 skrivs ut
	3) 1 2 skrivs ut
	4) 1 1 skrivs ut
	5) Det blir ett ValueError

C)
`for i in range(5, 0, -1):`
`  print(i, end=' ')`
	1) 5, 4, 3, 2, 1 skrivs ut
	2) 0, 1, 2, 3, 4 skrivs ut
	3) 5, 4, 3, 2, 1, 0 skrivs ut
	4) 0, 1, 2, 3, 4, 5 skrivs ut

D)
`def f(x):`
`  return x * 2`
`y = f(2)`
 	
Vad har variabeln `x` för datatyp?
	1) int
	2) float
	3) str
	4) list
	5) x har ingen bestämd datatyp

E) Vad är ett annat namn för `__init__`-metoden i klasser?	1) Generator 2) Main-metod 3) Konstruktor 4) Skapare

F)
`def f(x, y):`
`  return x + y`
Vad är `x` och `y`?
	1) Argument
	2) Faktorer
	3) Attribut
	4) Parametrar

G) print([1, 2] + ([3, 4]*2))	Vad skrivs ut? 1) [1, 2, 3, 4, 3, 4] 2) [7, 10] 3) [1, 2, 6, 8] 4) Det blir fel
H) Vad kan man säga säkert om s? s = Student('Johan')	1) s är en referens till en instans av klassen Student 2) Vi vet ej vad s är av för datatyp 3) Det blir fel då vi måste anropa Student.__init__('Johan') 4) print(s.name) kommer ge utskriften 'Johan'

A.2. Skriv en funktion `pow` som givet två positiva heltal `x, y` beräknar x^y, `x` upphöjt till `y`, med en loop och multiplikation [`x*x*...*x` (`y` gånger).].

```
def pow(x, y):
    z = 1.0
    for i in range(y):
        z = z * x
    return z
```

A.3. Skriv en funktion `kth_element(lst, k)` som givet en lista returnerar det `k = [0, len(lst))` minsta elementet. Exempel:

```
print(kth_element([1, 2, 5, 4, 3], 2))
print(kth_element([1, 2, 5, 4, 3], 1))
```

Ger utskrift:

```
3
2
```

```
def kth_element(lst, k):
    lst_sorted = sorted(lst)
    return lst_sorted[k]
```

A.4. Skriv en funktion `less(lst, a)` som returnerar en lista med index för alla element i `lst` som är mindre än `a`. Exempel:

```
print(less([1, 2, 6, 4, 0, 7, 9, 3, 2], 3))
```

Ger utskrift:

```
[0, 1, 4, 8]
```

```
def less(lst, a):
    result = []
    for i, x in enumerate(lst):
        if x < a:
            result.append(i)
    return result
```

eller

```
    return [i for i, x in enumerate(lst) if x < a]
```

A.5. Skriv en klass Person som representerar ett namn och e-postadress. Klassen ska innehålla en metod `print_person` (utöver konstruktorn) som skriver ut namnet och e-postadressen på en rad.

Skriv även kod för att skapa ett person-objekt med namnet 'Jan Karlsson' och e-postadress 'jan.karlsson@gmail.com'.

```
class Person:
    def __init__(self, name, address):
        self.name = name
        self.address = address

    def print_person(self):
        print(f'{self.name} (self.address)')

p = Person('Jan Karlsson', 'jan.karlsson@gmail.com')
```

A.6. Skriv en funktion `word_freq(s)` som skapar och returnerar ett lexikon med ordfrekvenser för strängen s. Exempel:
print(word_freq('hej hej monica hej på dig monica'))
ger utskriften:
{'hej': 3, 'på': 1, 'dig': 1, 'monica': 2}

```
def word_freq(s):
    words = s.split()
    res = {}
    for w in words:
        if w in res:
            res[w] += 1
        else:
            res[w] = 1
    return res
```

A.7. Skriv en funktion `random_value(lst)` som givet en lista med element returnerar ett slumpmässigt element från listan och kod som anropar och skriver ut returvärdet från metoden för listan ['a', 'b', 'c'] (random.random() ger ett slumpmässigt flyttal mellan 0.0 och 1.0).

```
def random_value(lst):
    r = int(random.random() * len(lst))
    return lst[r]
```

A.8.

Givet två geometriska punkter A och B som representeras av tupler/par av två flyttal, skriv en funktion `euclidean_distance(a, b)` som beräknar det Euklidiska avståndet mellan dem, vilket ges av formeln:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Kvadratroten kan beräknas med math.sqrt(...).

```
def euclidean_distance(a, b):
    return math.sqrt((a[0]-b[0])**2+(a[1]-b[1])**2)
```

Del B

I denna del ska alla svar skrivas på ett löst papper. Kom ihåg att skriva tentamenskod på varje blad.

Följande program (i filen `game.py`) är givet, vilket implementerar ett litet Tic-Tac-Toe-spel.

```
import board as brd

def main():
	sz = 3
	b = brd.Board(sz)

	player = 0
	symbols = ['X', 'O']
	while b.count() < sz*sz:
		print('--------------------')
		print(b)
		print(f'Player {symbols[player]}\'s turn')
		x = int(input('x: '))
		y = int(input('y: '))
		try:
			b.place(x, y, symbols[player])
			w = b.check_for_winner()
			if w is not None:
				print(b)
				print(f'The winner is player {w}.')
				return
			player = 1 - player
		except ValueError as ve:
			print(ve)
	print(b)
	print('It is a tie. No winner.')

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

Testutskrift vid en provkörning:

```
--------------------
...
...
...
```

```
--------------------
XOX
.O.
.X.
```

```
Player X's turn
x: 0
y: 0
--------------------
X..
...
...

