Omkontrollskrivningskomplettering i DD1331 Grundläggande programmering

1. Skrivtiden är 2024-01-22 klockan 08:00–09:00. Vissa studenter har extra skrivtid.
2. Diskutera inte uppgifterna eller lösningarna och posta inget om dem på sociala medier eller Git förrän klockan 10:15.
3. Facit kommer att publiceras senare under dagen, När detta sker kommer ett anslag på Canvas.
4. Denna kontrollskrivning har ett undantag från regeln med 1 uppgift per papper. Ni som kompletterar en annan del än låd– och pildiagram förväntas lämna in alla svar på ett enda papper.
5. Se bara till att inte använda baksidor(na) på (de) papper som ni lämnar in eftersom scannern endast scannar framsidor.
6. Tillåtna hjälpmedel: En programmeringsbok, skrivmaterial, mat, medicin och vattenflaska.
7. Kontrollskrivningskompletteringen består av 3 delar med 5 frågor i varje del. För godkänt krävs rätt på 3 frågor av 5 på den del som du kompletterar, under förutsättning att du klarade de andra delarna på ordinarie omkontrollskrivning.
8. Om du får godkänt så kommer resultatet direkt i Ladok.
9. Om du får underkänt så kommer det direkt i Ladok och det kommer inte att stå någonting på uppgifterna O1-O6 på Canvas.
10. Det går inte att komplettera kompletteringen. Om du får underkänt nu så är nästa chans ordninarie kontrollskrivning hösten 2024.
11. Alla program och alla exempel går i Python3 oavsett om det står i uppgiften eller inte.

Exempel på formatet för ifyllning ses nedan.

11 Q
12 X Y Z
13 73-74 86-89

Lycka till! /Marcus

Kontrollskrivning i DD1331 Grundläggande Programmering

1 Terminologi

Lärandemål: beskriva källkoden till ett dataprogram med rätt terminologi,

1. Avgör om någon av dessa alternativ är en giltig identifierare (dvs ett giltigt (men inte nödvändigtvis lämpligt) namn på en variabel, klass eller funktion) eller om dess svenska text uttrycker något giltigt om de andra svarsalternativen. Du behöver pricka in alla rätta svar. Minst ett svarsalternativ är giltigt.

   (A) `3little_neutrons`

   (B) `4 8 15 16 23 42`

   (C) `1048576`

   (D) for

   (E) ultimate-tier-champion

   (F) `+`

   (G) `"barn"`

   (H) Alla ovanstående alternativ är giltiga identifierare.

   (I) Inget av ovanstående alternativ.

2. Klassificera denna symbol enligt Python3:s syntax: `:`
   Exakt ett alternativ är rätt. Engelska alternativet syns inom parentes.

   (A) Identifierare (identifier).

   (B) Avgränsare (delimiter).

   (C) Operator (operator).

   (D) Litteral (literal).

   (E) Nyckelord (keyword).

3. Vad är det för skillnad på en identifierare (identifier) och ett nyckelord (keyword) i Python3? Endast ett alternativ är rätt.

   (A) Nyckelord kan inte användas som identifierare.

   (B) Nyckelord kan namnge funktioner och klasser men inte variabler.

   (C) Identifierare visar vilken typ av sats som kommer.

   (D) Variabler som pekas ut av nyckelord kan inte förändras.

   (E) Nyckelord skrivs med bokstäver men identifierare skrivs alltid enbart med andra tecken.

4. Vilket eller vilka av följande alternativ *är* satser (statements) i Python3? Minst ett alternativ är rätt. Observera att de måste *vara* satser för att räknas. Att ingå i en sats räknas inte.

   (A) for-satser.

   (B) if-satser.

   (C) Operatorer.

   (D) while-slingor.

   (E) Alla ovanstående alternativ är satser.

   (F) Inget av ovanstående alternativ är satser.

5. Vilket eller vilka av följande alternativ får ingå i ett uttryck? Minst ett svarsalternativ är rätt.

   (A) for-satser.

   (B) Funktionsanrop.

   (C) if-satser.

   (D) Litteraler (Literals på engelska).

   (E) Opteratorer.

   (F) Variabler.

   (G) Alla ovanstående alternativ får ingå i ett uttryck.

   (H) Inget av ovanstående får ingå i ett uttryck.

2 Datatyper, klasser och typkonverteringar

Lärandemål: beskriva och tillämpa grundläggande datatyper, klasser och typkonverteringar. Ingen kompletterar del 2 denna gång.

