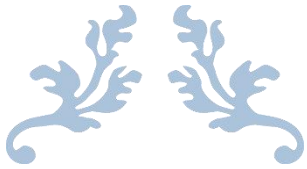


PYTHON

BY SADEK
01127683025

Basics of python

- لعمل طباعة نستخدم امر ()print ويوضع بداخل ال( ) ما سنقوم بطباعته
- امر print يقوم بنزول الى سطر جديد بشكل تلقائى
- ينتهى السطر البرمجى بمجرد انتهاء السطر فى الكود ولاكن فى حالة اننا قمنا بوضع اكثر من امر فى نفس السطر فاننا نقوم بوضع ; لتدل على ان السطر البرمجى قد انتهى
- لعمل comment فى الpython نستخدم # قبل الcomment
- الpython تحتوى على بينات مثل int و float و string و الboolean ويمكن معرفة نوع الdata من خلال type() function
- كل الdata فى الpython عبارة عن object
- Python كما يوجد بها انواع data مثل الfloat و int و string و Boolean فانها تحتوى ايضا على انواع اخرى مثل

Data type	notes	example
list	• هو يشبه الarray فى باقى اللغات ولاكنه ليس array • يتكون من مجموعة من البينات داخل [ ]	[1,2,3,4,5,6]
tuple	• هو يشبه الlist ولاكن مع الكثير من الاختلافات • يتكون من مجموعة من العناصر داخل ( )	(1,2,3,4,5,6)
dictionary	• تتكون عناصره على شكل key و value وتوضع داخل { }	{num1 : 1 , num2 : 2}

- عملية انشاء متغير او variable فى الpython تتم عن طريق كتابة اسم الvariable ثم علامة = ثم القيمة كتالى **variable1=5**
- كتابة اسم المتغير تتم تحت وجود مجموعة من القواعد مثل انها لاتبدا برقم او انها لاتبدا ب special character عدا ال _
- لغة الpython هى لغة sensitive اى انها تفرق بين الحروف الكبيرة والصغيرة
- يمكن ان يحدث overwrite على المتغير اثناء الrun time حتى وان اختلفت نوع الdata كتالى

```
x=1
x="Ali"
```

  - **قيمة x فى النهاية هى Ali**
- يمكن انشاء اكثر من متغير واضافة لهم قيم فى سطر واحد كتالى **a,b,c = 1,2,3**

Escape Sequences character

- Escape character هى مجموعة من الcharacter الى تقوم بوظيفة معينة مثل
  1. **\b** وهى تقوم بعمل back space اى انها تحذف اخر character
  2. **** فقط تقوم بعمل escape للfunction الخاص بالcharacter الذى بعدها
     - على سبيل المثال **\"** تقوم بجعل ال" بدون خصائص فى الكود وانما تكون character عادى
  3. **\n** وهى تقوم بالانتقال الى سطر جديد
  4. **\r** تقوم بالاتيان على اول السطر وتبديل النص المطبوع بالنص الذى بعدها
     - تستخدم فى عمل update للمعلومات المطبوعة
  5. **\t** يستخدم لعمل tab
- لعمل concatenation فى الpython نستخدم +
- لايمكن عمل concatenation بين رقم و string
- Print function يمكن ان تاخذ اكثر من string كا argument وتطبعهم معا يوجد arguments اخرين لتعديل عملية الطباعة مثل :
  1. sep وهو اختصار لseparator ويستخدم لطباعة characters فاصلة بين كل argument والقيمة الdefault له هى المسافة
  2. end وهو اخر character سيتم وضعه بعد عمل print وهو السبب فى جعل الprint تقوم بنزول سطر جديد وهذا لان القيمة الdefault له هى \n

```
(function) def print(
    *values: object,
    sep: str | None = " ",
    end: str | None = "\n",
    file: SupportsWrite[str] | None = None,
    flush: Literal[False] = False
) -> None

Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
```

String

- الstring يمكن كتابته داخل ' ' او داخل " "
- لعمل string محفوظ فيه النص كما هو فى الsource code حتى وان كان مكتوب فى اكثر من سطر نضعهة بين " " " " " " او بين ' ' ' ' ' '
- يمكن الوصول الى حرف فى الstring او عنصر فى الlist او فى الtuple عن طريق استخدام [ ] ووضع ال index بدخلها
- ال Index فى الpython يمكن ان يكون بالسالب
- يمكن عمل slicing اى اخذ اكثر من عنصر كاجزء من خلال استخدام علامة **:** داخل [ ] و وضع البداية والنهاية كتالى **[1:5]**
- النهاية الموضوعة لاتاخذ بمعنى فى المثال السابق سياتى فقط بالعناصر التى لها index 1 و 2 و3 و4
- فى حالة عدم وضع البداية فى الslicing سيبدا من index 0 وفى حالة عدم وضع النهاية سينتهى الى اخر عنصر
- فى حالة عدم وجود نهاية او بداية فى الslicing فانه سيطبع العناصر كاملة
- يمكن اضافة الى الslicing **:** اخرى لتحديد مقدار الstep وتوضع الstep بعدها كتالى **[0:10:2]** وهنا سيطبع الارقام الزوجية فقط
- Len() هى function تقوم بارجاع عدد عناصر الstring او list او tuple
- Strip() هى function تقوم باذالة المسافات فى اول واخر الstring ويوجد منها rstrip() للازالة من اليمين و lstrip() لليسار
- اذا كنا نريد اذالة character اخر غير المسافة نقوم باضافته الى ( ) فى strip function
- zfill(). هو function يقوم باضافة اصفار من الشمال الى الرقم لجعله متساوى فى الحجم مع باقى الارقام على سبيل المثال سيقوم بتحويل 2 الى 02 او الى 002 لكى تكون مناسبة فى الحجم مع الرقم 223 مثلا او 100 ونقوم بوضع الlength المفترض اخراجه داخل ال( ) مثل **"2".zfill(3)**
- lower(). تستخدم لجعل الجملة التى قبلها بالحروف الsmall
- upper(). تستخدم لجعل الجملة التى قبلها بالحروف الcapital
- isupper() و islower(). هم functions يقومو بتحقق اذا كان الstring الذى قبلها مكون من حروف capital او small ويقومو بارجاع قيمة true او false
- index(). هو function يقوم بارجاع الindex الخاص بالجملة او الخاص بالحرف الموضوع داخل ال ( ) وذا تكررت الجملة او الحرف سيقوم بارجاع مكان او جملة او حرف فقط
- Index function اذا لم تجد الكلمة تقوم بعمل error ولاكن توجد function اخرى تقوم بارجاع 1- فى حالة لم تجد العنصر وهى find().
- split(). تقوم بتقسيم الstring الذى قبلها الى اكثر من جزء ووضعهم داخل array وتقوم بتقسيم بناء على المسفات بين الكلمات والحروف
- عندما نريد تقسيم الstring بناء على character اخر غير المسافة نقوم بوضع هذا الcharacter داخل ال( ) كا argument
- يمكن تحديد argument اخر وهو max split وهو اقصى عدد من الsplit التى سيقوم بها الfunction
- توجد function اخرى وهى splitlines() وهى تقوم بارجاع السطور كل سطر هو عنصر فى list
- توجد function اخرى وهى rsplit(). وهى نفس الfunction ولاكن تبدا التقسيم من اليمين
- center(). هو function يقوم باخذ الstring الذى قبله وارجاعه موضوع فى منتصف string اخر عن طريق وضع characters قبله وبعده
- center. ياخذ argument 2 الاول وهو وهو عدد الcharacter المفترض رجوعها و الاخر وهو الcharacter الذى سيوضع فى البداية والنهاية
- مثال على center function كتالى

```
print ("sadek".center(11,"*"))
```