Player O's turn
x: 1
y: 1
--------------------
X..
.O.
...

Player X's turn
x: 2
y: 0
--------------------
X.X
.O.
...

Player O's turn
x: 1
y: 0
--------------------
XOX
.O.
...
Player X's turn
x: 1
y: 2
```

```
Player O's turn
x: 0
y: 1
--------------------
XOX
OO.
.X.

Player X's turn
x: 2
y: 1
--------------------
XOX
OOX
.X.

Player O's turn
x: 2
y: 2
--------------------
XOX
OOX
.XO

Player X's turn
x: 0
y: 2
XOX
OOX
XXO

It is a tie. No winner.
```

B.1. Definiera klassen `Board` och skriv en konstruktor som givet en spelplansstorlek `n` (heltal) skapar en nästlad lista `n x n` (bestående av en lista med rader, som i sin tur består av kolumner med pjäser) som enda attribut, initialt fylld med tomma element (antingen `None` eller `'.'`).

B.2. Skriv en metod `count(self)` som räknar hur många platser har fyllts med en pjäs.

B.3. Skriv metoden `place(self, x, y, symbol)` i klassen `Board` som placerar den givna symbolen på plats `x` och `y`. Om `x` eller `y` är utanför spelplanen, eller om symbolen är en annan än `'X'` eller `'O'` ska ett fel signaleras (`assert` fungerar bra för detta ändamål). Om platsen redan är upptagen ska ett ValueError ges med ett lämpligt felmeddelande på formen:

```
Illegal move at ({x}, {y}).
```

B.4. Skriv metoden __str__(self) i klassen Board som returnerar en sträng som representerar spelplanens status i ett rutnät, exempelvis:

```
.X.
OX.
XO.
```

Tips: '\n' är representationen för radbrytning.

B.5. Antag att du har en metod (i klassen `Board`), `get_configurations(self)`, som returnerar alla tänkbara *`n`-konfigurationer* där `n` är spelplanens storlek, i form av en lista med strängar, som kan innebära att spelet har vunnits av någon av spelarna.

Exempelvis för spelplanen i B.4:
`['.X.', 'OX.', 'XO.', '.OX', 'XXO', '...', '.X.', '.XX']`

Skriv en metod `check_for_winner(self)` som testar om någon av spelarna har vunnit spelet (om vi har lika många i rad som storleken på spelplanen, utan att antaga att den är 3 stor).

Om spelare `'X'` har vunnit, returnera `'X'`, om spelare `'O'` har vunnit, returnera `'O'`, och om ingen har vunnit, returnera `None`.

Lösningar B-delen (varning för fel med indenteringen):

```
class Board:
      def __init__(self, sz):
          self.b = []
          self.sz = sz
          for _ in range(sz):
            row = []
            for _ in range(sz):
            row.append('.')
            self.b.append(row)

      def place(self, x, y, symbol):
          assert(x >= 0 and x < self.sz)
          assert(y >= 0 and y < self.sz)
          assert(symbol == 'X' or symbol == 'O')
          if self.b[y][x] != '.':
                raise ValueError(f'Illegal move at ({x}, {y})')
          self.b[y][x] = symbol

      def check_for_winner(self):
          config = self.get_configurations()
          x_win = ''.join(['X']*self.sz)
          o_win = ''.join(['O']*self.sz)
          for i in range(len(config)):
              if config[i] == x_win:
            return 'X'
              elif config[i] == o_win:
            return 'O'
          return None

      def count(self):
          c = 0
          for i in range(self.sz):
            for j in range(self.sz):
                if self.b[i][j] != '.':
                c = c + 1
      return c

      def __str__(self):
      s = []
      for r in range(self.sz):
            for c in range(self.sz):
            s.append(self.b[r][c])
            s.append('\n')
      return ''.join(s)
```

```
        ### Givna funktioner
        def get_row_str(self):
            res = []
            for r in range(self.sz):
                res.append(''.join(self.b[r]))
            return res

        def get_col_str(self):
            res = []
            for c in range(self.sz):
              res_inner = []
              for r in range(self.sz):
              res_inner.append(self.b[r][c])
              res.append(''.join(res_inner))
            return res

        def get_diags_str(self):
            res1 = []
            res2 = []
            for it in range(self.sz):
              res1.append(self.b[it][it])
              res2.append(self.b[self.sz-it-1][it])
            return [''.join(res1), ''.join(res2)]

        def get_configurations(self):
            return self.get_row_str() + self.get_col_str() +
self.get_diags_str()
```

Referensblad:

Listor
len(lst) - Ger listans längd.

Strängar
`s.join(lst)` - sammanfogar en lista till en sträng med s mellan varje element.
Exempel `'-'.join('1', '2', '3')` ger `'1-2-3'`.
`s.count(subs)` - räknar antalet förekomster av strängen `subs` i strängen `s`.
Exempel `'abbabba'.count('bb')` ger `2`, `'abbabba'.count('b')` ger `4`.

Sekvenser
`range(start, stop, step)` - skapar en sekvens med heltal från start till men inte med stop med steglängd step.
Exempel:
`list(range(7, 2, -1))` ger `[7, 6, 5, 4, 3]`.