بسم الله الرحمن الرحيم

ال البداية

البداية

ملاحظة: لن ينفع نقل الكود نسخ لصق .

سأقوم بشرح أساسيات مكتبة argparse من خلال لغة البرمجة python .

عمل المكتبة التعامل مع البارامترات التي تمرر للبرنامج من سطر الأوامر كهذا المثال في الأسفل حيث يمثل --version البارامتر الممرر أو command line arguments .

```
myprogram.py  --version
```

سأضع مثال وأشرح سطر سطر .

```
import argparse

parser = argparse.ArgumentParser
(prog="myprogram",usage="%(pros)s [-b] [-v]",epilog="",description="")

parser.version = '1.0'

parser.add_argument('-V','--version', action='version')
```

1. في أول سطر قمت بإستدعاء مكتبة argparse .
2. في ثاني سطر قمت بإنشاء كائن object من Class الفئة ArgumentParser وممكن وضع تعريف بالبرنامج في description ليظهر عند عرض المساعدة وفي prog إسم البرنامج وإذا لم يوضع هذا الخيار سيتم تحديد إسم البرنامج تلقائيا بناء على إسم الملف التنفيذي للبرنامج ايضا ممكن تحديد usage لبعض الشرح لطريقة الإستخدام وسيتم تعويض %(prog) بإسم البرنامج ,وهناك أيضا epilog وهي رسالة تظهر في أخر سطر عند عرض المساعدة على العموم ربما المفيد أكثر شيء description وكان هذا مجرد مثال لا تأخذوا به حرفيا.
3. قمت بإضافة attribute لهذا الكائن سميته version وأعطيته قيمة التي ستمثل رقم إصدار البرنامج .
4. الأن أضفت argument أو خيار سمها ما شئت أضفت V- و --version ,أي عندما يمرر المستخدم أي من هذين البارامترين للبرنامج سيتم طباعة فيمة version مع الملاحظة أن حرف V حرف كبير ممكن ان نضع أي شيء لكن أحببت أن ألفت نظرك لحالة الأحرف .

```
import argparse
parser = argparse.ArgumentParser()
parser.version = '1.0‘
parser.add_argument('-V','--version', action='version')
parser.add_argument('-y', '--assumeyes', action="store_true")
args = parser.parse_args()
print(args.assumeyes)
```

1. الجديد أولا أضفت خيار -y و --assumeyes وال action هي store_true بمعنى وبإختصار لو ادخل المستخدم أي منهم سيتم إرجاع True كما سيظهر معنا في الأسفل .
2. من خلال دالة parse_args() سيتم قراءة كل ما مرره المستخدم من سطر الأوامر وحفظ النتيجة في متغير سميته args .
3. بما أنا أضفنا بارامتر --assumeyes وال action كانت store_true في حال قام المستخدم بتمرير -y أو --assumeyes ستكون قيمة args.assumeyes هي True غير ذلك ستكون False مع الملاحظة يوجد أيضا action إسمها store_false وعملها عكس store_true أي سيتكون False في حال تمرير أحدهما .

ملاحظة : للإختصار في باقي الشرح لن اكرر كل أسطر الكود المذكورة في الأعلى إلا ربما في بعض الحالات وساعتبر انها موجودة .

```
parser.add_argument('-v','--verbose', action='count')
```

1. ال action التي تسمى count أي التعداد وعملها تعداد كم مرة تم تمرير -v أو --verbose مثال لنفترض قام المستخدم تمرير -v مرتين -v -v أو حتى هكذا -vv ستكون قيمة args.verbose هي كم مرة تم تمريرها أي معنا هنا مرتين أي رقم 2 .

```
parser.add_argument('-s', '--saveas', action="store",type=str,metavar="SAVE_LOCATION")
```

1. ال action التي تسمى store عملها بشكل بسيط حفظ نتيجة ما يأتي بعد تمرير -s أو --saveas لنفترض لديك برنامج يقوم بتنزيل الصور وأضفت هذا الخيار للمستخدم إذا أراد تحديد مكان حفظ الصور يمكنك أن تتفحص إذا ما كانت قيمة args.saveas فارغة ,إذا لم تكن فارغة مثلا تقوم بتحديد إذا ماكان المسار المعطى موجود وعندها تقرر ماذا تفعل ...إلخ مع الملاحظة المهمة أني حددت النوع type ان يكون نص str أي ما يمرره المستخدم سيحفظ كنص .
2. الmetavar هو نص سيظهر في المساعدة عند عرضها بعد -s لتوضيح أن يدخل المستخدم مسار لحفظ النتيجة.

```
parser.add_argument('-i', '--id',required=True, action="store",type=str)
```

1. الجديد معنا required=True أي هذا الخيار إلزامي يجب على المستخدم تمريره .

```
parser.add_argument('-e', '--years', action="store",default=[2021],nargs='+',type=int)
```

