Salbuespenak

Baliteke programazioan bikaina izatea. Baina, hala ere, zure programek huts egin dezakete, gauza batzuk zure programaren kontroletik kanpo daudelako. Beraz, nahiz eta zure programa zuzena izan, gauzak oker joan daitezke.

Adibidez:

- Zure programak erabiltzaileari zenbaki bat idaztea eskatzen badio, baina erabiltzaileak letrak idazten baditu ala ezer idazten ez badu, zure programak huts egingo du.
- Zure programak fitxategi bat irakurri behar badu, baina fitxategi hori existitzen ez bada, zure programak huts egingo du.
- Zure programak sarera konektatu behar badu, baina zure ordenagailua konektatuta ez badago, zure programak huts egingo du.

Ikus dezakezunez, egoera batzuetan programak ezin du kontrolik izan. Zorionez, badugu mekanismo bat aukera ematen diguna gauza horietako bat gertatzen bada gure programak huts egin ez dezan eta besterik gabe amai dadin. Eta mekanismo hori salbuespenak dira.

Salbuespenak Python-en

Adibidez, demagun honako programa oso sinple bat dugula, erabiltzaileari zenbaki bat eskatu eta biderketa bat egiten duena:

```
balioa = input ("Sartu zenbaki bat:")
balioa = int(balioa)
karratua = balioa * balioa
print ("Karratua da:", karratua)
```

Erabiltzaileak behar ez duena sartzen badu, honako hau ikusiko dugu:

```
Sartu zenbaki bat: x
Traceback (most recent call last):
File "salbuespena.py", line 2, in < module>
balioa = int (balioa)
ValueError: invalid literal for int () with base 10: 'x'
```

Salbuespen baten bidez, programak huts egitea ekidin dezakegu, eta gutxienez erabiltzaileari errore mezu bat erakutsi. Salbuespena lengoaiaren egitura bat gehiago da, eta forma hau du:

```
try:
  huts_egin_dezakeen_kodea
except:
  huts_egitean_gertatzen_dena
```

Ikus dezagun aurreko adibidea, salbuespen blokearen barruan huts egin dezakeen kodea babestuz:

```
balioa = input ("Sartu zenbaki bat:")

try:
  balioa = int (balioa)
  karratua = balioa* balioa
  print ("Karratua da:", karratua)
except:
  print ("Errorea datua bihurtzean!")
```

Orain, datu oker bat sartuz gero, honako hau ikusiko dugu:

```
Sartu zenbaki bat: x
Errorea datua bihurtzean!
```

Halaber, programa hobetu ahal izango litzateke, balioa berriro eskatzeko eta ez amaitzeko.

Errore motaren araberako salbuespen espezifikoak daude, eta errore-mezu zehatzagoa erakusteko erabil daitezke.

Fitxategien kudeaketa

Orain arte datu gutxi erabili ditugu, erabiltzaileak pantaila bidez idazten duena edo aldagaietan daukaguna. Baina datu kopuru handiagoak erabili nahi baditugu, fitxategietan irakurri eta idatz dezakegu. Era guztietako fitxategiak daude: testua, multimedia (musika, bideoa) eta fitxategi bitarrak. Horiek guztiak programa batetik erabil daitezke. Sarrera gisa, ikus dezagun nola erabil ditzakegun testu fitxategiak.

Fitxategien irakurketa

Fitxategi bat irakurri ahal izateko, alde batetik fitxategi hori egon behar da, gero ireki eta irakurri ahal izango dugu. Kode honetan, programaren leku berean dagoen fitxategi bat irakurtzen da:

```
fitxategia = open("testua.txt", "r")
edukia = fitxategia.read()
print(edukia)
fitxategia.close()
```

Kontuan hartzekoak:

- Fitxategia irakurtzeko, lehenik eta behin ireki egin behar da, honako honekin: `open("testua.txt", "r")`

- Fitxategia irekitzean, haren izena adierazi behar da, eta, beste direktorio batean badago, fitxategiaren *path* edo "bidea". Adibidez `Testuak` izeneko karpetan egonez gero, hau izango litzateke bidea: `Testuak/testua.txt`.
- `"r"` parametroak adierazten du fitxategia irakurketa moduan bakarrik irakurtzen dugula.