⟶ ***sadek***

- count(). هو function يقوم باخذ الstring داخل ال( ) ويقوم بالبحث عن عدد مرات وجوده فى الstring الذى تم استخدام الfunction عليه
- الstring الذى سيوضع داخل ( ) فى count function يكون case sensitive
- يمكن ان ياخذ الcount function argument 2 اخرين وهم البداية والنهاية الذى سيبحث فهم
- replace().هو function يستخدم لاستبدال كلمة او جملة او حرف فى string ب كلمة اخرى او جملة اخرى او حرف اخر وياخذ قيمتان الاولة هى المراد حذفها والثانية المراد وضعها مكان المحذوفة
- str() هو function يقوم بتغير نوع الdata الموضوعة بداخل ال ( ) الى string
- لايمكن طباعة string و number معا فى جملة print واحدة لذالك يفضل استخدام str() function مع الرقم
- startwith(). هو function يقوم باختبار النص الذى قبله هل يبدا بكلمة معينة وتوضع هذه الجملة داخل ال( ) ويمكن اضافة 2 argument اخرين لتحديد مكان البحث ويقوم بارجاع true او false
- endwith(). وهى العكس الخاص ب startwith()
- isidentifier(). هو function لتحقق من ان string يحقق شروط الidentifier وهى الشروط الموضوعة لكتابة variable مثلا ويقوم بارجاع Boolean
- isalpha(). هو function لتحقق من ما اذا كان الstring يتكون من حروف alpha فقط ام لا ويقوم بارجاع Boolean
- isalnum(). ويتاكد اذا كان الstring يتكون من alpha او numbers ام لا
- join(). هو function العكس من split() حيث انه ياخذ list وتوضع داخل () ويقوم بارجعها string والstring الموضوغ قبل join هو سيكون الفاصل بين كل كلمة والاخرى
- الformatting هى طريقة تنسيق النص والارقام معا وهى طريقة بديلة للconcatenating
- يمكن عمل format باضافة {} بداخل الstring وهى تمثل المكان الذى سيوضع فيه الvaluable او الرقم وبعد نهاية الstring نقوم باضافة format(). ونضع داخل ال() المتغيرات او الارقام بالترتيب الذى نريد وضعه فى الstring
- لتحديد نوع الdata عند الطباعة نقوم بذالك كالتالى
  1. فى الstring نضع s: داخل {} هكذا **{:s}**
  2. فى الinteger نضع d: داخل {} هكذا **{:d}**
  3. فى الfloat نضع f: داخل {} هكذا **{:f}**

- لتحديد عدد القيم التى تظهر من الfloat بعد العلامة العشرية نقوم بعملها كتالى **{:.2f}** وهنا سيقوم بارجاع رقمين عشريين فقط
- هناك طريقة افضل لعمل format وهى من خلال وضع حرف f قبل كتابة الstring و بدلا من وضع {} سنقوم باستخدمها ووضع داخلها الvariable كتالى **{variable}**

Numbers

- فى الpython يوجد نوعان من الnumbers وهم integer و float ويمكن التحويل بينهم بشكل طبيعى
- لتحويل integer الى float نستخدم function ()float ويوضع الرقم المراد تحويله بين ()
- لتحويل الfloat الى integer نستخدم function ()int
- لعمل اس نستخدم **
- ()abs هو function يقوم بعمل محدد للارقام اى انه يجعلها جميعا موجبة
- ()pow هو function يقوم بعمل اس رقم معين وياخذ قيمتان الاولة وهى الاساس والثانية وهى الاس
- يمكن عمل اس لرقم عن طريق استخدام ** بدلا من function ()pow
- ()Max هى function تاخذ اكثر من قيمة وتقوم بارجاع القيمة الاكبر
- ()Min هى function تاخذ اكثر من قيمة وتقوم بارجاع القيمة الاصغر
- ()round هو function يقوم بتقريب العدد الموضوع داخل ( )

List

- عناصر الlist تكون داخل [ ] وتكون مرتبة اى يمكن الوصول لها من خلال الindex
- يمكن اضافة او حذف التعديل فى الlist بشكل طبيعى جدا
- يمكن اذالة اكثر من عنصر ووضع مكنهم عنصر واحد ولا يشترط وضع نفس العدد من العناصر
- يمكن تكرار العناصر الموجودة فى الlist اكثر من مرة ويمكن للعناصر ان تكون بdata type مختلفة
- لاضافة عنصر الى الList نقوم باستعمال امر ()append. على الlist ونضع العنصر داخل ال ( )
- يمكن ان يكون العنصر المضاف الى ا لlist هو List اخرى
- عند اضافة عناصر list الى اخرى نقوم باستعمال امر ()extend. كتالى **a.extend(b)**
  وهنا جميع عناصر الb اصبحت فى الa
- يمكننا عمل نفس المثال السابق كتالى **a+b**
  هنا لم تتغير عناصر الa او الb
- ()remove. هو function يقوم باذالة عنصر من list ويوضع العنصر داخل ال( )
- ()sort. هو function يقوم بترتيب عناصر الlist ولاكن يجب ان يكونو من نفس النوع اولا
- ()reverse. هو function يقوم بعكس ترتيب عناصر الlist
- Reverse ليس العكس الخاص بsort حيث ان reverse يقوم بعكس ترتيب العناصر بغض النظر كانو فى ترتيب تصاعدى او تنازلى ام لا
- ()clear. هو امر يقوم بحذف جميع عناصر الlist
- ()count. هو function يقوم بعد عنصر فى الlist ونقوم بوضع العنصر داخل ( )
- يمكننا استخدام index function على الlist للاتيان بالindex الخاص بعنصر معين
- ()insert. هى function تقوم بوضع عنصر فى Index معين وتقوم بازاحة العنصر الموجود فى هذا الindex وجعله بعده
- ()pop. هو function يقوم باذالة عنصر من الlist وارجاعه وياخذ الindex الخاص بهذا العنصر داخل ال( )

Tuple

- الtuple يشبه الlist ولاكم توجد بينهم اختلافات واولهم ان العناصر فى الtuple توضع بين ( )
- الtuple مثل الlist من ناحية ان عناصره مرتبة ويمكن الوصول اليها باستخدام الindex
- لايمكن وضع او اذالة او تعديل فى الtuple على عكس الlist
- عناصر الtuple يمكن تكرارها والtuple يمكن ان يحتوى على انواع مختلفة من الdata
- معظم الmethods التى تم استخدامها فى الList يمكن استخدامها فى الtuple
- Count function يمكن استخدامها فى الtuple ايضا
- Index function يمكن استخدامها ايضا

Set

- الset لديها اختلاف كبير عن الList او الtuple ومن ذالك ان عناصرها توضع داخل { }
- عناصر الset ليست مرتبة ولايمكن الوصول اليها من خلال الindex
- لايمكن عمل slicing للset بحكم انها غير مرتبة و لانستطيع استخدام الindex عليها
- يمكن وضع بداخلها number و string و tuple فقط اى العناصر التى لا يمكن التعديل عليها ولذالك لايمكن اضافة list او dictionary فيها
- العناصر فى الset تكون unique
- يمكن استخدام clear function لازالة عناصر الset