3 Felsöka flödeskontroll och uttryck

Lärandemål: beskriva, tillämpa och felsöka flödeskontroll samt logiska och aritmetiska uttryck.

11. Vilka värden på a, b och c gör följande uttryck **sant**? Minst ett alternativ är rätt och du behöver ange alla rätta svar.

```
(a or b) and (b or not c) and (c or not a)
```

(A) a, b, c = True, True, True
(B) a, b, c = True, True, False
(C) a, b, c = True, False, True
(D) a, b, c = True, False, False
(E) a, b, c = False, True, True
(F) a, b, c = False, True, False
(G) a, b, c = False, False, True
(H) a, b, c = False, False, False
(I) Inget av ovanstående alternativ.

12. Vad är felet med detta Python3-program? Det ska skriva ut `5` när användaren matar in det som står på nästa rad:
`2` Minst ett svarsalternativ är rätt.

```
1 n = input()
2 print(3 + n)
```

(A) Indexfel.
(B) Syntaxfel.
(C) Typfel.
(D) Inget av ovanstående alternativ är rätt.

13. Vad skriver koden nedan ut?

```
1 i = 2
2 while i <= 9:
3         i += 3
4 print(i)
```

(A) 2
(B) 5
(C) 8
(D) 9
(E) 11
(F) 14

14. Ange hur många gånger detta program skriver ut ordet på nästa rad:
`Rad`
Endast ett alternativ är rätt.

```
1 for i in range(3,9,2):
2     print("Rad")
```

(A) 0

(B) 1

(C) 2

(D) 3

(E) 4

(F) 5

15. Vad skriver denna kod ut? När print anropas med end="" så görs inget nyradstecken eller mellanslag efter en utskrift.

```
builder = ""
for i in range(4):
    if i % 2:
        builder += "B"
    else:
        builder += "A"
print(builder)
```

(A) `AAAA`
(B) `AAAB`
(C) `AABA`
(D) `AABB`
(E) `ABAA`
(F) `ABAB`
(G) `ABBA`
(H) `ABBB`
(I) `BAAA`
(J) `BAAB`
(K) `BABA`
(L) `BABB`
(M) `BBAA`
(N) `BBAB`
(O) `BBBA`
(P) `BBBB`

4 Räckvidd, livslängd och aliasing

Lärandemål: beskriva en variabels räckvidd och livslängd samt att greppa aliasing. Ingen kompletterar denna del denna gång.

5 Rekursion

Felsöka, och med rätt terminologi beskriva rekursiva algoritmer. Ingen kompletterar denna del denna gång.

6 Låd- och pildiagram

Lärandemål: grafiskt beskriva kopplingen mellan variabelnamn, typer och data.

På var och en av nedanstående uppgifter behöver du rita en bild över minnet vid en viss position under programmets körning. Du behöver ha med:

(i) En låda för globala variablerna längst ner till vänster i diagrammet.

(ii) En stackpost (Engelska: stack frame) per aktivt funktions-/metodanrop staplade i en stack. I stackpostens övre vänstra hörn ska det stå vilken funktion som har anropats följt av ett kolon och raden som funktionsanropet skedde på. Stacken växer uppåt så det senaste oavslutade funktions- eller metodanropet ska stå överst. Om det är en metod som har anropats, kom ihåg parametern self som ska peka ut objektet som metoden opererar på. Observera att stacken innehåller en stackpost per anrop och inte en stackpost per funktion. Skillanden märks särskilt vid rekursiva anrop.

(iii) Ta bort stackposter från avslutade funktions-/metodanrop och uppdatera referensräkningen för eventuella objekt som pekats ut av deras lokala variabler och parametrar.

(iv) Se till att pilarna har spetsar och inte bara är streck. Pilarna utgår från variabler, attribut, listindex eller dictionary-nycklar och spetsen pekar alltid på ett objekt på heapen. Om pilarnas linjer behöver korsa varandra, gör en broliknande böj. Det blir lättare att felsöka och rätta diagrammen med så få korsande linjer som möjligt.

(v) Den tredje typen av låda i ett låd- och pildiagram förutom stackposter och den globala lådan är objekten på heapen. Överst i dessa lådor skriver vi objektets typ med en heldragen linje under. Om objektet endast har ett värde så skrivs detta värde under strecket i objektet. Om objektet däremot har attribut eller nycklar så ritas dessa som små lådor med namnet inuti lådan och pilar som utgår från dessa lådor visar bindningen till ett annat objekt. Om objektet har listindex så skrivs indexen bredvid de små lådorna.

