نزهة مع

بايثون

نزهة مع بايثون

ألفه : محمد الغافلي

thebsom@hotmail.com

تم بحمد الله بتاريخ

٢٨/١٢/٢٠١٢

٢٠١٢/١٢/٢٨

الافتتاحية

خططت حين كتابتي لكتاب "مقدمة في البرمجة عن طريق لغة بايثون" أن لا يكون ذلك الكتاب آخر كتاباتي في اللغة ، الكتاب و إن كان بداية لتعلم البرمجة فهو بالتأكيد ليس كل شيء و خطتي كانت أن أكتب سلسلة كتب تقسم تعلم البرمجة إلى عدة مراحل و تمر عليها واحدة واحدة ، قسمت تعلم البرمجة إلى مقدمة في البرمجة ، التعود على البرمجة ، تعلم قواعد لغة برمجة و أخيرا تعلم مكتبات اللغة.

كل مرحلة من المراحل السابقة مهمة و من الصعب بدء مرحلة دون المرور بالمرحلة التي تسبقها ، هذا الكتاب يهتم بالمرحلة الثانية و هي التعود على البرمجة ، من خلال حديثي مع طلاب البرمجة أجد أن التعود على البرمجة من أكثر المشاكل التي تواجههم ، كثيرا ما أصادف طلاب لديهم مقدمة في البرمجة و يعرفون الأفكار العامة لكن يصعب عليهم تطبيقها ، دائما يواجه هؤلاء مشاكل في المفاهيم المتقدمة في البرمجة بسبب عدم تعودهم حتى على المفاهيم الأساسية.

لا يمكن أن تتقدم أخي المتعلم إلا بعد التعود على البرمجة بشكل عام و التعود يكون بالممارسة ، لذلك تمت كتابة هذا الكتاب على شكل مشروع يتم حله على أجزاء ، في كل جزء هناك أمر جديد لكن المهم في كل درس هو الممارسة و التعود على أسلوب الحل و ليس تعلم أمور جديدة ، من المفيد أن تقوم بكتابة برنامج الدرس و حتى محاولة كتابته بطريقة مختلفة.

في نهاية هذه الافتتاحية أحمد اللهـ العلي العظيم الذي وفقني لكتابة هذا العمل ، أشكر والدتي الحبيبة على كل ما قدمت لي خلال حياتي منذ كنت طفلا صغيرا ، الشكر أيضا كل من ساهم في تشكيل معرفتي في البرمجة.

محمد الغافلي
thebsom@hotmail.com

نبذة عن الكاتب

أحب البرمجة و أستخدمها باستمرار ، تعلمت لغة جافا في بداياتي و بعدها تعلمت لغة سي++ ، استخدمت لغة سي++ لعدة سنوات و قمت بكتابة سلسلة دروس فيها ، تعلمت بعدها لغة بايثون و وجدت فيها لغة مناسبة جدا للمستخدم المحترف و المبتدئ ، قررت كتابة سلسلة كتب في هذه اللغة تأخذ طالب البرمجة لدخول هذا العالم على مراحل ، ابتدأت السلسلة بكتاب عنوانه "مقدمة في البرمجة عن طريق لغة بايثون" مثّل المرحلة الأولى و هي إعطاء مقدمة في البرمجة و تعريف معناها للدارس ، معظم البرامج التي أكتبها برامج شخصية أستفيد منها للقيام ببعض المهام بشكل أسرع ، من التجارب التي أحببتها خلال دراستي الجامعية قيامي بتصميم معالج و تصميم لغة تجميع و مجمع (Assembler) له.

مقدمة الكتاب

نزهة مع بايثون ، حين تخرج مع آخرين في نزهة فإنك مع الوقت تكسر حواجز الخجل و الرسميات بينكم ، تصبحون أصحابا و تعتادون على بعضكم ، بالمثل يساعدك هذا الكتاب لكسر الحواجز بينك و بين بايثون لتعتاد عليها و تصبح أمرا طبيعيا بالنسبة لك ، لا يكفي في البرمجة أن تحفظ أفكارها فقط بل التعود على هذه الأفكار يجعلها دائما تمر على بالك في الوقت المناسب.

الكتاب يدور حول مشكلة نحاول كتابة برنامج لحلها ، المشكلة أن لدينا روابط كثيرة و نحن نريد رابطا واحدا منها ، سنقوم خلال هذا الكتاب بالبحث عن هذا الرابط و إيجاده ، عملية البحث و استخراج الرابط من بين روابط كثيرة تمر على عدة مراحل ، في كل درس نقوم بمرحلة منها و نشرح كيفية حلها.

هذا الكتاب يفترض أن الدارس عارف بلغة بايثون بشكل عام و كيفية كتابة البرامج و تشغيلها و هو مبني بالتحديد على محتوى كتاب مقدمة في البرمجة عن طريق لغة بايثون ،يركز هذا الكتاب على التطبيق و كتابة البرامج لتقوم بمهمة معينة و يشرح بشكل سريع بعض الأوامر و الدوال المستخدمة للقيام بالمهمة.

الإصدار الثالث من لغة بايثون هو أحدث إصدار وقت كتابة هذا الكتاب ، بدأ هذا الإصدار من اللغة بالانتشار لكن الإصدار الثاني ما زال منشرا بشكل واضح ، الكتاب يستخدم الإصدار الثالث من اللغة لأنه يعتبر مستقبل اللغة و نتوقع أن يستخدمه الدارس إذا قرر الاستمرار في تعلم لغة بايثون.