Testu fitxategia horrelako zerbait izan liteke, eta programak hori bera erakutsiko luke pantailan.

```
Hau testu bat da
hainbat lerrokoa
Eta irakur daiteke
oso erraz
```

Lerroz-lerro irakurtzen?

Aurreko adibidean, bat-batean irakurri dugu fitxategiaren eduki osoa, testu aldagai batean gordeta. Baina batzuetan, baliteke fitxategia lerroz-lerro irakurri nahi izatea. Horretarako, honako funtzio hau erabili behar dugu:

```
fitxategia = open("testua.txt", "r")
lerroak = fitxategia.readlines()
for lerroa in lerroak:
  print (lerroa.strip())  # lerroaren amaieran dagoen \n karakterea kendu

fitxategia.close()
```

JSON fitxategiak

Aurrekoa bezalako testu fitxategi sinpleek informazioa izan dezakete, baina ez dira programa baterako oso datu erabilgarriak. Programa batek erraz manipulatu ditzakeen datuak irakurri edo gorde nahi baditugu, formatu jakin bat erabiltzea komeni da. Programazioko formatu ezagunenetako bat JSON formatua da. Python-en hiztegi egituren antza duen formatua da. Python lengoaiaren zerrendak bezala ere irudikatzeko aukera badu.

Hurrengo edukia JSON formatuan dago. Hainbat objektu dituen zerrenda da. Fija zaitez, JSON **motako objektuak Python-eko hiztegien berdinak dira!**:

```
[
  {"id": 66, "izena": "Ada"},
  {"id": 2, "izena": "Neko"},
  {"id": 4, "izena": "Bug"}
]
```

Formatu horren alde ona da gure Python programara erraz inporta daitekeela, betiere zuzena bada, noski.

Eduki hori irakurri eta datu horiek hiztegi zerrenda bihurtzeko, `json` liburutegia erabiliko dugu. Fitxategi horren edukia automatikoki kargatu ahal izango dugu aldagai batean. Hortik aurrera, eduki hori guztia zerrenda gisa erabili ahal izango dugu, non elementu bakoitza hiztegi bat den!:

```
import json

fitxategiarea = open("testua.json", "r")
edukia = json.load(fitxategiarea)

for pertsonaia in edukia:
    print(pertsonaia["izena"])

fitxategiarea.close()
```

Pantailan, hau ikusiko dugu:

```
Ada
Neko
Bug
```

Fitxategien idazketa

Fitxategiak idazteko, prozesua antzekoa da, baina bi gauza egin behar ditugu:

- Fitxategia idazkera moduan irekitzea.
- `write` funtzioa erabili edukia idazteko.

Kode honekin, testu-lerro pare bat idatziko ditugu fitxategian:

```
fitxategia = open("testua.txt", "w")
fitxategia.write("Idatzi lerro bat\n")
fitxategia.write("Idatzi beste lerro bat\n")
fitxategia.close()
```

KONTUZ! Horrela idazten badugu, fitxategiaren edukia ezabatuko baitugu. Fitxategiaren edukia horrela geratuko litzateke:

```
Lerro bat idazten dut
Beste lerro bat idazten dut
```

Edukia fitxategiaren amaieran gehitu nahi badugu, fitxategia `"a"` (*append*) moduan ireki behar dugu:

```
fitxategia = open("testua.txt", "a")
fitxategia.write("Gehitu lerro bat\n")
fitxategia.write("Gehitu beste lerro bat\n")
fitxategia.close()
```

Orain, fitxategiaren edukia hau izango litzateke:

```
Lerro bat idazten dut
Beste lerro bat idazten dut
Gehitu lerro bat
Gehitu beste lerro bat
```

Fitxategi batean JSON idazten

JSON formatuko fitxategi baten kasuan, kontutan hartu behar da idazteko unean gure datuak testu bihurtu behar direla Zorionez, automatikoki egiten duen funtzio bat dago: `json.dumps()`