- توجد function تسمى union وهى تقوم بنفس عمل extend حيث انها تدمج 2 set معا
- ()add. هى function تقوم باخذ عنصر الموضوع داخل ال () واضافته الى الlist
- ()remove. هو function يقوم باذالة العنصر الموجود داخل ال() من الlist وفى حالة عدم وجود هذا العنصر فى الList سيحدث error
- ()discard. هى function تقوم ايضا باذالة العنصر من الset ولاكن اذا لم يكن العنصر موجودا فى الlist لن تظهر رسالة error
- لعمل اتحاد ل 2 sets نستخدم علامة | كتالى **a | b** وهى مثل الunion
- للاتيان بالعناصر الموجودة فى set معينة وليست فى الاخرى نستخدم علامة - كتالى **a-b**
- للاتيان بالعناصر المشتركة بين 2 sets نستخدم علامة & كتالى **a & b**
- للاتيان بالعناصر الغير موجودة فى كلا 2 sets وانما موجودة فى واحدة فقط نستخدم علامة ^ كتالى **a ^ b**
- عند التعامل مع الsets فاننا نواجه مصطلحان وهما الsuperset و subset وهما يعنيان وجود set كبيرة يمكن تكوين منها مجموعة من الsets الاخرى ولذالك فا الset الكبيرة هى الsuperset و الset الصغيرة المكونة منها تسمى subset
- ()issuperset. هى function تقوم بتحقق من كون الset المستخدم عليها الfunction هى superset من الset الموجودة داخل ال() اى ترى هل عناصر الset التى داخل ال() موجودة جميعها فى الset الكبيرة ام لا وتقوم بارجاع قيمة Boolean
- ()issubset. وهى function تقوم بتحقق من ما اذا كان الset المستخدم عليها الfunction هى subset من الموجودة داخل ال()
- ()isdisjoint. هى function تقوم بتحقق من ما اذا كان الsets 2 لا توجد بينهم عناصر مشتركة

Dictionary

- يتم تخزين فيه الdata على شكل key و value ويتم الفصل بينهم ب**:** وتوضع بيناته داخل { }
- الkey فى الdictionary يجب ان يكون عنصر سابت لايمكن تغييره لذالك فلا يمكن ان يكون List ولاكن يمكن ان يكون number او string او tuple
- الvalue يمكن ان تكون اي نوع من البينات
- الkey يجب ان يكون unique وفى حالة التكرار سياخذ اخر قيمة تم وضعها
- الdictionary ليس مرتب اى لا يمكن استخدام عليه الindex وانما نقوم بوضع الkey داخل ال[ ] للاتيان بالvalue
- يمكن استخدام الlen function للاتيان بعدد العناصر فى الdictionary
- يمكن استخدام clear function فى الdictionary
- يمكن اضافة قيمة لdictionary من خلال وضع key غير موجود داخل [ ] واضافة له قيمة بعد علامة = كتالى dict[key]=value
- يمكن اضافة عنصر الdictionary ايضا من خلال ()update. ويوضع داخل ال( ) الkey و الvalue محاطين ب{ }
- ()popitem. هو function يقوم بارجاع اخر عنصر تم اضافته الى الdictionary

Boolean

- الvalue boolean هم فقط true و false
- يتم استخدمهم بشكل كبير فى الcontrol flow مثل ال if statement
- ال0 او اى بينات فارغة مثل string لايحتوى على اى character او list بدون عناصر جميعهم يمثلوا false بينما اى قيمة اخرى تمثل true
- الnone هو data ويشبه الnull فى باقى اللغات وهو ايضا يمثل false
- لاختبار اكثر من شرط معا يمكننا استخدام and و or ولعكس شرط يمكننا استخدام not

- Python تدعم الassignment operators مثل =+ و =-
- يوجد فى python الcomparison operators مثل == و =! و < و > والتى تقوم بارجاع قيم true او false
- يمكن تحويل الnumber الى string من خلال ()str ويوضع الnumber بداخل ال( )
- ()tuple هو function يقوم بتحويل الdata الموضوعة بداخل ال( ) الى tuple ولاكن ليس كل البينات يمكن تحويلها وانما الdata التى تتكون من عناصر مثل الstring (يتكون من حروف) و set و list و dictionary
- ()list تقوم بتحويل الdata التى بداخل ال( ) الى list ويجب ان تكون هذه الdata مكونة من عناصر
- ()set تقوم بتحويل الdata التى بداخل ال( ) الى set ويجب ان تكون هذه الdata مكونة من عناصر
- ()dict هو function يقوم بتحويل ما بداخل ال( ) الى dictionary ولاكن توجد شروط لذالك
  1. لا يمكن تحويل الstring
  2. يمكن تحويل tuple ولاكن يجب ان يحتوى الtuple على instead tuples كل منهم له قيمتان وهذان القيمتان سيمثلان key و value
  3. يمكن تحويل الlist ولاكنها تتبع نفس شرط الtuple اى يجب ان تكون على هذا الشكل **[[key1,value1],[key2,value2],[key3,value3]]**
  4. لايمكن تحويل الset
- ()Input هو function يقوم باظهار رسالة وهى التى توضع بين ال ( ) ويطلب من الuser ادخال بينات
- الinput function تقوم بارجاع string حتى لو تم كتابة رقم فى الterminal ولذالك اذا اردنا استخدامه كا رقم يجب علينا استخدام الfunction الخاصة بالتحويل

Control flow

- تتكون الstatement if من if و elif و else وتختلف فى الsyntax عن باقى اللغات
- elif هى else if فى باقى اللغات
- الsyntax الخاص ب if فى لغة الpython كتالى

```
if True :
    print("hello")
```

  - **يتكون الsyntax من كلمة if ويتبعه الشرط ويكون مكان الtrue ثم نضع : ونقوم بكتابة الstatement التى ستنفذ فى حالة تحقق الشرط**
  - **يجب ان تكون الstatement متقدمة عن جملة if بtab كما هو موضح ولا تكون فى نفس مستوها**
- الbody الخاص باى function فى الpython يجب ان يكون متقدم عن الheader function بمسافة او tab
- elif لها نفس syntax الخاص ب ال if ولاكن else تكتب بدون شرط
- يمكن استخدام اكثر من شرط مع استخدام ال or و and و not
- يمكننا وضع if بداخل if وبذالك تسمى nested if وهى هدفها الاساسى اضافة التحقق من شرط اخر بالاضافة الى شرط الif الاساسية
- عند البحث عن عنصر داخل عناصر اخرى سواء كان حرف فى كلمة او كلمة فى جملة او عنصر فى list او فى tuple نستخدم in و not in وهى تسمى membership operators وهى عبارة عن اداوت شرطية تقوم بارجاع true او false لتدل على وجود او عدم وجود عنصر فى مجموعة من العناصر
- While loop فى الpython تكتب كتالى

```
while x<3 :
    print("hello")
```