الدرس الأول من الكتاب يشرح المشكلة التي نريد كتابة برنامج يحلها و خلال خمسة دروس تليه جزء من المشكلة في كل درس ، كل درس مقسم إلى أقسام يتم كتابة جزء من البرنامج في كل قسم منها.

يحتوي الكتاب على روابط لبعض الملفات التي يتم تحميلها من شبكة الإنترنت ، يتم استخدام هذا الخط لتعليم النص الذي يحتوي على رابط ، *يتم تعليم بعض الأمور المهمة في الدرس بهذا الخط* ، معظم الدروس ستحتوي على أوامر بلغة بايثون ، سيتم كتابة هذه الأوامر بالخط التالي :

```
print ( 'this is a python program' )
```

أسلوب التلوين المستخدم للأوامر سيكون مشابها للتلوين المستخدم في برنامج جي إدت (gedit).

هذا الكتاب يعتبر إحدى الخطوات في المسيرة البرمجية للدارس ، بعد الانتهاء من هذه الدروس بإمكان الدارس الإكمال في تعلم خصائص لغة بايثون و كيفية الاستفادة منها أكثر ، إن أراد الدارس بإمكانه التحول إلى لغة أخرى و سيلاحظ التشابه الكبير بين الأفكار في لغة بايثون و في مختلف اللغات و يتعلم الاختلاف بينها.

الفهرس

الدرس الأول

ما هي المشكلة ؟

هذا الدرس يشرح المشكلة التي نحاول حلها ، لدينا مهمة معينة يصعب علينا القيام بها يدويا لذلك نريد كتابة برنامج يقوم بها بشكل تلقائي و سريع ، الدرس أيضا يعطينا بعض المعلومات التي تساعدنا في حل المشكلة ، التمرين يحتاج منك تحميل ملف يحوي قائمة روابط إنترنت ، بإمكانك تحميل الملف من هذا الرابط.

١.١ ما هي المشكلة ؟

لدينا قائمة روابط إنترنت لملفات كثيرة ، أحد هذه الملفات هو صورة لطفل صغير يلبس ملابس على شكل بطريق ، لا نعرف أي ملف من هذه الملفات هو الصورة التي نريدها و من الصعب علينا تحميل كل الملفات و فتحها واحدا واحدا فالروابط كثيرة و قد يكون اتصالنا بطيئا ، نريد أن نكتب برنامجا يقرأ الروابط و يعطينا الروابط القليلة من المحتمل أن تكون الصورة المطلوبة من بينها و يترك الروابط البقية التي يعرف أن الصورة ليست منها ، يقوم البرنامج بهذه المهمة اعتمادا على بعض المعلومات عن اسم الملف الذي نريده.

١.٢ اسم الملف المطلوب

لا نعرف اسم الملف المطلوب بالتحديد لكننا نعرف بعض المعلومات عنه و هي كما يلي :

١. الملف هو صورة و ينتهي اسمه بامتداد *.gif* أو *.jpg* أو *.png* ، هناك صيغ صور أخرى لكننا في هذا التمرين لم نستخدم غير الصيغ المذكورة.
٢. اسم الملف يحوي الكلمة Penguin ، لا نعرف إذا كانت الكلمة في اسم الملف مكتوبة بالأحرف الانجليزية الكبيرة أو الصغيرة.

٣. حجم الملف بين ١٠٠ كيلو بايت و ٣٥٠ كيلو بايت لكننا لا نعرف الحجم بالضبط ، اسم الملف ينتهي بنقطة ثم حجم الملف ثم نقطة ثم امتداد الملف ، مثلا أحد الملفات اسمه Ubuntu.*4M*.pdf ، لاحظ كيف أن الاسم ينتهي بـ *4M*.pdf أي أن حجم الملف هو ٤ ميغا بايت ، بعد حجم الملف نجد نقطة ثم امتداد الملف و هو pdf ، حجم الملف مكتوب بالكيلو بايت أو بالميغا بايت.

١.٣ ماذا سيفعل برنامجنا

البرنامج سيقوم بقراءة الروابط كلها ثم طباعة روابط الملفات التي تستوفي ما سبق ، أي أنه يطبع فقط روابط الملفات التي لها امتداد *.gif* أو *.jpg* أو *.png* و أيضا يحوي اسمها كلمة Penguin و أيضا حجمها بين ١٠٠ كيلو بايت و ٣٥٠ كيلو بايت ، سنقوم بجزء من هذه المهمة في كل درس من الدروس القادمة.

١.٤ تعلمنا في هذا الدرس

- المشكلة التي نحاول حلها : تحميل الصورة التي تحوي طفلا يرتدي ملابس على شكل بطريق.
- معلومات عن اسم الملف الملف الذي نريده :
  - امتداده *.gif* أو *.jpg* أو *.png*..
  - اسم الملف يحوي الكلمة Penguin.
  - حجم الملف بين ١٠٠ كيلو بايت و ٣٥٠ كيلو بايت.

في الدرس القادم سنتعلم كيف نقرأ الروابط من ملف قائمة الروابط.

