PROGRAMMERING-PYTHON
TENTAMINA
2019-06-07

Kontaktperson: Johan Thunberg, tel: 0729-773846

Tentamensinstruktioner

Poängsättning

Tentamina är indelad i två delar

Del 1. Innehåller uppgifter för betyg 3, totalt 20 poäng. Du måste få minst 10 poäng för att få betyg 3.

Del 2. Innehåller uppgifter för betyg 4 och 5 totalt 10 poäng. Du måste ha fått betyg 3 på Del 1 för att kunna få poäng på Del 2. Del 2 kommer alltså inte att beaktas (eller betraktas) om betyg 3 inte har erhållits på Del 1.
För betyg 4 krävs det 6 poäng och för betyg 5, 8 poäng.

Redovisning

Tänk på att vara noggrann och strukturerad. Det är Du som skall visa vad Du kan! I bilagan finns information du kan behöva.

Hjälpmedel

Penna och kautschuk.

Lycka till!

Del 1 (20 poäng)

1. Du ska skriva ett program som fungerar som en kassaapparat. Input är hur mycket man har handlat för, samt hur mycket man har betalat (i kronor). Output är vad man ska få tillbaka. För outputen ska det anges hur många sedlar och mynt man ska få tillbaka av respektive valör. Det räcker att begränsa sig till en typ av sedel samt enkronor. **(2p)**

2. Under bra förutsättningar kan bromssträckan *s* (givet i meter) för en bil med hastigheten *v* (givet i km/h) och friktionstal *f* beräknas med formeln: s = (v**2)/(250*f). Friktionstalet *f* är större än 0 men mindre än 1. Antag att när bilen börjar bromsa så är det en vägg 50m framför bilen. Skriv ett program som frågar användaren efter en hastighet v och ett friktionstal f och räknar ut om bilen kommer krocka med väggen eller hinna stanna innan. **(2p)**

3. Vad kommer att skrivas ut på skärmen när följande kod körs?

```
a = 0
b = -1
while b <= 1:
    for c in range(-1,2):
        a += b - c
    b +=1
print(a)
```

(2p)

4. Skriv ett program där du utmanar datorn i spelet sten, sax och påse. Du väljer först ett av de tre alternativen, sedan väljer datorn. Det räknas ut vem som vann och resultatet skrivs ut på skärmen. Ett program har påbörjats nedan, fyll i det som fattas.

```
import random

ditt_val = input("Ange sten, sax eller påse:")
datorns_val = random.choice(["sten","sax","påse"])

print("Du valde:", ditt_val, ". Datorn valde",
datorns_val)

# Skriv fortsättningen här…
```

5. Vad skrivs ut på skärmen när koden nedan körs? Vad gör funktionen bar?

```
def bar(string):
    tuple1 = tuple(string.split())
    s1 = ""
    s2 = ""
    for i in tuple1[1]:
        s1 = i + s1
    for i in tuple1[0]:
        s2 = i + s2
    return s1 + " " + s2

print(bar("Kalle Kula"))
```

6. Vad skrivs ut på skärmen när koden nedan körs? Vad gör funktionen foo?

```
def foo(lista1, lista2):
    lista3 = []
    for i in lista1:
        if not i in lista2:
            lista3.append(i)
    for i in lista2:
        if not i in lista1:
            lista3.append(i)
    return lista3


print(foo([1,2,3],[3,2,1]))
print(foo([1,3,5],[2,4,6]))
```

(2p)

7. Betrakta funktionen nedan. *string* är en sträng med bokstäver mellan a och z (små bokstäver). *key* är en sträng med siffror (mellan 0 och 9).

- Förklara kortfattat hur funktionen använder *key* för att kryptera *string*.