1. الجديد معنا أولا default أي اني أعطيت قيمة إفتراضية في حال لم يقم المستخدم بتمرير شيء .
2. و nargs='+' وعملها تمكين المستخدم من تمرير list بعد --years أو -e مثال 2020 2021 --years مع ملاحظة علامة الزائد + تعني ان في حال وجود هذا الخيار إجباري أن يتم تمرير على الأقل قيمة واحدة بعده, يوجد أيضا علامة الإستفهام ? ومعناها قيمة واحدة لكن إختيارية وليست إجبارية وعلامة النجمة * وعملها ذات عمل + لكن تمرير القيم إختياري وممكن مثلا نريد أن نجبر المستخدم في حال إستخدم هذا الخيار نجبره على إدخال مثلا قيمتين نضع بكل بساطة العدد الذي تريد nargs=2 وعندها المستخدم مجبر إدخال قيمتين مثلا 2020 2021.

```
parser.add_argument('-m', '--maxmounth', action="store",default=12,type=int,choices=range(1, 13))
```

1. سيتم حفظ النتيجة لقيمة عددية من نوع int والقيمة الإفتراضية 12 وملزم المستخدم أن تكون القيمة في حال تمريرها بين رقم 1 و رقم 12 ممكن أن نحدد القيمة بشكل صريح من خلال list وممكن إستخدام range المتوفرة في python .

```
parser.add_argument('-f', action='append_const', const=42)

args = parser.parse_args()

print(args.f)
```

1. أولا نلاحظ ممكن أن نضع شيء يحتوي حرف واحد فقط وعندها ممكن الوصول للقيمة من خلال الحرف بدل الكلمة .
2. عمل ال action المسمى append_const بإختصار عند كل تمرير ل -f سيتم إضافة رقم 42 للقائمة التي ستحفظ في args.f , مثال pyprogrm.py -f -f ستكون قيمة args.f تساوي [42,42] .

```
my_parser.add_argument('-b', action='store_const', const=50,help="Save value 50",dest="my_b")
```

1. عمل ال action التي تسمى store_const شيء مثل عمل append_const لكن لن تكون قائمة list أي عند وجود -b سيتم حفظ قيمةواحدة والتي حدتها بقيمة 50 في args.b .
2. الجديد معنا help ممكن أن نشرح عن -b ليتم عرضه عند عرض المساعدة عند تمرير --help أو -h أو عند حصول خطأما مثلا عندما يدخل المستخدم قيمة نصية والنوع type حدد أن يكون رقم int .
3. خيار dest ممكن أن يغير إسم ال attribute الإفتراضية مثلا سيصبح args.my_b بدل args.b مع الملاحظة في حال كان البارامتر يحتوي علامة الناقص مثلا --id-i عندها للوصول إليه نستبدل علامة الناقص ب علامة _ أي للوصول لقيمته نكتب args.id_i لأن الsyntax الخاصة ببياثون لا تسمح بعلامة الناقص أو ممكن إستخدام كما ذكرت خيار dest .

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('first', action='store')
parser.add_argument('others', action='store',
nargs=argparse.REMAINDER)

args = parser.parse_args()

print('first = %r' % args.first)
print('others = %r' % args.others)
```

1. لنفرتض نريد شيء مبسط أكثر لا نريد تحديد شيء ممكن أن نعمل شيء كهذا ستمثل first أو بارامتر يمرر و others باقي البارامترات الممررة argparse.REMAINDER .

```
# Program file name myprogram.py

import sys

program_file_name = sys.argv[0]
if len(sys.argv) >=3:
    first  = sys.argv[1]
    others = sys.argv[1:]
```

1. في بايثون من دون مكتبة argparse ممكن الوصول لكل شيء ممرر من خلال متغير argv في مكتبة sys وقيمته عبارة عن قائمة أول عنصر منها إسم ملف البرنامج أو السكربت والباقي هي البارامترات الممررة ,في الحالات البسيطة ممكن الإستعانة بهذه الطريقة لكن في الأمور المعقدة سيخرج الأمر عن السيطرة وطالما يوجد شيء جاهز الأفضل إستخدامه.

لمن أراد الشرح بالإنكليزية هذا الدليل منقول كشرح وليس كترجمة حرفية من موقع realpython هناك بعض الأشياء القليلة في الشرح التي لم أتطرق إليها هنا .

https://realpython.com/command-line-interfaces-python-argparse

روابط خاصة بي:

https://github.com/yucefsourani

https://yucefsourani.github.io

https://arfedora.blogspot.com

رابط وثائق المكتبة حيث يوجد الكثير من الأشياء لم تذكر هنا أو في الشرح الإنكليزي مثل وضع دالة في default و غيره لكن هذه الأشياء التي قد يحتاجها المستخدم والتي توفر له كل ما يريده :

https://docs.python.org/3/library/argparse.html