الدرس الثاني

قراءة الروابط من الملف

في هذا الدرس سنفتح الملف الذي يحوي قائمة الروابط التي نبحث فيها ، سنقوم بقراءة الروابط واحدا واحدا و طباعتها على الشاشة ، في الدروس القادمة سنستخرج أسماء الملفات من كل رابط لنبحث عن الملف الذي نريده.

٢.١ قراءة الروابط من الملف

الملف يحوي روابط بالملفات التي نريد أن نبحث فيها ، كل سطر من الملف يحوي رابطا واحدا فقط ، في هذه الحالة نستطيع قراءة الأسطر واحدا واحدا ، نستطيع حينها استخراج اسم الملف من كل رابط.

نستطيع تلخيص الكلام السابق في أربع نقاط ، الأولى فتح الملف و الثانية أن قراءة سطر واحد من الملف ، النقطة الثالثة هي الاستمرار في قراءة سطر واحد كل مرة و النقطة الرابعة هي التوقف عن القراءة إذا انتهى الملف.

- لفتح الملف *نستخدم الدالة open* و نكتب داخلها اسم الملف ، الملف الذي نريد قراءته هو url.txt ، نستطيع فتحه بهذا الأمر :

```
infile = open ('urls.txt')
```

- لقراءة سطر واحد من الملف *نستخدم الدالة readline* من المتغير infile :

```
line = infile.readline ()
```

المتغير line يحوي السطر الذي قمنا بقراءته.

- للاستمرار في القراءة *نكتب حلقة* نقرأ فيها سطرا واحدا من الملف ، بهذا الشكل نستمر في القراءة دون توقف ، نستطيع أن نكتب الحلقة بهذا الشكل :

```
while condition :
    line = infile.readline ()
```

علينا أن نكتب شرط الحلقة مكان كلمة condition.

- نريد التوقف عند الوصول لنهاية الملف ، الدالة readline ترجع نصا فارغا إذا وصلنا لنهاية الملف ، لأننا نحفظ نتيجة readline في المتغير line يمكننا استخدامه في شرط الحلقة ، *إن كانت قيمة line تساوي نصا فارغا نوقف الحلقة* ، أما إذا لم تكن كذلك ننفذ الحلقة ، بإمكاننا كتابة شرط الحلقة بهذا الشكل :

```
  line != ''
```

هذا الشرط صائب إذا كان line يحوي شيئا و عندها يتم تنفيذ الحلقة ، يصبح الشرط خاطئا إذا كان line لا يحوي شيئا و عندها نعرف أننا وصلنا لنهاية الملف و يتم إيقاف الحلقة.

بتجميع النقاط كلها يصبح برنامجنا حتى الآن بهذا الشكل :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = infile.readline ()
```

هناك بعض الملاحظات على البرنامج ، الملاحظة الأولى هي أن البرنامج يقرأ أسطر الملف دون أن يقوم بشيء ، في الوقت الحالي سنقوم بطباعة الرابط على الشاشة فقط و في الدروس القادمة سنعدل البرنامج حتى يقوم بالمطلوب ، نستطيع الطباعة باستخدام الدالة print في بداية الحلقة و يصبح البرنامج بعدها بهذا الشكل :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    print (line)
    line = infile.readline ()
```

الملاحظة الثانية هي أننا نقرأ سطرا قبل الحلقة و بعدها ندخل الحلقة ، في الحلقة ننفذ الأوامر على الرابط ثم نقرأ الرابط التالي ، قراءتنا لسطر واحد قبل الحلقة سببه أننا لفحص شرط الحلقة (line != '') علينا أولا أن نعطي قيمة للمتغير line فلا يمكن أن نفحص المتغير قبل إعطاءه أي قيمة.

٢.٢ تعلمنا في هذا الدرس

- فتح الملف.
- قراءة سطر واحد من الملف.
- الاستمرار في القراءة حتى الوصول لنهاية الملف.

في الدرس القادم سنتعلم كيفية استخراج اسم الملف من كل رابط ، بدل أن نطبع الروابط على الشاشة سنقوم فقط بطباعة اسم كل ملف.

الدرس الثالث

فصل اسم الملف عن الرابط

في الدرس السابق قرأنا في حلقة كل الروابط الموجودة في الملف ، في هذا الدرس سنفصل اسم الملف عن الرابط و سنطبع اسم الملف فقط.

٣.١ حذف السطر الجديد من الرابط

استخدمنا في السطر الماضي دالة readline لنقرأ سطرا واحدا من الملف ، هذه الدالة لا تعطينا النص المكتوب في السطر فقط بل أيضا رمز السطر الجديد في نهايته ، برنامجنا للدرس الماضي لم يكن يطبع رابطا في كل سطر بل كان يفصل بين كل رابط و آخر بسطر فارغ ، سبب ذلك أن أمر print يضيف سطرا جديدا في نهاية الطباعة و المتغير line أيضا يحوي سطرا جديدا آخر.

نستطيع حذف السطر الجديد من الرابط *باستخدام الدالة strip* ، نستخدم دالة strip بهذا الشكل :

```
line = line.strip ('\n')
```

تقوم دالة strip بحذف الرمز الذي نكتبه داخلها من بداية و نهاية النص و تعطينا نسخة معدلة منه ، في الأمر السابق كتبنا رمز السطر الجديد في الدالة و تقوم الدالة بحذف السطر الجديد من نهاية النص و بهذا الشكل نتخلص منه.