  - **يجب التاكد جيدا من الشرط الموضوع فى الloop لانه سيظل يعمل طالما ان الشرط صحيح**
- يمكن اضافة else الى الwhile وهى تتحقق فى حالة كان الشرط الخاص بالloop بfalse
- For loop يستخدم فى python عند تنفيذ كود على مجموعة من العناصر او عند تنفز كود عدد محدد من المرات ويكتب كتالى

```
for x in a :
    print(x)
for x in range(1,3) :
    print(x)
```

  - **فى الfor الاولة ال X هو variable يمثل العنصر فى ال a**
  - **فى ال for الثانية سيتم طباعت ال X مرتان فقط عند 1 و 2**
- يمكن ايضا اضافة else الى الfor loop ولاكنها لا اعمل الا عند انتهاء الloop
- يمكن عمل loop على الdictionary وهذا الloop يقوم بالمرور على الkeys الموجدة فى الdictionary
- يمكن عمل for loop بداخل for loop اخرى وتسمى nested loop ويمكن استخدامها فى بعض التطبيقات مثل عمل access لlist 2D او عمل access ل list 2 معا
- يمكننا استخدام الcontinue فى الloop لتخطى خطوة ويمكننا استخدام break لالغاء الloop
- يوجد ايضا pass وهى تستخدم فى حالة اردنا ان نضع function لها Body فارغ (لايمكن ان نجعل body فارغ لان ذالك سايسبب error ولذالك نستخدم pass)

Functions

- الsyntax الخاص بانشاء function تكون كتالى

```
def function_name() :
    #code
```

- يمكننا استخدام return لكى نقوم بارجاع قيمة من الfunction عند استخدامها
- يمكننا انشاء قيمة default للparameter عن طريق اضافة له قيمة عند انشاء الfunction داخل ال ( )
- فى حالة اننا نريد وضع عدد من الparameter ولاكن العدد غير معلوم يمكننا استخدام packing argument وهى ان نضع علامة * قبل الparameter وبذالك فان الparameter يتحول الى tuple يمكننا وضع العدد الذى نريد من الargument بداخله
- الpacking argument يجب وضعه فى نهاية () فى حالة كان معه parameters اخرين
- عندما نريد وضع dictionary كا parameter فاننا نضع قبل اسم الparameter علامة **
- يتم وضع الparameter الحاصل له packing بعلامة ** فى النهاية حتى بعد tuple packing argument
- عند وضع tuple كا argument فى الcall يجب وضع قبله علامة * وكذالك عند استخدام الdictionary يجب وضع **
- Local variable هو variable معرف داخل function معينة ويتم اذالته من الذاكرة بمجرد انتهاء الfunction اى لا يمكن استخدامه خارج الزاكرة
- Global variable هو variable معرف فى الخارج اى ليس بداخل اى function ولا يذال من الذاكرة الا عند انتهاء البرنامج ويمكن لاى function استخدامه
- يمكن تحويل الlocal function الى global من خلال وضع الglobal keyword قبل الvariable عند انشائه

```
def fun():
```

```
    global x
    x=5
```

- **هنا رغم ان الx معرف داخل function فهو يعتبر global بسبب global keyword ويمكن استخدامه من خلال اى function**

- عند وجود global variable واخر local بنفس الاسم وتم استدعائهم من داخل الfunction قسيتم استخدام الlocal بينما ان تم استدعائه من الخارج سيتم استخدام الglobal لان الlocal يذال بمجرد انتهاء الfunction
- عند وجود global variable واخر معرف داخل function ولاكن يستخدم global keyword فانه سيتم عمل update لقيمة المتغير
- يمكن انشاء recursion function فى python وهى الfunction التى يمكن ان تنادى نفسها
- عند انشاء function ونريد اضافة documentation لها بحيث يسهل على مستخدمها فهم ما تقوم به الfunction او كيفية استخدامها فاننا نستخدم ''' ''' او """ """ ونضع الdocumentation فى المنتصف

File handling

- يمكننا اختيار file و وضعه فى variable لكى نقوم بتنفيذ عليه مجموعة من الاوامر ويحدث ذالك من خلال open() function
- Open() function تاخذ قيمتان الاولة وهى مسار الملف و الثانية وهى ال mode
- انواع الmode كتالى
  1. "a" وهى append وتستخدم لاضافة محتوى اضافى على الملف وفى حالة استخدمها وكان الملف غير موجود ستقوم بانشائه
  2. "r" وهى read وتستخدم لقرائة محتوى الملف وهى القيمة الdefault فى حالة عدم كتابة mode وستقوم باظهار error فى حالة عدم وجود الملف
  3. "w" وهى write وتستخدم للكتابة على ملف وفى حالة وجود به بينات سيتم حذفها والكتابة عليها وفى حالة عدم وجود الملف سيتم انشائه
  4. "x" وهى create وتستخدم لانشاء ملف وستقوم باعطاء error فى حالة وجود الملف
- يوجد نوعان من المسارات التى يمكن اضاقتها
  - Relative وهو يبدا من مكان وجود ملف الpython وabsolute وهو يبدا من الroot وهو فى الوندوز **C:/** وفى الينكس **/**
- فى الوندوز ليس c:/ هو الroot الوحيد وانما اى برتشن هو root بذاته مثل D:/ او E:/
- يفضل وضع absolute pass لان الrelative يعطى error فى كثير من الاحيان بسبب عدم تحقق ان المكان الحالى الخاص بملف الpython هو working directory
- يمكن تغير الworking directory من خلال امر os.chdir() ويتم وضع مسار الworking directory الجديد داخل ال( )
- يمكننا تغير الworking directory وجعله هو مجلد الملف الحالى من خلال هذا الكود

```
import os
os.chdir(os.path.dirname(__file__))
```

  - **import os هى لاضافة os module وهذا الmodule يستخدم لتعامل مع نظام التشغيل**
  - **__file__ هو متغير build in فى python يشير الى الملف الحالى**
  - **os.path.dirname() هو امر للاتيان بمسار المجلد للملف الموضوع داخل ()**
- عند تغير الworking directory يمكننا استخدام الrelative path بشكل طبيعى دون حدوث error
- يمكننا الاتيان بمسار ملف من خلال امر os.path.abspath() ويوضع المتغير المحفوظ فيه الملف داخل ()
- بعد تنفيذ امر open مع read mode يمكننا استخدام الاتى
  1. name. لمعرفة اسم الملف
  2. read(). لقراءة الملف كامل ويمكننا وضع داخل ال( ) عدد الcharacter المراد قرائتها فقط
  3. readline(). تستخدم لقرائة line كامل
  4. readlines(). يقوم بارجاع جميع الاسطر فى list
- عند تننفيذ امر open مع write mode يمكننا استخدام الاتى
  1. write(). لوضع نص فى الملف واذا كان يحتوى على نص فانه يزال
  2. writelines(). لوضع list تحتوى على اكثر من نص فى ملف معا وتوضع الlist فى ( )
- فى append mode يمكننا استخدام write() function كما فى الwrite ولاكن الفرق اننا سوف نضيف الى الملف ولن نحذف البينات الموجودة فيه
- يمكن عمل for loop على الfile وتعتبر الاسطر هى الاجزاء التى يتكون منها والتى يحصل عليها الiteration
- يمكننا حذف الملف من خلال استخدام remove(). ونضع مسار الملف داخل ()

Build in function

- all() هى function تقوم بتعامل مع مجموعة من العناصر معا وتوضع الlist الموجود بها العناصر داخل ال( )
- تستخدم all() عند التعامل مع خاصية يفترض ان تكون فى جميع العناصر ومن امثلة ذالك

```
numbers=[1,2,3,4]
if all(numbers) :
    print("there is no 0")
```