Hurrengo adibidean, json fitxategi baten edukia aldagai baten barruan kargatzen da. Gero elementu bat gehituko diogu zerrenda horri. Fitxategia berriro irekiko dugu, idazkera moduan, eta `write` bat egingo dugu, `json.dumps` erabiliz edukia testu bihurtzeko:

```
import json

fitxategia = open("testua.json", "r")
edukia = json.load(fitxategia)
fitxategia.close()

pertsonaia = { "id": 666, "izena": "Gumball" }
edukia.append(pertsonaia)

fitxategia = open("testua.json", "w")
fitxategia.write(json.dumps(edukia))
fitxategia.close()
```

Liburutegiak

Programak gero eta konplexuagoak diren heinean, litekeena da funtzio asko definitu behar izatea, edo diseinua klaseetan bereizi behar izatea, etab. Dena fitxategi berean eduki dezakegun arren, ez litzateke gure kodea antolatzeko modurik onena. Egokiena da klase bakoitza bere fitxategian bereiztea, eta funtzio edo funtzio-multzo bakoitza bere fitxategian biltzea.

Kodea fitxategietan eta karpetetan antolatu ondoren, beste fitxategi batzuetan berrerabil ditzakegu. Ikus dezagun adibide sinple bat. Funtzio hau definituko dugu `matematika.py` izeneko fitxategi batean:

```
Batuketa (a, b):
  return a + b

  def kendu(a, b):
    return a - b

  def batu(a):
    return a - 1
```

Orain, fitxategi hori beste programa batean sar dezakegu, `import` aginduaren bidez. Direktorio berean badaude, besterik gabe egin daiteke:

```
import matematika

balio1 = 5
balio2 = 10

emaitza = matematika.batu(balio1, balio2)
print(emaitza) # 15
```

6.0 Ariketa

Aurreko ariketa batean, pasahitzen sorgailu bat egitea proposatu zen. Erabili kode bera, baina sar ezazu fitxategi baten barruan. Sortu beste fitxategi bat kode hori inportatzeko eta erabiltzeko.

`sortu.py` fitxategia:

```
import random

def ausazkoa (max):
    return random.randint(0, max - 1)

def pasahitzaSortu (luzera):
    hizkiak = ["a","b","c","d","e","f","g","h","i","j","k","l",
        "m","n","ñ","o","p","q","r","s","t","u","v","w","x","y","z",
        "0","1","2","3","4","5","6","7","8","9",".","-","_","!","$"]
    pasahitza = ""

    for i in range(luzera):
        hizkia = hizkiak[ausazkoa(len(hizkiak))]
        pasahitza = pasahitza + hizkia

    return pasahitza
```

Eta fitxategian honela erabiliko genuke:

```
import sortu

pasahitza = sortu.pasahitzaSortu(8)

print(pasahitza)
```

Emaitza:

```
g3ep-ahx
```

Klaseekin gauza bera egin daiteke. Demagun `PantailaIrakurgailua` izeneko klasea dugula `pantaila_irakurgailua.py`izeneko fitxategi batean. Kontsolatik datuak irakurtzeko aukera ematen digun klase bat da:

```
class PantailaIrakurgailua:
  def irakurZenbaki (self, mensaje = "Sartu zenbaki bat:"):
    numero = input (mezua)
    return int (zenbakia)

  def irakurTestua(self, mensaje = "Sartu testua:")
    testua = input (mezua)
    return testua
```

Orain, klase hori beste fitxategi batean berrerabil dezakegu, gure fitxategiarekin batera.

```
import pantaila_irakurgailua
import matematika

irakurgailua.PantailaIrakurgailua()
balio1 = irakurgailua.irakurZenbaki()
print (matematika.batu(balio1))
```

Pantailan horrelako zerbait ikus daiteke:

```
Sartu zenbaki bat: 6
7
```

6.1 Ariketa

Definitu `Menu` izeneko klase bat, honako funtzio hauek dituena:

1. `def init__(self, aukerak)`: aukeren zerrenda bat jasotzen du parametro gisa.
2. `def erakutsi(self)`: aurretik zenbaki bat duen aukerak erakusten ditu, `print` deituz.