٣.٢ استخراج اسم الملف

لاستخراج اسم الملف علينا أولا معرفة تركيب الرابط ، الروابط عادة ما تكون أجزاء يفصل بين الجزء و الآخر منها *خط مائل /* ، اسم الملف هو آخر جزء من الرابط ، أول ما علينا فعله هو فصل أجزاء النص عن بعضها ، نستطيع القيام بذلك *باستخدام الدالة split* ، نستخدم دالة split بهذا الشكل :

```
name = line.split ('/')
```

دالة split تفصل أجزاء النص عن بعضها و ترجعها على شكل قائمة

نصوص ، الرمز الذي نكتبه داخل الدالة هو الفاصل بين أجزاء النص و استخدمنا هنا الرمز / لأنه الفاصل بين أجزاء الرابط ، خزنا النتيجة في متغير جديد اسمه name يمثل اسم الملف الذي نريد فحصه.

نتيجة الدالة السابقة هي قائمة مشابهة للتالي :

```
['https:', '', 'example.com', 'path', 'filename.png']
```

القائمة السابقة هي مثال لا أكثر ، القائمة التي ستظهر لنا فيها عناصر أكثر و لا نعرف عدد العناصر بالضرورة ، نحن نريد العنصر الأخير فقط.

عادة يتم ترقيم العناصر ابتداء من العدد صفر لكننا نستطيع استخدام الأعداد السالبة لترقيم العناصر من النهاية ، مثلا العنصر الأخير رقمه -١ ، العنصر قبل الأخير رقمه -٢ و هكذا ، للحصول على العنصر الأخير من القائمة نقوم فقط *بتحديد العنصر -١ من القائمة* بهذا الشكل :

```
name = name[-1]
```

السطر السابق يخزن قيمة آخر عنصر من القائمة -وهو اسم الملف- في المتغير name ، بهذا الشكل نكون قد استخرجنا اسم الملف من السطر دون أن نعرف رقمه من الأمام ، يبقى لنا التأكد إن كان يطابق شروط اسم الملف الذي نبحث عنه ، في هذا الدرس سنكتفي بطباعة اسم الملف و نترك البحث عن الملف المطلوب للدروس القادمة :

```
print (name)
```

يصبح برنامجنا بعد التعديل كما يلي :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = line.strip ('\n')

    name = line.split ('/')
    name = name[-1]

    print (name)
    line = infile.readline ()
```

البرنامج السابق يطبع فقط أسماء الملفات على الشاشة ليس الروابط ، قمنا بذلك لأن الروابط كثيرة و طويلة و يصعب علينا قراءتها ، سنقوم بطباعة الروابط بعد الانتهاء من كتابة البرنامج كاملا في نهاية الكتاب.

٣.٣ تعلمنا في هذا الدرس

- حذف السطر الجديد من نهاية النص.
- تقسيم النص إلى أجزاء اعتمادا على رمز يفصل بين الأجزاء.
- اختيار آخر عنصر من القائمة.

في الدرس القادم سنتعلم فحص امتداد الملف و وجود كلمة Penguin في اسمه.

الدرس الرابع

فحص الامتداد

في الدرس السابق استخرجنا اسم الملف من كل رابط ، كما ذكرنا في الدرس الأول فإن الملف الذي نبحث عنه صورة و امتداده إما jpg أو gif أو png ، في هذا الدرس سنفحص امتداد الملفات قبل طباعتها على الشاشة و سنطبع فقط اسماء الملفات ذات الامتدادات المذكورة ، بهذا الشكل نكون قد قللنا من الاحتمالات التي نتوقع أن يكون اسم الملف من بينها.

٤.١ فحص نهاية اسم الملف

امتداد الملف دائما يكون في النهاية ، بإمكاننا فحص نهاية النصوص باستخدام الدالة endswith ، نعطي هذه الدالة مقطعا نصيا و الدالة تعطي نتيجة صائبة إذا كان النص ينتهي بمقطع الفحص ، مثلا لو أردنا أن نفحص إذا كان اسم الملف ينتهي بالمقطع jpg. نكتب :

```
name.endswith('.jpg')
```

الأمر السابق يرجع لنا قيمة صائبة إذا كان اسم الملف ينتهي بالامتداد jpg.

٤.٢ فحص أكثر من شرط

لسنا متأكدين من امتداد الملف بالضبط ، قد يكون jpg أو gif أو png لكنه ليس من خارج الاحتمالات الثلاثة المذكورة ، بإمكاننا فحص كل الاحتمالات و استخدام عملية *أو (or)* بين كل فحص و آخر ، عملية أو تأخذ عبارتين تعطيان نتيجة منطقية (إما صائبة أو خاطئة) و تصبح النتيجة النهائية صائبة إذا كانت أي عبارة منهما صائبة ، نستطيع أن نكتب أي عدد من العبارات و يجب أن نكتب كلمة or بين كل عبارة و أخرى ، نقوم بذلك بهذا الشكل :

```
name.endswith('.jpg') or name.endswith('.gif') or
name.endswith('.png')
```

نتيجة العبارة السابقة صائبة إذا كانت أي واحدة من نتائج الفحوص صائبة ، أي لو كان الاسم ينتهي بالامتداد jpg أو بالامتداد gif أو الامتداد png ،

تلاحظ أن الأمر السابق طويل و يأخذ سطرين و قد يكون صعب القراءة ، بإمكاننا ترتيب الأمر بشكل أجمل بأن نكتب كل عملية فحص في سطر ، إذا أردنا تقسيم عبارة ما في أكثر من سطر علينا أن نكتب العبارة بين قوسين كما يلي :