(vi) För vissa objekt mäter vi referensräkning. Exempel inkluderar men är inte begränsade till: listor, objekt av klasser som definierats i koden samt heltal större än 256 eller mindre än -5. Vi mäter inte referensräkning på heltal n där $-5 \leq n \leq 256$ samt objekten None, True och False. Referensräkningen mäter antalet pilar som pekar ut objektet. För att vara mer noggrann så räknas endast de pilar som har sin utgångspunkt från den globala lådan, stackposter för funktioner/metoder som inte har returnerat ännu och objekt som har referensräkning större än 0. Referensräkningen skriver vi i en liten låda längst ner till höger på relevanta objekt.

(vii) Ta med alla referensräknade objekt som har varit utpekade av namn, attribut eller andra objekt under programmets körning. Om det är möjligt att skräpsamlaren (the garbage colllector) har tagit bort ett objekt, låt det stå kvar men rita en pil från skräpsamlaren till objektet som ska tas bort och uppdatera referensräkningen för övriga objekt som pekas ut av pilar från objekt med referensräkning 0.

26. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Bike:
2      def __init__(self, a, b):
3          self.a = a
4          self.b = b
5
6
7  class Tram:
8      def __init__(self, a):
9          self.a = a
10
11
12 def h(a, b, v):
13     c = a + b
14     d = Bike(a, c)
15     v.append(c)
16     return
17
18 def g(v):
19     x = 1
20     y = 2
21     h(x, y, v)
22     # Raden med kommentar
23     print("Bike or the Tram?")
24
25
26
27 def f():
28     v = []
29     for i in range(3):
30         v.append(Tram(i))
31     g(v)
32
33
34 f()
```

27. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Mushroom:
2      def __init__(self, a, b):
3          self.a = a
4          self.b = b
5
6
7  class Forest:
8      def __init__(self, a, b, c):
9          self.a = a
10         self.b = b
11         self.c = c
12
13
14 def r(a, b):
15     if a==b:
16         # Raden med kommentar
17         print("The Mushroom forest!")
18         return a + b
19     elif b < a:
20         return r(a-1, b)
21     else:
22         return r(a, b-1)
23
24
25 def f():
26     m = Mushroom(1, 2)
27     z = Forest(3, 4, 5)
28     r(z.a, z.c)
29
30
31 f()
```

28. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Twice:
2      def __init__(self, a, b):
3          self.a = a
4          self.b = b
5
6
7  class Golfer:
8      def __init__(self, a):
9          self.a = a
10
11
12 def h():
13     a = Golfer(8)
14     b = Twice(9, 10)
15     return b
16
17
18 def g(v1, v2):
19     v3 = [v1, v2]
20     v3.append([6, 7])
21     a = h()
22
23
24 def f():
25     v1 = [1, 2, 3]
26     v2 = [4]
27     v3 = v1[:1]
28     v3.append(5)
29     g(v1, v2)
30     # Raden med kommentar
31     print("Twice is not never!")
32
33
34 f()
```

29. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Resistance:
2      def __init__(self, a, b):
3          self.a = a
4          self.b = b
5
6
7      def m(self):
8          r = Capacitor(4)
9          return
10
11
12 class Capacitor:
13     def __init__(self, a):
14         self.a = a
15
16
17 def h():
18     v = [1, 2, 3]
19     return Resistance(1, 2)
20
21
22 def g():
23     h()
24     r = Resistance(12, 13)
25     r.m()
26     # Raden med kommentar
27     print("Resistors are awesome!")
28
29
30 g()
```

30. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Engine:
2      def __init__(self, a, b):
3          self.a = a
4          self.b = b
5
6
7  def r(n):
8      if n == 0:
9          # Raden med kommentar
10         print("The engine that runs the world!")
11         return 1
12     return n*r(n-1)
13
14
15 def g():
16     e1 = Engine(1000, 2000)
17     e2 = Engine(Engine(3000, 4000), 5000)
18     return e2
19
20
21 def f():
22     a = g()
23     b = 4
24     c = r(b)
25
26
27 f()
```

Här är frågorna slut. Om du har tid kvar, gå tillbaka och kontrollera dina svar.