3. `def hautatu (self, zenbakia)` Itzuli `True`, baldin eta aukeratutako zenbakia menuan badago; bestela, itzuli `False`.

Gero klase hori 6.2.py fitxategian inportatu eta erabili.

Fitxategia:

```
class Menu:
    def __init__ (self, aukerak):
        self._aukerak = aukerak

    def erakutsi (self):
        for i in range(len(self._aukerak)):
            print(f"{i+1} {self._aukerak[i]}")

    def hautatu (self, zenbakia):
        return zenbakia > 0 and zenbakia <= len(self._aukerak)
```

6.2.py fitxategia, programa nagusia:

```
import menu
nireMenua = menu.Menu(["Erakutsi", "Kendu", "Irten"])

nireMenua.erakutsi()

if nireMenua.hautatu(1):
    print("1. aukera badago menuan")
else:
    print("1. aukera ez dago menu honetan")
```

Emaitza:

```
1 Erakutsi
2 Ezabatu
3 Irten
Oraingo 1. aukera
```

Proposatutako ariketak

6.0 Ariketa

Sortu `Fitxategia` izeneko klase bat, honako funtzio hauekin:

1. `def __init__ (self, fitxategiIzena)`: ireki beharreko fitxategi izena jasotzen du parametro gisa.
2. `def irakurri (self)`: fitxategiaren edukia jasotzen duen kate bat itzultzen du.

3. `def idatzi (self, edukia)`: fitxategian parametro gisa pasatzen zaion edukia idazten du.

Gero, liburutegi bezala erabili behar duzu beste fitxategi batean inportatuz.

Fitxategia:

```
class Fitxategia:
  def __init__(self, fitxategiIzena):
      self._fitxategiIzena = fitxategiIzena

  def irakurri(self):
      fitxategia = open(self._fitxategiIzena, "r")
      datuak = fitxategia.read()
      fitxategia.close()

      return datuak

  def idatzi(self, eduki):
      fitxategia = open(self._fitxategiIzena, "w+")
      fitxategia.write(eduki)
      fitxategia.close()
```

6.0.py fitxategia, programa nagusia:

```
import fitxategia
from datetime import date

nireFitxategia = fitxategia.Fitxategia("6.0.txt")

print("Aurreko edukia: ", nireFitxategia.irakurri())

nireFitxategia.idatzi("Eduki aldatuta!!! " + str(date.today()))
print("Edukia:", nireFitxategia.irakurri())
```

Testu-fitxategia:

```
Horixe da gaur egungo edukia.
```

Emaitza:

```
Aurreko edukia: Eduki aldatua!!! 2024-08-18
Edukia: Edukia aldatuta!!! 2024-08-23
```

6.1 Ariketa

Sor ezazu `Zerrenda` izeneko klase bat, honako funtzio hauekin:

1. `def __init__ (self, fitxategiIzena)`: json fitxategi baten izena hartzen du parametrotzat, eta haren edukia zerrenda batean kargatu behar da. Zerrenda hori atributu gisa definituko da. Edukiak hiztegi-zerrenda bat izan behar du, formato honekin `[{"id": 1, "izena": "Juan"}, {...}]
2. `def existitzenDa(self, izena)`: `True` ala `False` itzuli parametro gisa pasatzen den `izena` zerrendan badago.
3. `def xehetu (self)`: zerrendako izen guztiak letra xeheetara pasatu behar dira.
4. `def posizioa (self, izena)`: izen hori dagoen posizioa itzuli behar duzu.

Gero, klase hau erabili behar duzu beste fitxategi batean inportatuz.