  - **هنا قام بتحقق من ما اذا كانت جميع الارقام true ام لا (جميع الارقام true ماعدا 0)**
- يوجد مثال اخر وهو

```
numbers=[1,2,3,4]
if all(numbers) > 0 :
    print("all numbers is positive ")
```

- **هنا يقوم بتحقق من ما اذا كانت جميع الارقام موجبة**

- any() هى function مثل all() تتعامل مع مجموعة من العناصر معا ولاكنها تقوم بتحقق من خاصية يفترض ان تتوجد فى عنصر واحد على الاقل
- bin() هى function بسيطة تستخدم لتحويل الرقم الى binary
- id() هو function لارجاع الid الخاص بالvariable فى الذاكرة ويوضع الvariable داخل ال( )
- sum() هى function تقوم بجمع مجموعة من الارقام فى iterator مثل الlist او غيرها وتوضع الlist داخل ( )
- الsum function يمكن ان تاخذ قيمة ثانية وهى الindex التى ستبدا الجمع منه وفى حالة عدم كتابته ستقوم بالجمع من البداية
- round() هى function تقوم بتقريب الارقام ويوضع الرقم داخل ال ( ) ويمكن اضافة قيمة ثانية لها وهى عدد الارقام العشرية المفترض التقريب منها
- range() هى function تقوم بارجاع list من الارقام ويتم اضافة للfunction بداية الارقام والنهاية وبتاكيد النهاية لا تحسب ويمكن ايضا اضافة الsteps الخاصة بتقدم الارقام
- فى حالة قمنا بكتابة رقم واحد فقط فى الfunction سيتم اعتباره النهاية لان البداية لها قيمة default ب 0
- abs() هى function تقوم بارجاع قيمة absolute اى قيمة موجبة
- pow() هى function تقوم بنفس عملامة ** اى انها تقوم بعمل اس للرقم وهي تاخذ قيمتان الاولة وهى الاساس والثانية وهى الاس
- min() تقوم بارجاع اصغر عنصر ويتم اضافة لها مجموعة من العناصر او iterator
- max() هى عكس الmin function
- Map function هى واحدة من اهم الbuild in function والتى تستخدم لتنفيز function معينة على عناصر iterator
- Map function هتاخذ function و iterator كا argument وتقوم بارجاع map object وهو عبارة عن iterator والfunction التى تاخذها تكون اما function معرفة من قبل او تكون lambda function
- Lambda function هى function بدون اسم تقوم بعمل وظيفة بسيطة ويجب ان تتكون من سطر واحد فقط كتالى

```
lambda parameter1 :  parameter1*2
```

- **ما بعد علامة : هو جملة الreturn**

- filter() هو function مثل الmap تماما ولاكن الاختلاف فى الوظيفة حيث ان الfilter يقوم بارجاع العناصر التى تعطى true لشرط معين فقط على عكس الmap التى وظيفتها هى ارجاع جميع العناصر بعد اتمام عليها عملية معينة
- الfunction الموجودة كا argument هى ما تقوم بارجاع true بعد اختبار العنصر

Modules

- الmodule هو ملف يحتوى على مجموعة من الfunction المكتوبه مسبقا
- لكى نقوم باضافة module فاننا نضيق كلمة import ثم اسم الmodule ثم بعد ذالك يمكننا استخدام جميع الfunction الموجودة فيه
- جميع الmodules موجودة فى ملف Lib فى ملف لغة python
- لكى نقوم باستخدام الfunction فاننا نكتب اسم الmodule ثم نضع علامة . ثم نكتب اسم الfunction (كما فى os module)
- لكى نقوم باضافة function او اثنان او عدد معين فاننا نكتب from ثم نكتب اسم الmodule ثم import ثم اسم الfunctions وعند استعمال هذه الطريقة يجب عند استخدام الfunction كتابة اسمها فقط دون الحاجة الى وضع اسم الmodule ثم علامة .
- لكي نقوم بعمل module فانه يجب تسمية ملف الmodule باسم الmodule
- من الاماكن التى يقوم فيها الpython بالبحث عن modules هى مسرات النظام مث ملف lib فى ملفات الpython وايضا الملف الذى نحن موجودين به حاليا ويمكننا اضافة مسار ايضا من خلال sys ال module عن طريق الاكود الاتى **sys.path.append("path")**
- يمكننا استخدام الmodules باسم مستعار لتسهيل الكتابة وتلاشى مشكلة انها قد تتعارض مع الكود على سبيل المثال لانه يمكن ان تتواجد function او variable بنفس اسم الmodule ولعمل هذا الاسم المستعار نقوم باضة as و بعدها الاسم لجملة الimport
- Package هى folder من مجموعة من الmodule
- يمكننا تنزيل external package عن طريق package manager الخاص بالpython وهو pip من خلال امر **pip install pk_name**
- يمكننا عمل list بجميع الpackage التى تم تنزيلها من خلال امر **pip list**

Try , Except

- يستخدمان فى التعامل مع الerror فى حالة ظهوره
- Try تكتب اولا ويوضع بعدها الcode الذى يمكن ان يظهر منه الerror وبعدها تكتب الexcept ثم الكود الذى سيعمل فى حالة حدوث الerror
- يكتبان كتالى :

```
num=input("enter your number : ")
try :
    num=int(num)
except :
    print("invalid number")
```

- **فى هذه الحالة اذا قام الuser بادخال رقم فان الكود سيعمل بشكل طبيعى ويقوم بتحويل الinput الى int ولاكن فى حالة ادخال احرف هنا سيحدث error وبتالى فالكود الموضوع فى الexcept سيعمل وسيظهر بدلا من الerror**
- يمكننا اضافة else اليهم لتشغيل كود معين فى حالة عدم حدوث الerror وهى اختياري

- يمكن ان يكون للكود الواحد اكثر من نوع من الerror واننا نريد اظهار رسالة مختلفة لكل error ولذالك فيمكن اضافة نوع الerror بعد كلمة except لتحديد الكود الذى سيعمل لكل error وبتالى فيمكن اضافة اكثر من except كتالى :

```
num=input("enter your number : ")
try :
    num=int(num)
    print(10/num)
except ZeroDivisionError:
    print("zero is not allowed")
except ValueError :
    print("invalid number")
```

  - **فى حالة ان الuser قام بادخال صفر فسيحدث error وهو القسمة على الصفر وبتالى فاول except هى من ستعمل**
  - **وفى حالة قام الuser بادخال حرف بدلا من رقم ستقوم الexcept الاخيرة بالعمل**

```
enter your number : h
Traceback (most recent call last):
  File "c:\Users\ahmed\Desktop\test.py", line 3, in <module>
    num=int(num)
ValueError: invalid literal for int() with base 10: 'h'
PS C:\Users\ahmed>
```

- اذا اردنا معرفة نوع الerror يمكننا ذالك من خلال الterminal من خلال التسبب بالerror ورؤية ماذا يسمى كتالى
- يمكننا ان نرى من الصورة التالية ان اسم الerror موضوع فى السطر الاخير فى الerror وهو **ValueError**
- يمكننا اضافة except فى النهاية بدون نوع لتشغيل code معين فى حالة ظهور اى نوع اخر من الerror لم نقم بتحديده وتسمى الglobal except