- Hur kan man, genom att byta ut ett enda tecken i koden (om man bortser från funktionsnamnet), skapa en funktion som dekrypterar den sträng som funktionen *encrypt* har krypterat. Frågan är vilket tecken som ska bytas ut och till vad?

```
def encrypt(string, key):
    length_string = len(string)
    length_key    = len(key)

    nr1 = length_string//length_key
    nr2 = length_string%length_key
    key = nr1*key + key[0:nr2]

    string2 = ""
    for i in range(0,len(string)):
        string2 += chr(97 + (ord(string[i]) \
                             - int(key[i])-97)%25)
    return string2
```

8. Skriv en funktion som skapar en lista med slumpade bokstäver, och som sedan skriver ut alla dessa bokstäver i listan till en fil. **(2p)**

9. I denna uppgift ska du (virtuellt) singla slant. Du börjar med att skapa en klass Mynt som innehåller en variabel för sidan som är uppåt på myntet (ett mynt har två sidor: krona och klave). Klassen Mynt ska innehålla en funktion singla() (att kasta myntet) samt en funktion get_sida_upp() (returnerar sidan på myntet som är uppåt).

När du har är färdig med klassen så ska du skriva ett program som skapar tre mynt (instanser av Mynt) och kastar (singlar) dessa tills alla tre har samma sida uppåt (dvs. alla tre är klave eller alla tre är

krona, detta görs genom att anropa funktionen singla()). **(2p)**

10. Med hjälp av en dictionary ska du skapa en funktion *card_value(card)* som räknar vilket värde ett kort har i en kortlek. T.ex. så ska

```
print(card_value("hjärter ess"))
print(card_value("klöver knekt"))
print(card_value("ruter 2"))
```

generera output

```
14
11
2
```

Du kan välja var någonstans du vill definiera din dictionary, t.ex. inne i funktionen. Det viktiga är att den används för att ta reda på vilket värde som ett kort har. **(2p)**

Del 2 (10 poäng)

Du ska skapa en klass Spelautomat som simulerar en riktig så kallad enarmad bandit. Spelet går ut på att få flera stycken av samma symbol i rad på roterande hjul positionerade bredvid varandra. När så sker, betalas vinst ut.

Specifikationen är:

- Spelaren har en kassa att röra sig med som man kan göra insättningar till.
  - Man kan endast betala med pengar (ej kort).
  - Man kan när som helst, innan ett spel har börjat hämta ut pengarna i spelarkassan.
- Spelautomaten har också en kassa.
  - En vinstutbetalning begränsas av mängden kontanter i kassan.
  - När ett spel börjar, sätts det satsade beloppet in i spelautomatens kassa.
  - När ett spel avslutas sätts det eventuella vinstbeloppet in i spelarens kassa.
  - När en spelautomat skapas, finns 1000 kr i kassan.
- Spelet går till som sålunda att de olika hjulen snurrar ett slumpmässigt antal steg från sina ursprungliga konfigurationer och stannar på en viss kombination. Du kan t.ex. anta att
  - Det är tre hjul och att
  - varje hjul har 8 symboler (bokstäverna a-h) och att
  - antalet steg som snurras per hjul ges av ett positivt slumpmässigt heltal. Du kan också
  - spara olika vinstkombinationer i en dictionary som returnerar faktorn som de satsade pengarna ska multipliceras med för respektive vinstkombination.

Apendix till tentamina i Programmering DI2006
Python: sammanfattning

Kod	**Förklaring**

`name = "Kalle"`
`age = 10`
`pi = 3.14`
`savings= BankAccount()`
	En variabel av **typen string**
	En variabel av **typen int**
	En variabel av **typen float**
	En variabel av typen BankAccount

`print(y)` `print("Hej")`	**Skriver ut** värdet av variabeln y. Skriver ut "Hej"
`x = input("Namn: ")`	**Skriver ut**: "Namn: " och **pausar tills användaren skriver in** något och trycker på Enter. **Lagrar sedan** texten användaren skrev in, i variabeln x.

`x = 15`
`if x < 10:`
`  s = "x < 10"`
`  print(s)`
`elif x < 20:`
`  s = "x < 20"`
`  print(s)`
`elif x == 20:`
`  print("x = 20")`
`else:`
`  print("else:")`
`  print("x > 20")`
	Villkorssatser.
	
	if testar om ett villkor är uppfyllt.
	
	elif hänger ihop med if och testas bara om testet i if misslyckas.
	
	else hänger också ihop med if och testas bara om det i if och eventuella elif misslyckas.

