ABB Industrigymnasium

Joakim Flink 2020-03-04

Läs från excel

Vad vi ska lära oss:

1. Läsa information från Excelfiler
2. Göra funktionella metoder
3. Sortera arrayer

Innehåll

Instruktioner

1) Hämta data från Excel

Det första vi ska göra är att vi ska installera openpyxl, detta för att kunna ladda in Exceldokument i python. Detta göra vi genom att skriva följande i terminalen:

```
pip install openpyxl
```

När detta är installerat så skapar vi en ny pythonfil och skriver in följande högst på sidan:

```
import openpyxl
wb= openpyxl.load_workbook('mello.xlsx')
sheet= wb.active
```

Vad det betyder är att vi laddar in biblioteket, öppnar en excelfil som heter mello.xmlsx och väljer det första bladet (det som är aktivt). Du kan ladda ner excelfilen här.

För att få veta vad som ligger på en viss plats i bladet, till exempel första värdet på A skriver man:

```
print(sheet['A1'].value)
```

Testa ändra bokstav och siffra för att se vad som visas.

Man kan även göra det på andra sätt, till exempel såhär, vilket är mer dynamiskt:

```
print(sheet.cell(row=1, column=1).value)
```

Enligt detta sätt väljer vi vilken rad vi vill ha en cell ifrån och vilken kolumn, sen plockar vi ut dess värde. Detta sätt är vackrare om man ska använda det dynamiskt, vilket vi ska, med hjälp av funktionell programmering.

2) Funktionell programmering, vad är det?

Funktionell programmering är ett koncept där man använder förkortade funktioner för att göra samma sak som en vanlig funktion gör, fast kortare.

För att lägga värden i en array med hjälp av en forloop, har vi tidigare gjort såhär:

```
värden=[]
for i in range(2,10):
        värden.append(i)
print(värden)
```

Med hjälp av funktionellprogrammering kan vi korta ner detta till en enskild rad:

```
värden=[ i for i in range(2,10)]
print(värden)
```

Det man kan tänka är att där man vill spara värdena ska vara längst till vänster, sen ska själva aktiviteten ligga inuti [brackets]. Först skriver man vad man vill göra (i detta fall spara det nuvarande värdet i loopen) och sedan loopen och dess villkor.

Man kan även lägga in fler faktorer som här nedan då man kräver att värdet ska vara jämt delbart med 0 för att läggas till.

```
värden=[ "Värde "+str(i) for i in range(2,10) if i%2 is 0]
```

Och det finns inget som säger att du måste nöja dig med en loop,

```
värden=[ "Värde "+str(i) for i in range(2,10) if i%2 is 0 if i is not 8]
```

3) Använda Excelvärden

För att använda värdena från excelfilen kan vi använda våra kunskaper och skriva på följande sätt:

```
artist=[]
for i in range(2,sheet.max_row):
    artist.append(sheet.cell(row=i, column=2).value)
print(artist)
```

Det vi gör är att vi skapar en array, väljer att lägga alla värden i kolumn 2 från och med rad 2 till slutet av excelfilen. Sedan printar vi såklart ut det för att se vad det är. Men varför inte göra detta funktionellt?

```
artists=[ sheet.cell(row=i, column=2).value for i in range(2,sheet.max_row)]
print(artist)
```

Vad är det som visas? Förhoppningsvis en array med alla artister som deltog i mellon 2019, oj vilket år!

Hur ska vi då göra för att få ut fler värden? Jo vi skapar en Dictionary och hämtar de värden vi vill ha och ger den unika variabelnamn. För att få ut nano skriver vi såhär:

```
Nano = {'Namn': sheet.cell(row=2, column=2).value, 'Sång': sheet.cell(row=2,
column=3).value, 'Poäng': sheet.cell(row=2, column=6).value, 'Röst':
sheet.cell(row=2, column=5).value}
print(Nano)
>>> {'Namn': 'Nano', 'Sång': '"Chasing Rivers"', 'Poäng': 54, 'Röst': 1096662}
```

Men det är väl lite bökigt och o praktiskt att skriva sådär för alla 27 deltagare? Då gör vi inte det utan använder funktionell programmering:

```
artists=[{"Namn":sheet.cell(row=i, column=2).value,
        "Sång":sheet.cell(row=i, column=3).value,
        "Poäng":sheet.cell(row=i, column=6).value,
        "Röst":sheet.cell(row=i, column=5).value}
for i in range(2,sheet.max_row) ]
```

På detta sätt får vi en array med en Dictionary för varje artist och dess i min mening väsentliga information.