OOP

- الobject يتكون من attributes و methods
- الattributes هم الdata الخاصة بالobject بينما الmethod هم الfunction الخاصة بالobject
- الclass هو هيكل للobject يتم من خلاله انشاء مجموعة من الobjects مختلفة فى الdata الخاص بها
- لانشاء class نقوم بكتابة الsyntax الاتى

```
class class_name :
    #code
```

- يفضل كتابة اول حرف من الclass ب capital letter
- فى كل مرة يتم انشاء object من الclass فانه يتم تشغيل function __init__ والتى تقوم بعمل initialize لdata الخاصة بالobject لذالك عند انشاء اى class يجب انشاء بداخلها function __init__ وهى مثل الcontractor فى باقى اللغات
- Init function يلزم ان يحتوى على parameter واحد على الاقل وهو self parameter وهو اول parameter يوضع
- Self parameter يشير الى instance الخاص به
- باقى الparameter الخاص بالobject تكتب بعد ال self parameter
- بداخل الinit function نقوم باضافة ال attribute الخاصة بال object كتالى

```
    def __init__(self,fname) :
        self.name=#value
```

  - قمنا هنا بانشاء attribute وهو الname ويمكننا ان نضيف له قيمة او ان نجعل قيمته توضع من الparameter الموجودة بجانب self
- Self function يجب اضافتها الى اى method وليس لل__init__ فقط
- عند عمل access لاى attribute او method من داخل الclass فتكون من خلال الself parameter ثم علامة . ثم اسم الattribute
- لانشاء object فاننا نقوم بكتابة اسم الclass ونضع بين ال() الattribute الخاصة به ويمكننا اضافته الى variable لتسهيل التعامل معه
- عندما نريد عمل access على الattribute الخاصة بobject معين فاننا نستخدم علامة . ثم اسم الattribute وهكذا ايضا عندما نريد عمل access لل method
- يمكننا وضع attribute للclass خارج ال init ولاكن عند عمل access عليها من داخل الclass يلزم اضافة اسم الclass ثم علامة . ثم الattribute
- __init__ هى magic method ويوجد غيرها كثير ولديهم مهام مختلفة

Magic method	Her job	example
__class__	تستخدك لمعرفة اى class تم انشاء من خلاله هذا الobject او الinstance	**Ali.__class__**
__str__	يستخدم لارجاع رسالة مقرؤة يمكن من خلالها التعريف عن الclass	**Profile.__str__**
__len__	تستخدم لارجاع length الobject ويتم استخدمها تلقائيا عند استعمال len()	**Profile.__len__**

Inheritance

- الinheritance هو نظام من خلاله يتم انشاء class من class اخر والclass الجديد يتمتع بجميع attribute و الmethod الخاص بالclass القديم
- الclass الجديد يحتوى على attributes و methods خاصة به غير موجودة فى الclass القديم
- لعمل inheritance فاننا عند انشاء الclass الجديد فاننا بعد اسمه نقوم باضافة () وبداخلها اسم الclass الذى سنورث منه الattribute و الmethod
- فى حالة كتابة function بنفس الاسم فى الclass الجديد فانها تقوم بعمل overwrite على الموجودة فى الclass القديم
- يمكننا عمل multiple inheritance اى اننا نقوم لعمل ورثة من اكثر من class وذالك من خلال وضع اكثر من قيمة داخل ال() ونفصل بينهم ب ,

- بوجد privacy للattribute و للmethod الخاصة بالclass وهى ثلاث انواع
  1. Public تعنى انه يمكن عمل access على الattribute او الmethod من داخل وخارج الclass وهى الحالة default
  2. Private تعنى ان الmethod و الattribute لايمكن عمل لها access الا من داخل الclass
  3. Protected وتعنى ان الmethod و الattribute يمكن عمل access لها من داخل الclass و من الclasses الوارثة
- لجعل attribute او method يكون private نضع _ double فى بداية اسمه بينما لعمله protected نضع _ واحدة فقط
- لتعديل القيم الprivate من خارج الclass فاننا سنحتاك الى وجود method داخل الclass تقوم بعمل get و set لهذه القيمة

SQL Lite

- الdatabase مهمة جدا فى انشاء الapplication حيث اساخدم لتخزين الdata فيها
- يتم تنظيم وتخزين البينات فى tables وكل table يحتوى على الاعندة الخاصة به
- يوجد انواع عديدة من الdatabase sql ولاكن النوع الذى سنتعامل معه سيكون SQL Lite
- يمكن تخزين انواع عديدة من البينات داخل الdatabase مثل date و integer و text
- لتعامل مع الsql lite يجب عمل import للmodule الخاص بها باستخدام امر **import sqlite3**
- لانشاء ملف database او للاتصال بواحد موجود نقوم باستخدام امر **sqlite3.connect()** ونضع داخل ال() مسار الملف ويفضل ان نضع هذا الconnection فى variable لتسهيل التعامل به
- ()execute. هو امر يستخدم لتنفيذ query معينة ونضع الquery كا string داخل ال()
- يوجد برنامج يسمى DB browser for SQLite يمكن استخدامه لعمل تصفح للdatabase بشكل مرئى
- ()close. تستخدم لايقاف الconnection بالdatabase
- الطريقة السليمة لتنفيذ اوامر ليست من خلال الconnection وانما من خلال الcursor
- لانشاء cursor من خلال الconnection نستخدم امر **()cursor.** على الconnection ويفضل بعد ذالك وضعه فى variable لتسهيل استعماله
- يمكننا استخدام امر ()execute. مع الcursor لتنفيذ query معينة
- عند استعمال الcursor يجب قبل قطع الconnection عمل save لتغيرات من خلال امر **()commit.**
- لاسترداد بينات نقوم باستعمال امر execute ونضع داخل ال() الquery ثم نستعمل امر **()fetchone.** على الcursor ليجلب لنا row من النتائج
- الrow يرجع فى شكل tuple
- يوجد امر **()fetchall.** ويستحدم لارجاع الداتا كلها مرة واحدة ويضعها فى array
- يوجد ايضا **()fatchmany.** ويقوم بارجاع عدد معين من الrows وهذا العدد يوضع داخل ال()

Request

- لتنزيل مكتبة request نستخدم امر **pip install requests** فى الterminal
- لعمل get request نستخدم امر ()get. ويكتب الurl داخل ال() كا string ويمكننا تخزين الrequest فى variable لاستخدامه بعد ذالك كتالى :

```
import requests

request1=requests.get("https://facebook.com")
```

- يمكننا استخدام ()dir لمعرفة الfunction الممكن استخدامها على الrequest
- يمكننا معرفة شرح لكل method فى الrequest من خلال استخدام امر ()help
- للحصول على الhtml code او ال response body الخاص بالrequest فاننا يمكننا استخدام **.text** مع جملة print كتالى :

```
print(request1.text)
```

- فى حالة طباعة request1 فقط سيقوم باظهار فى الterminal انه response object وال stats code الخاص بالresponse
- text. مفيدة فى الاتيان بالcontent فى شكل Unicode ولاكن عند الاتيان بصورة او اى file فاننا نريد الاتيان به كا bytes ولذالك نستخدم **.content**
- لتخزين الcontent فى file نستخدم الكود الاتى

```
request1=requests.get("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQgPbd2MBbw3o5_yzYC_pPjoVNKUx7WCrMN3g&s")
with open("img_name.png" , "wb") as f :
    f.write(request1.content)
```