```
(name.endswith('.jpg') or
 name.endswith('.gif') or
 name.endswith('.png'))
```

لاحظ كيف بدأنا الأمر بقوس و أغلقنا القوس في نهاية الأمر ، عملية الفحص بهذا الشكل أسهل في القراءة.

٤.٣ طباعة الأسماء التي تطابق الشروط

بعد أن قمنا بعملية الفحص نستطيع استخدام نتيجة الفحص (صائبة أو خاطئة) لمعرفة الأسماء التي تطابق الشروط ، بدل أن نقوم بعملية الطباعة مباشرة نستخدم عبارة if لفحص الشرط ، نطبع اسم الملف فقط إذا كان الشرط صائبا ، نكتب عبارة if بهذا الشكل :

```
if (name.endswith('.jpg') or
        name.endswith('.gif') or
        name.endswith('.png')) :
    print (name)
```

لاحظ كيف أننا كتبنا الشرط بشكل يسهل علينا الفريق بينه و بين الأمر داخل عبارة if و هو أمر الطباعة print ، عبارة if السابقة تطبع فقط أسماء الملفات التي تنتهي بامتداد jpg أو gif أو png ، نستطيع استخدام هذه العبارة داخل الحلقة مكان طباعة اسم الملف أي بدل أن نطبع اسم الملف مباشرة نفحص أولا امتداده و نطبع الاسم إذا كان الامتداد يطابق الشروط ، يصبح البرنامج بهذا الشكل :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = line.strip ('\n')
```

```
    name = line.split ('/')
    name = name[-1]

    if (name.endswith('.jpg') or
            name.endswith('.gif') or
            name.endswith('.png')) :
        print (name)
    line = infile.readline ()
```

البرنامج السابق يطبع فقط أسماء الملفات التي تطابق أحد الامتدادات المذكورة في الشرط.

٤.٤ تعلمنا في هذا الدرس

- فحص نهاية اسم الملف.
- فحص أكثر من شرط.
- طباعة الأسماء التي تطابق الشروط.

من الخصائص التي نعرفها عن اسم الملف الذي نريده هي أنه يحوي كلمة Penguin ، في الدرس القادم سنتعلم فحص وجود كلمة Penguin في أسماء الملفات و سنطبع فقط الملفات التي تطابق الشروط المذكورة في هذا الدرس و أيضا تحوي كلمة Penguin.

الدرس الخامس

البحث عن كلمة

في الدرس السابق استخرجنا أسماء الملفات التي تنتهي بالامتدادات jpg أو png أو gif ، ذكرنا في الدرس الأول أن اسم الملف الذي نريده يحوي كلمة penguin ، في هذا الدرس سنقوم بالبحث عن هذه الكلمة و نطبع فقط أسماء الملفات التي توافق هذا الشرط و الشروط السابقة.

٥.١ تجاهل حالة الأحرف

يتم التفريق في البرمجة بين الأحرف الانجليزية الصغيرة و الأحرف الكبيرة ، نريد البحث عن كلمة penguin لكننا لا نعرف حالة أحرفها فقط تكون على شكل penguin أو Penguin أو PENGUIN أو أي شكل آخر ، من الصعب علينا فحص كل الاحتمالات لاسم الملف لكننا نستطيع بدل ذلك أن نحول جميع الأحرف إلى أحرف صغيرة و نفحص فقط كلمة penguin ، نقوم بهذا باستخدام الدالة lower ، لو أردنا تغيير كل أحرف المتغير name إلى صغيرة نكتب ما يلي :

```
small = name.lower ()
```

الأمر السابق يعطينا نفس المتغير name لكن مع الأحرف الصغيرة ، لا يحدث أي تغيير في المتغير name لكننا نحصل على نسخة منه لكن بأحرف صغيرة و نستطيع حفظها في متغير جديد ، في هذا المثال أسمينا المتغير الجديد small ، بعدها نستطيع التأكد من وجود كلمة penguin في المتغير small بالحروف الصغيرة فقط و إذا وجدنا الكلمة نعرف أنها موجودة في المتغير name حتى لو لم تكن أحرفها كلها صغيرة.

عند وجود أحرف صغيرة في المتغير name أو رموز مثل الرمز السفلي (_) أو علامة ناقص أو زائد لا يحصل تغيير على أي منها عند استخدام الدالة lower بل يتم فقط تغيير الحروف الكبيرة إلى صغيرة ، هناك دالة تشبه لدالة lower و هي upper ، تقوم هذه الدالة *بتحويل الأحرف إلى الحالة الكبيرة* بدل الصغيرة.