`age == 100`	Exempel på villkor man kan testa. **Kom ihåg: == INTE =** när man vill testa ett villkor!
`x < 100 and y < 100`	Exempel på villkor man kan testa. Testar om både variabeln x **och** variabeln y är mindre är 100.

for tal **in** range(100): print(tal)	**for loop** För varje tal från 0 till och med 99: Skriv ut talet.
for num **in** [1, 6, 2, 5]: print(num)	För varje nummer i listan [1, 6, 2, 5]: Skriv ut numret.

var = 10
while var < 100:
  print(var)
  var = var + 1
	while loop
	Medans villkoret
	var < 100 är uppfyllt, gör följande om och om igen:
	- Skriv ut värdet av var
	- Öka var med ett .

def funktionsnamn(a, b):
  r = a*b + 10
  r = r/10
  r = r – 10
  return r
	Funktionsdefinition. Denna kod körs först då funktionen anropas. a och b kallas för **parametrar** och är variabler utan värde som kan användas inuti funktionen.

funktionsnamn(1, 2) x = funktionsnamn(2, 5)	**Anrop** av funktionen *funktionsnamn* kör koden i funktionsdefinitionen. **Argumenten** 1 och 2 är värden som lagras i parametrarna i funktionsdefinitionen

fil=open("filnamn", "r")
 	
for ord in fil:
  //gör något
 	
text=fil.read()
line=fil.readline()
	Öppna en fil och läser den ord för ord. w- skriv , a-append
	
	läser en fil till variabel text
	
	läser en rad från fil till variabel line

lista=[] for var in lista	Skapa tom lista Iterera genom lista

Skapa klasser och objekt

```
class BankAccount:

     def __init__(self, iname, ibalance):
          self.__name=iname
          self.__balance=ibalance

     def deposit(self, amount):
          balance=balance-amount
```

```
savigs= BankAcount(”kalle”, 100)    // skapa objekt
```

Användbara funktioner:

```
int(a)  // koverterar a från String till int
str(a)  // konverterar a från int till String
ord(a)  // konverterar a från char till int
chr(a)  // konverterar a från int till char

random.randint(a,b) // slumpar tal mellan a och b

text.split(delimitator)  Dela en sträng i en lista av ord.
Argumenten delimitator används som separator. Metoden returnerar
en lista.
```

Andra String metoder:

Table 8-1 Some string testing methods

Method	Description
`isalnum()`	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
`isalpha()`	Returns true if the string contains only alphabetic letters and is at least one character in length. Returns false otherwise.
`isdigit()`	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
`islower()`	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
`isspace()`	Returns true if the string contains only whitespace characters and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`).
`isupper()`	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.

Table 8-3 Search and replace methods

Method	Description
`endswith(substring)`	The *`substring`* argument is a string. The method returns true if the string ends with *`substring`*.
`find(substring)`	The *`substring`* argument is a string. The method returns the lowest index in the string where *`substring`* is found. If *`substring`* is not found, the method returns –1.
`replace(old, new)`	The *`old`* and *`new`* arguments are both strings. The method returns a copy of the string with all instances of *`old`* replaced by *`new`*.
`startswith(substring)`	The *`substring`* argument is a string. The method returns true if the string starts with *`substring`*.

List metoder

Table 7-1 A few of the list methods

Method	Description
`append(item)`	Adds *item* to the end of the list.
`index(item)`	Returns the index of the first element whose value is equal to item. A `ValueError` exception is raised if item is not found in the list.
`insert(index, item)`	Inserts *item* into the list at the specified *index*. When an item is inserted into a list, the list is expanded in size to accommodate the new item. The item that was previously at the specified index, and all the items after it, are shifted by one position toward the end of the list. No exceptions will occur if you specify an invalid index. If you specify an index beyond the end of the list, the item will be added to the end of the list. If you use a negative index that specifies an invalid position, the item will be inserted at the beginning of the list.
`sort()`	Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
`remove(item)`	Removes the first occurrence of *item* from the list. A `ValueError` exception is raised if item is not found in the list.
`reverse()`	Reverses the order of the items in the list.

Dictionary metoder:

Table 9-1 Some of the dictionary methods

Method	Description
`clear`	Clears the contents of a dictionary.
`get`	Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
`items`	Returns all the keys in a dictionary and their associated values as a sequence of tuples.
`keys`	Returns all the keys in a dictionary as a sequence of tuples.
`pop`	Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
`popitem`	Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.
`values`	Returns all the values in the dictionary as a sequence of tuples.