I mitt fall, vill jag inte ha alla data utan bara info om de som gick vidare från deltävlingen. Hur gör man det lättast då, jo genom att använda en ifsats.

```
Vidare=[{"Namn":sheet.cell(row=i, column=2).value,
        "Sång":sheet.cell(row=i, column=3).value,
        "Poäng":sheet.cell(row=i, column=6).value,
        "Röst":sheet.cell(row=i, column=5).value
} for i in range(2,sheet.max_row)
    if not sheet.cell(row=i, column=8).value.startswith("Utsl") ]
```

Det vi gör är att säger att alla celler på kolumn 8 som inte börjar med ”Utsl” ska sparas i arrayen, alltså de som går vidare. Hade jag gjort likadant med kolumn 2 och startswith(”N”) och tagit bort not i början så hade jag fått en artist i arrayen, Nano.

4) Fina till det

Nu har vi två arrayer, Artister och Vidare. Varför inte fint skriva ut vilka som är med och tävlar i år och vilka som gått vidare? För detta ska vi såklart använda funktionellprogrammering.

```
[print(artist["Namn"]," är med i år!") for artist in artists]
[print(artist["Namn"],"var en av de",len(Vidare)," som gick vidare med
låten",artist["Sång"]) for artist in Vidare]
```

Snyggt det ger oss bland annat utskrifterna:

```
Arvingarna  är med i år!
Nano var en av de 15  som gick vidare med låten "Chasing Rivers"
Wiktoria var en av de 15  som gick vidare med låten "Not With Me"
```

Alltså med bara en enkel rad kan vi prydligt skriva ut värden från arrayer och göra mycket annat. Det vi vill göra nu är ju att veta vilka som har fått mest poäng, dessa borde ju på något sätt ge en hint om vem som vann mellon.

Detta gör vi genom att använda sorted.

```
Vidare = sorted(Vidare, key=lambda k: k['Poäng'])
Vidare = sorted(Vidare, key=lambda k: k['Poäng'], reverse=True)
```

I exemplet ovan sorterarar vi på två olika sätt, med högst poäng först samt sist i listan. Vi väljer att spara den som Vidare och använder som tidigare nämnt Poäng variabeln ur Dictionarin som mätstock.

Det vi nu vill göra är att bara visa de som har mest poäng, detta kan vi nu göra genom att klippa av arrayens längd.

```
Vidare=Vidare[0:10]
```

Det vi gör ovan är att säga att vi vill ha kvar värden på plats 0 till 10 i arrayen, alltså de högsta poängmässigt tack vare vår sortering.

Man kan även få en sista minuten impuls eller bara vilja visa vissa värden i t.ex en print men behålla själva arrayen i sin helhet, då kan man göra klippningen i loopen.

```
[print(artist["Namn"]," ",artist["Poäng"]," ",artist["Röst"]) for artist in
Vidare[:3]]
```

Sådär, då kan du lite funktionell programmering, ta in excel filer, välja ut värden och sortera dem. Vill du veta hur man skriver till excelfiler föreslår jag att du fortsätter läsa i Automate the boring stuff with python , kap 12. Vill du veta mer om funktionell programmering så föreslår jag att du kikar i SERIOUS PY THON, kap 8.

Uppgifter

I denna uppgift ska vi arbeta med Farmen, inte vilken Farmen som helst utan Farmen den tredje säsongen 2003. Datat vi vill använda finns här och är extraherat från Wikipedia.

Ni behöver inte använda enbart funktionell programmering men försök få till det ibland.

Datat är uppdelat i tre delar, deltagare, veckosammanfattning och tittarsiffror. Du får ut de korrekta namnen genom att skriva följande sak i kod när du anropat rätt xlsx fil:

```
print(wb.get_sheet_names())
```

För att använda ett visst sheet kan man skriva följande:

```
Deltagar_sheet= wb.get_sheet_by_name('Deltagare')
```

1) Presentera deltagarna

Skapa en fin array med dictionaries i för varje deltagare. Skriv sedan fint ut vart deltagern kommer ifrån, hur gammal hen är, vad hen jobbar med och självklart vad den heter.

2) Räkna ut

1) vilken som var populäraste grenen
2) De tre som tävlade mest
3) Om någon blev hemskickad mer än en gång?

3) Berätta

Berätta i en fin sammanhängande text vad som hände varje vecka, vem var storbonde, vem åkte ut och om man vill hur många som tittade denna vecka.

4) Statistik

Ordna veckorna efter där flest människor har tittat. Skriv även ut vilket avsnitt som var mest populärt och minst per vecka och totalt.

Referenser

Automate the boring stuff with python, Kap 12 Working with Excel Spreadsheets Al Sweigart

SERIOUS PY THON, kap 8 Functional Programming.Julien Danjou