٥.٢ فحص وجود كلمة داخل نص

في القسم السابق حولنا أحرف المتغير name إلى الحالة الصغيرة و حفظنا النص الجديد في المتغير small ، نريد الآن التأكد من وجود كلمة penguin في المتغير ، نستطيع القيام بهذا باستخدام عملية in ، نستخدم هذه العملية بهذا الشكل :

```
'penguin' in small
```

عملية in لها أكثر من استخدام في لغة بايثون ، نستطيع استخدامها مع النصوص لفحص وجود نص معين داخل نص آخر ، في المثال السابق نفحص وجود النص الأول penguin داخل النص الثاني و هو المتغير small ، إذا كانت كلمة penguin موجودة في المتغير small تكون نتيجة العملية صائبة أما إذا لم تكن الكلمة موجودة في المتغير تكون النتيجة خاطئة ، نستطيع استخدام هذا الشرط في جملة if بالإضافة إلى الشروط السابقة ، بنفس الطريقة أيضا نستطيع فحص الامتداد للمتغير small لأن الامتداد قد لا يكون بأحرف صغيرة دائما ، مثلا قد يكون JPG بدل jpg في المتغير name ، نستطيع أن نستخدم المتغير small في عملية فحص الامتداد لأن الامتداد في هذا المتغير سيكون دائما بالأحرف الصغيرة.

في الدرس السابق فحصنا وجود واحد من عدة امتدادات للملف و استخدمنا عملية or ، هذه المرة نريد أن يكون أحد الامتدادات موجودا و أيضا تكون كلمة penguin داخل اسم الملف أي أن لدينا شرطان يجب أن يكونا صائبين معا ، هنا نستخدم عملية و *(and)* ، شرطنا في الدرس السابق كان كما يلي :

```
(name.endswith('.jpg') or
 name.endswith('.gif') or
 name.endswith('.png'))
```

لاحظ أن الشرط موجود بين قوسين ، بعد إضافة فحص كلمة penguin يكون شرطنا النهائي كما يلي :

```
('penguin' in small and
(small.endswith('.jpg') or
 small.endswith('.gif') or
 small.endswith('.png')))
```

لاحظ أننا وضعنا شروط الدرس السابق بين قوسين لأنها في الحقيقة تشكل شرطا واحدا و هو أن اسم الملف ينتهي بأحد امتدادات الصور ، أيضا وضعنا الشرط الكلي أي فحص الامتداد و فحص كلمة penguin بين قوسين لأننا كتبنا لاشرط في أكثر من سطر.

بعد أن حددنا الشرط و كيفية فحصه سنعدل برنامج الدرس السابق ليفحص كلمة penguin أيضا و يطبع أسماء الملفات التي توافق الشروط ، يصبح برنامجنا بعد ذلك كما يلي :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = line.strip ('\n')

    name = line.split ('/')
    name = name[-1]

    small = name.lower()

    if ('penguin' in small and
            (small.endswith('.jpg') or
             small.endswith('.gif') or
             small.endswith('.png'))) :
        print (name)

    line = infile.readline ()
```

٥.٣ تعلمنا في هذا الدرس

- تجاهل حالة الأحرف.
- فحص وجود كلمة داخل نص.

في الدرس القادم سنتعلم استخراج حجم الملف ، ذكرنا في الدرس الأول أن الاسم يحوي أكثر من مقطع و أن حجم الملف هو أحد هذه المقاطع ، سنستخرج هذا المقطع من الاسم.

الدرس السادس

استخراج حجم الملف

في الدرس السابق استخرجنا أسماء الملفات التي تحوي كلمة penguin ، في هذا الدرس سنقوم باستخراج حجم الملف ، أسماء الملفات التي نريد البحث فيها تحوي حجم الملف لذلك سنبحث في اسم الملف عن حجمه.

٦.١ تقسيم النص

ذكرنا في الدرس الأول أن اسم الملف يتكون من ثلاثة مقاطع ، الأول هو عبارة ، الثاني هو حجم الملف و الثالث هو امتداد الملف ، يفصل بين كل مقطع و آخر نقطة ، أحد الملفات التي نبحث فيها اسمه *Gens_2.16.6.95_i386.1M.deb* ، المقطع الأول هو Gens_2.16.6.95_i386 ، لاحظ أن هذا المقطع يحوي نقاطا لكنه مقطع واحد ، المقطع الثاني هو حجم الملف و هو 1M أي ١ ميغا بايت ، المقطع الأخير هو امتداد الملف و هو deb ، من السهل علينا معرفة حجم الملف لكننا نريد من الحاسب أن يفحص يستخرج حجم الملف لكل الملفات ، نستطيع أن نقسم نصا إلى قائمة نصوص باستخدام الدالة split ، نستخدم الدالة بالشكل التالي :

```
splitted = name.split ('.')
```

الأمر السابق يقسم name إلى عدة أقسام و يعطينا قائمة بها ، قمنا بتخزين القائمة في المتغير splitted ، النص الذي نكتبه بين القوسين هو الفاصل بين كل قسم و قسم من أقسام name ، ذكرنا أن كل جزء من اسم الملف مفصول عن الأجزاء الأخرى بنقطة لذلك كتبنا نقطة بين القوسين في الدالة split ، القائمة المخزنة في splitted تصبح بهذا الشكل :