  - للحصول على صورة فاننا نستخدم get request
  - قمنا باستخدام ()open لانشاء الملف وقمنا بادخال اسمه كا اول argument وبعد ذالك حددنا الmod انه wb والذى يعنى write byte
  - قمنا بتسمية هذا الopen كا f وقمنا باستعماله لعمل write

- الكود السابق يمكن كتابته بطريقة اخرى كتالى :

```
file=open("img_test.png" , "wb")
file.write(request1.content)
```

- ميزة الطريقة الاول التى نستخدم فيها with انها تقوم بعمل ()close بشكل تلقائي
- يمكننا معرفة الstatus code من خلال **.status_code**
- يكمننا التحقق من ما اذا كان ال status code الذى حصلنا عليه هو 200 ام لا من خلال **.ok**
- يمكننا معرفة الheaders الخاصة بالresponse من خلال **.headers**
- من المواقع التى يمكن عمل عليها test لتجربة المكتبة هو موقع httpbin.org
- لايفضل وضع الparameters فى الURL وانما توجد طريقة اخرى لذالك وهى من خلال اننا نقوم بوضع الparameters فى dict ويتم اضافتهم الى الrequest كتالى :

```
par={"q":"tbn:ANd9GcQgPbd2MBbw3o5_yzYC_pPjoVNKUx7WCrMN3g","s":""}
request1=requests.get("https://encrypted-tbn0.gstatic.com/images",params=par)
```

- يمكننا ان نرى او نتحقق من الurl الخاص بالrequest من خلال **.url**
- فى حالة الpost request فان الparams option يسمى data ويأخذ ايضا dict كتالى :

```
par={"UserName":"admin","Password":"KB273692"}
request1=requests.post("http://192.168.1.1/api/system/user_login",data=par)
```

- اذا كانت الresponse الخاصة بالrequest من النوع JSON فا يفضل عدم استخدام text. فى عرضها وانما نستخدم ()json. حيث انها تقوم بتخزينها فى dict ومن خلال ذالك سيكون من السهل عمل access لاى عنصر فيه والتحكم فيه بشكل كبير
- لارسال Json فكل ما نحتاج هو ان نضع الrequest فى dict واستخدام json parameter بدلا من params او data
- لاضافة cookies فاننا نضعها فى dict ونقوم بوضعها فى الrequest من خلال cookies parameter
- لاضافة proxy نقوم بانشاء dict ووضع فيه قيمتان واحدة للhttp و الاخرى لل https كتالى :

```
proxy={"http":"127.0.0.1:8080","https":"127.0.0.1:8080"}
```

- عند استعمال proxy يجب ان لا ننسي عمل disable للverified من خلال وضع له ب false
-

Httpx & asyncio

- يمكننا تنزيل httpx من خلال امر **pip install httpx**
- تتميز httpx عن requests فى ان httpx تقبل جعلها asynchronies اى ان عملية ارسال الrequest و استلام الresponse تتم بشكل جانبي ولا يجب على الcode انتظار انتهائها لكى يكمل
- لجعل الكود يتم بشكل asynchronies يجب علينا استخدام asyncio module ووضع الكود داخل function وتعريف الfunction على انها asynchronies كتالى :

```
import httpx
import asyncio

async def test():
    pass
```

- فى الhttpx يمكننا انشاء client وهذا الclient يمكنه ارسال الrequests بشكل asynchronies وعملية انشائه ايضا يجب ان تكون asynchronies كتالى :

```
async def test():
    async with httpx.AsyncClient() as client :
        req=client.get("https://google.com")
```

- على عكس الrequests module فان الhttpx لا تقوم بارجاع response object وانما تقوم بارجاع object اخر الى حين انتهاء الrequest و وصول الresponse
- عندما نريد استلام الresponse نستخدم الامر الاتى :

```
async def test():
    async with httpx.AsyncClient() as client :
        req=client.get("https://google.com")
        response=await asyncio.gather(req)
        print(response[0].status_code)
asyncio.run(test())
```

  - للحصول على الresponse يجب ان نستخدم **await** keyword لانتظار الresponse
  - asyncio.gather(req) يقوم بارجاع الresponse فى array ولزالك يمكن ان تكون req اما array او variable
  - فى حالة req كانت array فان عند وضعها داخل gather() يجب وضع قبلها * لعمل extract لها
  - ميزة وضع الrequests فى array هو انه عند استخدام gather() عليها ستقوم بجعلهم يتم تشغلهم بسرعة عالية
  - لتشغيل الasynchronies function فعلينا وضعها كا argument داخل ال asyncio.run()

- Httpx يمكننا استخدامها تماما مثل requests بشكل synchronies كتالى :

```
r=httpx.get("https://www.google.com")
print(r)
```

- فى حالة استخدام asynchronies يجب التاكد من عدم وجود **rate limit** وفى حالة وجوده يمكننا التحكم فى عدد الrequest التى ترسل فى نفس الوقت من خلال التحكم فى عدد الrequests التى تتواجد فى الarray
- P

Hashlib

- هى المكتبة الرئيسية فى python لعمل hashing

```python
import hashlib

# مثال: إنشاء SHA-256 hash
data = "Hello, World!".encode('utf-8')
hash_object = hashlib.sha256(data)
hex_digest = hash_object.hexdigest()
print(hex_digest)
```

- توجد العديد من الfunctions لعمل الhash مثل sha256 وغيرها ويتم وضع القيمة المراد عمل لها hash فى ( )
- يمكن عمل hash object فارغ من خلال نفس الامر مع عدم وضع data داخل ال( )
- لطباعة الhash يلزم تحويله الى شكل يمكن طباعته ولذالك يمكننا استخدام امر hexdigest()

```python
hash_object = hashlib.sha256()
hash_object.update(b'Hello')
hash_object.update(b' World!')  # يعادل hashlib.sha256(b'Hello World!')
```

- Update() هى function تقوم باضافة كلمات او data اخرى الى الhash

```python
hash_value = hash_object.digest()  # b'\xce...'
hex_value = hash_object.hexdigest()  # 'ce...'
```

- لكى نرجع الhash فى شكل يمكن طباعته فاننا نستخدم functions كتالى :
  1. digest() وتستخد لارجاع الhash كا bytes
  2. Hexdigest() وتستخدم لارجاع الhash فى صورة نص من الhexadecimal

```python
# قائمة الخوارزميات المتاحة
print(hashlib.algorithms_guaranteed)
print(hashlib.algorithms_available)
```

- algorithms_guaranteed يستخدم لمعرفة الhashing الموجود فى كل الانظمة
- algorithms_available يستخدم لمعرفة الhashing المتاحة على النظام الحالى
- الfunctions الشائعة للhash :

```python
md5 = hashlib.md5()
sha1 = hashlib.sha1()
sha256 = hashlib.sha256()
sha512 = hashlib.sha512()
blake2b = hashlib.blake2b()
```