Zerrenda fitxategia:

```
import json

class Zerrenda:
    def __init__ (self, fitxategiIzena):
        edukia = open(fitxategiIzena, "r")
        self._datuak = json.load(edukia)
        edukia.close()

    def existitzenDa (self, izena):
        for datu in self._datuak:
            if datu["izena"] == izena:
                return True

        return False

    def xehetu (self):
        self._datuak = list(map(lambda datu: { "id": datu["id"], "izena":
datu["izena"].lower() }, self._datuak))

    def posizioa (self, izena):
        i = 0
        for datu in self._datuak:
            if datu["izena"] == izena:
                return i
            i += 1
        return -1

    def inprimatu (self):
        for datu in self._datuak:
            print(datu)
```

6.1 fitxategia, programa nagusia:

```
import zerrenda

nire_zerrenda = zerrenda.Zerrenda("6.1.json")
```

```
badaude = nire_zerrenda.existitzenDa("eugene")
if badaude:
    print("Dago!")

nire_zerrenda.xehetu()
nire_zerrenda.inprimatu()

badaude = nire_zerrenda.existitzenDa("eugene")
if badaude:
    posizioa = nire_zerrenda.posizioa('eugene')
    print("Badago!")
    print(posizioa)
```

`6.1.json` fitxategia:

```
[
    {
        "id": 3,
        "izena": "Juan"
    },
    {
        "id": 5,
        "izena": "Eugene"
    },
    {
        "id": 10,
        "izena": "Paul"
    }
]
```

Emaitza:

```
{'id': 3, 'nombre': 'juan'}
{'id': 5, 'izena': 'eugene'}
{'id': 10, 'izena': 'paul'}
Badago!
1
```

6.2 Ariketa

Sor ezazu `Zereginak` klase bat, honako funtzio hauekin:

1- `def __init__ (self)` `zereginak.json` izeneko fitxategia ireki behar duzu, eta honako formatu hau izango duten hiztegiak zerrenda batean kargatu: `{"id": 1, "zeregina"": "Ikasi zerbait"}`. Zerrenda hori atributua izango da. 2- `def sortu (self, zeregina)`: objektu berri bat sortu eta zerrendan gordetzen du. 3- `def ezabatu (self, id)`: ID horrek duen zerrendako zeregina bat

ezabatzen du. 4- `def gorde (self)`: gorde zerrenda `zeregina.json` fitxategian. 5- `def erakutsi (self)`: string batean itzultzen ditu zeregin guztiak.

Gero, klase hau erabili behar duzu beste fitxategi batean inportatuz.

Fitxategia: `zereginak.py`

```
import json

class Zereginak:
    def __init__ (self):
        fitxategia = open("zereginak.json", "r")
        self._zereginak = json.load(fitxategia)
        fitxategia.close()

    def sortu (self, id, zeregina):
        berria = { "id": id, "zeregina": zeregina };
        self._zereginak.append(berria)

    def gorde (self):
        fitxategia = open("zereginak.json", "w")
        fitxategia.write(json.dumps(self._zereginak))
        fitxategia.close()

    def ezabatu(self, id):
        self._zereginak = list(filter(lambda datua: datua["id"] != id,
self._zereginak))

    def erakutsi (self):
        emaitza = ""
        for datua in self._zereginak:
            emaitza += json.dumps(datua) + "\n"

        return emaitza
```

Fitxategia: `zereginak.json`:

```
[
    {"id": 3, "zeregina": "Go ikasi"},
    {"id": 5, "zeregina": "Rust ikertu"},
    {"id": 10, "zeregina": "Lo egin"}
]
```

Fitxategia:

```
import zereginak
```

```
nireZereginak = zereginak.Zereginak()

print(nireZereginak.erakutsi(), "\n---")

nireZereginak.sortu(2, "Gehiago ikasi")
print(nireZereginak.erakutsi(), "\n---")

nireZereginak.ezabatu(2)
print(nireZereginak.erakutsi(), "\n---")

nireZereginak.sortu(66, "Irakurri")
print(nireZereginak.erakutsi(), "\n---")
nireZereginak.gorde()
```