```
[ 'Gens_2', '16', '6', '95_i386', '1M', 'deb' ]
```

لاحظ أن الدالة قسمت الاسم إلى أقسام كثيرة ، القسم الأخير هو الامتداد و القسم الذي قبله هو الحجم ، الأقسام الباقية هي المقطع الأول من اسم الملف و المفترض أن يكون قسما واحدا فقط ، المشكلة في هذا المقطع هنا أنه يحوي نقاطا لذلك نجد أن الدالة اعتبرت كل نقطة في هذا المقطع فاصلا بين أقسام الاسم لذلك قسمت هذا المقطع إلى أقسام أيضا.

٦.٢ استخراج الحجم

في القسم السابق قسمنا اسم الملف إلى أجزاء و حفظنا قائمة الأقسام في المتغير splitted ، نتوصل إلى القسم الذي نريده بكتابة قوسين مربعين بعد اسم المتغير و نكتب بين القوسين رقم العنصر الذي نريد التوصل إليه ، ذكرنا في الدرس الثالث أننا نستطيع التوصل لعناصر القائمة من الخلف باستخدام الأعداد السالبة ، نحن نريد استخراج حجم الملف و هو القسم ما قبل الأخير لذلك نستخدم الرقم -٢ بهذا الشكل :

```
splitted[-2]
```

نستطيع باستخدام هذه الطريقة أن نحفظ حجم الملف في متغير :

```
size = splitted[-2]
```

المتغير size يحوي المقطع قبل الأخير من حجم الملف ، في الملف Gens_2.16.6.95_i386.1M.deb قيمة المتغير size تصبح النص 1M.

بعد إضافة أوامر تقسيم اسم الملف و استخراج الحجم يصبح برنامجنا بهذا الشكل :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = line.strip ('\n')

    name = line.split ('/')
    name = name[-1]

    small = name.lower()

    splitted = name.split ('.')
    size = splitted[-2]

    if ('penguin' in small and
            (small.endswith('.jpg') or
             small.endswith('.gif') or
             small.endswith('.png'))) :
        print (name)

    line = infile.readline ()
```

لاحظ أننا لم نقم بفحص حجم الملف بعد ، سنترك ذلك إلى الدرس القادم.

٦.٣ تعلمنا في هذا الدرس

- تقسيم النص.
- التوصل إلى عناصر القائمة من الخلف.

في الدرس القادم سنقوم بفحص حجم الملف و التأكد من أنه ليس كبيرا جدا و لا صغيرا جدا ، سنطبع فقط أسماء الملفات ذات الحجم الذي نتوقعه للملف الذي نبحث عنه.

الدرس السابع

فحص حجم الملف

في الدرس السابق استخرجنا من اسم الملف المقطع الذي يحوي الحجم ، المقطع الذي استخرجناه نص يحوي حجم الملف و بعده حرف K إذا كان الحجم بالكيلو بايت أو حرف M إذا كان الحجم بالميغا بايت ، حجم الملف الذي نريد قراءته بالكيلو بايت لذلك علينا التأكد أن الحرف هو K ، بالإضافة لذلك لا يمكننا مقارنة النصوص بالأرقام بل يجب مقارنة الأرقام مع بعضها.

٧.١ فحص الحرف

في نهاية مقطع حجم الملف الذي استخرجناه حرف يحدد إذا كان الحجم ميغا بايت أو كيلو بايت ، مثلا أحد الملفات اسمه Tulip_flower_bulbs.76K.jpg ، المقطع الذي استخرجناه هو 76K ، من الحرف K نعرف أن الحجم بالكيلو بايت و مقداره ٧٦.

نريد أولا التأكد أن الحرف هو K ، نستطيع التوصل لحروف النص كما نفعل مع القوائم ، نستطيع القيام بالأمر الأول بأن نفحص الحرف الأخير من النص بهذا الشكل :

```
size[-1] == 'K'
```

بقي لنا التأكد أن الرقم بين ١٠٠ و ٣٥٠.

٧.٢ فحص الرقم

أول ما علينا فعله لفحص الرقم هو استخراج الرقم من القسم بدون الحرف K أو M ، لغة بايثون تسمح لنا باستخراج جزء من النص بتحديد أول حرف و آخر حرف نريد استخراجه ، نقوم بذلك بهذه الطريقة :

```
size[0:3]
```

الأمر السابق يعطينا جزءا من النص size يبدأ من أول حرف (رقمه ٠) إلى الحرف الثالث (رقمه ٢) ، لو كان قسم الحجم هو 1002M على سبيل المثال فإن القسم الذي استخرجناه هو 100 ، لاحظ أن الرقم الثاني الذي نحدده أكبر من رقم آخر حرف نستخرجه بواحد ، مثلا إذا أردنا أن نستخرج

الحـروف إلـى الحـرف رقـم ٥ يكـون نكتـب بيـن القوسـين 0:6 ، إذا أردنـا استخراج الأحرف من الحرف رقم ٣ إلى الحرف رقم ٧ نكتب بين القوسين 3:8 ، في برنامجنا نريد استخراج الأحرف من الأول إلى قبل الأخير ، الأول رقمه صفر ، و هذا العدد الذي نحدده أولا ، لا نعرف رقم الحرف قبل الأخير لكننا نستطيع تحديد رقمه من نهاية النص أي -٢ ، لأننا نزيد واحدا على رقم آخر حرف نريد استخراجه فإننا نستخدم الرقم -١ داخل القوسين ، بهذا يكون أمر استخراج الرقم من حجم الملف بهذا الشكل :

```
size[0:-1]
```

القيمة التي استخرجناها قيمة نصية ، لا يمكننا مقارنة القيمة النصية مع القيم العددية حتى لو كانت تحوي أرقاما بل يجب علينا تحويل النص إلى متغير عددي ليمكننا مقارنته.