Emaitza:

```
{"id": 3, "zeregina": "Go ikasi"}
{"id": 5, "zeregina": "Rust ikertu"}
{"id": 10, "zeregina": "Lo egin"}

---
{"id": 3, "zeregina": "Go ikasi"}
{"id": 5, "zeregina": "Rust ikertu"}
{"id": 10, "zeregina": "Lo egin"}
{"id": 2, "zeregina": "Gehiago ikasi"}

---
{"id": 3, "zeregina": "Go ikasi"}
{"id": 5, "zeregina": "Rust ikertu"}
{"id": 10, "zeregina": "Lo egin"}

---
{"id": 3, "zeregina": "Go ikasi"}
{"id": 5, "zeregina": "Rust ikertu"}
{"id": 10, "zeregina": "Lo egin"}
{"id": 66, "zeregina": "Irakurri"}

---
```

6.3 Ariketa

Sor ezazu `Jokalaria` izeneko klase bat, eduki hau izango duena:

1. `def __init__ (self, izena, zenbakia)`: parametroak esleitzen dizkie `_izena` eta `_zenbakia`, atributuei.
2. `get`/`set` metodoak izenerako eta dortserako.
3. `def info (self)`: *string* bat itzultzen du jokalariaren informazioarekin.

Sor ezazu `Taldea` izeneko klase bat, eduki honekin:

1. `def karga(self)`: `jokalariak.json` izeneko fitxategi bat ireki behar duzu. Fitxategi horrek jokalari-hiztegien zerrenda bat izango du [{"izena": "Pele", "dortsala": 10}, {}. Eta fitxategiaren objektu bakoitzeko, `Jokalaria` motako instantzia bat sortu behar duzu, eta `this._jokalariak`izeneko zerrenda batean sartu.
2. `def erakutsi (self)`: jokalarien zerrenda osoa erakutsi behar duzu.
3. `def fitxaketa (self, izena, dortsala)`: jokalari berri bat sartu behar duzu zerrendan,`Jokalaria`ren instantzia bat sortuz.

`Taldea` klasea, `Jokalaria` klasea inportatu beharko duzu, erabili ahal izateko.

`jokalaria.py` 'fitxategia:

```
class Jokalaria:
  def __init__ (self, izena, zenbakia):
      self._izena = izena
      self._zenbakia = zenbakia

  @property
  def izena (self):
      return self._izena

  @izena.setter
  def izena (self, izena):
      self._izena = izena

  @property
  def zenbakia (self):
      return self.zenbakia

  @zenbakia.setter
  def zenbakia (self, zenbakia):
      self._zenbakia = zenbakia

  def toString (self):
      return self._izena + " " + str(self._zenbakia)
```

`taldea.py` taldea fitxategia:

```
import json
import jokalaria

class Taldea:
    def karga(self):
        edukia = open("./jokalariak.json")
        jokalariak = json.load(edukia)
        print("Kargatuta:", jokalariak)
        self._jokalariak = []
        for j in jokalariak:
            self._jokalariak.append(jokalaria.Jokalaria(j["izena"],
j["zenbakia"]))
```

```
    def fitxaketa(self, izena, dorsala):
        fitxategiBerria = jokalaria.Jokalaria(izena, dorsala)
        self._jokalariak.append(fitxategiBerria)

    def inprimatu(self):
        for jokalaria in self._jokalariak:
            print(jokalaria.toString())
```

`jokalariak.json` jokalarien fitxategia:

```
[
    {
        "izena": "Maradona",
        "zenbakia": 10
    },
    {
        "izena": "Pele",
        "zenbakia": 8
    }
]
```

`6.3.py` fitxategia, programa nagusiarekin:

```
import taldea

nireTaldea = taldea.Taldea()

nireTaldea.karga()
nireTaldea.inprimatu()
nireTaldea.fitxaketa("Gento", 11)
nireTaldea.inprimatu()
```

Emaitza:

```
Loaded: [{'zenbakia': 10, 'izena': 'Maradona'}, {'zenbakia': 8, 'izena':
'Pele'}]
Maradona 10
Pele 8
Maradona 10
Pele 8
Gento 11
```