٧.٣ تحويل النص لعدد

يمكننا تحويل النص إلى عدد باستخدام الدالة int ، نكتب int بعدها قوسان و بين القوسين نكتب القيمة التي نريد تحويلها ، لتحويل الرقم الذي استخرجناه من حجم الملف نكتب الأمر التالي :

```
int (size[0:-1])
```

بهذا الشكل يمكننا مقارنة الحجم بالعدد الذي نريد ، في الدرس الأول ذكرنا أن حجم الملف بين ١٠٠ و ٣٥٠ كيلو بايت ، لا يمكننا في البرمجة التأكد أن القيمة بين عددين بشكل مباشر لكننا نستطيع فحص أن القيمة أكبر من ١٠٠ و في نفس الوقت أصغر من ٣٥٠ ، نقوم بذلك كما يلي :

```
int (size[0:-1]) > 100 and int (size[0:-1]) < 350
```

بهذا الشكل نكون قد تأكدنا أن القيمة تقع بين ١٠٠ و ٣٥٠.

في هذا الدرس قمنا بفحص شرطين إضافيين و هما :

```
size[-1] == 'K'
int (size[0:-1]) > 100 and int (size[0:-1]) < 350
```

الشروط التي فحصناها في الدروس السابقة هي :

```
('penguin' in small and
            (small.endswith('.jpg') or
             small.endswith('.gif') or
             small.endswith('.png')))
```

بعد إضافة الشرطين الجديدين يصبح الشرط كاملا كما يلي :

```
('penguin' in small and
             size[-1] == 'K' and
             int (size[0:-1]) > 100 and
             int (size[0:-1]) < 350 and
            (small.endswith('.jpg') or
             small.endswith('.gif') or
             small.endswith('.png')))
```

بهذا نكون قد تأكدنا من كل الشروط في اسم الملف و نستطيع طباعة الروابط التي نريدها فقط ، بعد هذا سنجعل البرنامج يطبع روابط الملفات بدل أسمائها و نستطيع أن ننسخ الروابط و نحمل ملفاتها لنجد الملف الذي نريده.

يصبح البرنامج بعد التعديل كما يلي :

```
infile = open ('urls.txt')

line = infile.readline ()
while line != '' :
    line = line.strip ('\n')
    name = line.split ('/')
    name = name[-1]
    small = name.lower()

    splitted = name.split ('.')
    size = splitted[-2]

    if ('penguin' in small and
             size[-1] == 'K' and
             int (size[0:-1]) > 100 and
             int (size[0:-1]) < 350 and
            (small.endswith('.jpg') or
             small.endswith('.gif') or
             small.endswith('.png'))) :
        print (line)

    line = infile.readline ()
```

ستظهر لنا روابط معدودة من بين الروابط الكثيرة و من السهل علينا تحميلها و البحث عن الملف الذي نريده لقلة عددها ، في الدرس الأول ذكرنا أن الملف هو صورة لطفل صغير يلبس ملابس على شكل بطريق.

٤.٧ تعلمنا في هذا الدرس

- استخراج جزء من نص.
- تحويل نص يحوي أرقاما إلى رقم.

بهذا نصل إلى نهاية سلسلة نزهة مع بايثون.

خاتمة الكتاب

هذا الكتاب كان عبارة عن تمرين يساعد الدارس على ممارسة البرمجة و التعود عليها مع القليل من الأمور الجديدة ، هدفه كان فقط أن يجعل من البرمجة أمرا اعتيادي للشخص ، في نهاية هذا الكتاب كما في الكتاب السابق نجيب عن هذه الأسئلة : ماذا بعد و ما هي الخطوة التالية ؟

المرحلة التالية في مسيرتك البرمجية هي إتقان إحدى لغات البرمجة ، هذا الكتاب لم يعلمك الكثير من الأمور في لغة بايثون بل اعتمد بشكل كبير على معرفتك السابقة بهذه اللغة ، تحتاج دروسا مختصرة في ميزات اللغة و أوامرها المختلفة ، بمعرفتك الحالية باللغة لن تحتاج إلى شرح مفصل لكل أمر بل يكفيك شرح سريع مع مثال ، بإمكانك الاطلاع على الدروس الرسمية التي يقدمها موقع بايثون باللغة الانجليزية من هذا الرابط و هو يحتوي على الكثير من المواضيع التي قد تعرف بعضها ، ننصحك بالبقاء مع لغة بايثون لكن إن اخترت لغة أخرى فاستمر معها و تجنب تعلم لغات كثيرة في وقت واحد فهذا سيشتت تفكيرك و لن يفيد كثيرا ، بإمكانك أيضا تعلم الوحدات المتوفرة في لغة بايثون ، إذا كنت مهتما بأمر معين في البرمجة فابحث عن الوحدة التي تقوم بهذه الأمر و تعلمها ، الأفضل أن تزيد معرفتك بلغة بايثون قبل تعلم الوحدات المختلفة لكن بإمكانك تعلم بالوحدات حتى قبل ذلك.

في كتابنا القادم بإذن الله سنركز على لغة بايثون و أوامرها المختلفة بشكل مختصر و نذكر أمثلة لها ، سيساعدك ذلك لتصبح قادرا على البرمجة بشكل أكثر مرونة و سهولة.

في أمان الله
أخوكم محمد الغافلي