Tkinter

```
import tkinter

window=tkinter.Tk()

window.mainloop()
```

- Tkinter هى مكتبة تستخدام لعمل GUI فى python وهى مثبتة تلاقائيا فى الpython
- يتم انشاء نافذة من خلال عمل object من الTK class
- لكى نجعل النافذة تظل ظاهرة ولاتغلق فاننا نقوم لابضافة امر ()mainloop ويجب الانتباه الى ان الكود يجب ان يكون قبلها
- لتحكم فى طول وعرض النافذة نقوم باستعمال function وهى ()geometry على الwindow ونقوم باضافة لها العرض والطول

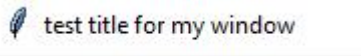
```
window.geometry("800x600")
```

كتالى :

```
window.resizable(False,False)
```

- يمكننا التحكم فى امكانية السماح للuser بتعديل طول وعرض النافذة من خلال امر resizable حيث اننا نضيف له قيمتان Boolean وهما لسماح بالتحكم فى العرض والطول
- يمكننا التحكم بعنوان الwindow من خلال ()title كتالى :

```
window.title("test title for my window")
```




```
window.config(background="gray")
```

- لتحكم فى خلفية النافذة فاننا نستخدم الامر الاتى :
- لتحكم فى الايقونة الخاصة بالwindow فاننا نستخدم ()iconbitmap ونضع لها مسار الايكونة
- فى حالة كان من المسموح تغيير حجم الwindow فيمكننا تحديد المدى المسموح تكبير وتصغير فيه الwindow كتالى

```
window.minsize(200,200)
window.maxsize(800,600)
```

- يمكننا انشاء اكثر من نافذة بطبع من خلال انشاء اكثر من object من خلال TK class
- لتقسيم البرنامج فاننا نستخدم الframe كما فى الكود الاتى :

```
frame=tkinter.Frame(width=800,height=100)
```

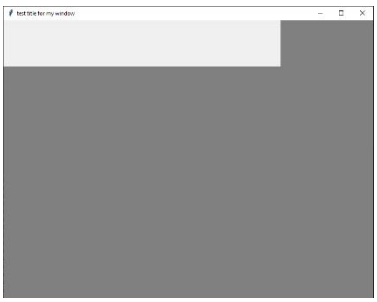
- عند انشاء اى عنصر مثل الframe فانه لا يتم اضافته الى الwindow بشكل تلقائى وانما يتم اضافته باستخدام احدى function التالية والاختلاف بينهم كتالى :

```
frame.pack()
frame.place(x=0,y=0)
```

1. Pack تقوم
   - بالتحكم فى مكان وضعه من خلال كونه قبل عنصر معين او بعده
   - تحكم بسيط فى كونه على اليمين او اليسار او فى الاسفل او الاعلى
   - يمكنها ايضا التحكم فى الpadding (و هى المسافة الداخليه)
2. Place تقوم ب
   - التحكم فى مكان العنصر باستخدام الاحدثيات الx و الy مما يعطى لنا حرية اكبر
   - تقوم بالتحكم فى طول وعرض العنصر الذى لدينا

- فى حالة وجود اكثر من window فوضع الframe او اى عنصر اخر كما فى المثال السابق لن يكون كافيا لان هنا الموضوع يتطلب تحديد الwindow التى سيوضع فيها العنصر (بسبب وجود اكثر من واحدة)

```
frame=tkinter.Frame(window,width=800,height=100)
frame.place(x=0,y=0,width=600)
```



- لانشاء button فاننا نستخدم الكود الاتى حيث نحدد :

```
button=tkinter.Button(frame2,text="click")
```

1. مكان وجوده
2. النص الموجود بداخله (text)
3. لون النص (fg)
4. لون خلفية الذر (bg)
5. التحكم فى شكل الmouse عند الوقوف عليه cursor
6. الfunction التى بفترض ان تعمل عند الضغط على الزر (command)

- لارفاق نص او label فاننا نستخدم ()label الموجودة داخل الTkinter

```
lable=tk.Label(root,text="this test lable", font=("Arial", 14), fg="blue")
```

- القيمة الاولة فى font parameter هى نوع الfont والثانية هى حجم الخط

- لاضافة حقل ادخال نستخدم الentry

```
entry = tk.Entry(root, width=30)
entry.pack()

def button_click():
    print(f"you etered {entry.get()}")

button = tk.Button(root, text="click on me", command=button_click, bg="yellow")
button.pack()
```

- قمنا هنا باضافة حقل ادخال واسمياه entry
- الcommand parameter نقوم باضافة فيه اسم الfunction التى يفترض ان تعمل عندما نضغط على الذر
- لكى نحصل على القيمة الموجودة داخل الentry فاننا نقوم باتسعمال get() function
- يوجد parameter يسمى justify هو يستخدم فى الentry لتحكم مكان كتابة النص (“left”,”right”,”center”)
- هناك المذيد من الtools الموجودة فى module اخر داخل الTkinter وهو ttk اى يجب وضع امر
- لانشاء combo box او ما يسمى select box فاننا نستخدم الامر الاتى

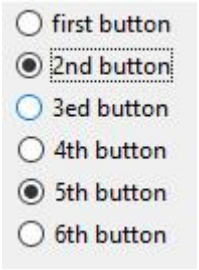
```
selectbox=ttk.Combobox(root,
                       values=["male","femal"],
                       state="readonly")
```

- الvalues parameter يستخدم لوضع الoption لمفترض للuser الاختيار منها
- الstate الموضوعة فى الشكل المقابل تمنع الكتابة داخل الselect box كما فى entry
- لاضافة radio button نقوم بالاتى :

```
radiob5=ttk.Radiobutton(root,text="5th button",value=5,variable=var)
```

- الvalue هى القيمة التى يحملها هذا الاختيار
- الvariable هى قيمة المتغير الذى سيحمل الvalue الخاصة بالذر المختار وكل مجموعة من الradio box لهم نفس الvariable يكونو متصلين معا كتالى :

```
radiob=ttk.Radiobutton(root,text="first button",value=1)
radiob2=ttk.Radiobutton(root,text="2nd button",value=2)
radiob3=ttk.Radiobutton(root,text="3ed button",value=3)
radiob4=ttk.Radiobutton(root,text="4th button",value=4,variable=var)
radiob5=ttk.Radiobutton(root,text="5th button",value=5,variable=var)
radiob6=ttk.Radiobutton(root,text="6th button",value=6,variable=var)
```



- من المعروف ان الradio button يتم استعماله لاختيار خيار واحد فقط ولاكن فى المثال السابق قمنا بعمل اكثر من اختيار لان كما هو واضح من الكود الثلاث radio button الاولة لهم variable مختلف عن الثلاث الاخيرين مما يعنى او اول ثلاث مجموعة والثلاث الاخر مجموعة ثانية
- لانشاء check box الذى يسمح لنا باختيار اكثر من اختيار فاننا نستخدم :

```
check = tk.Checkbutton(root, text="Agree?", variable=check_var)
```

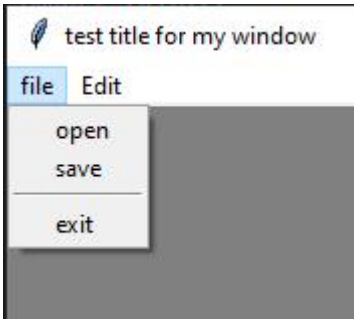
- مكتبة الTkinter تتيح لنا مجوعة من انواع المتغيرات والتى يمكن ان نستخدمها لحفظ الاختيرات بداخلها كتالى :

```
check_var = tk.IntVar()
```

- وللاتيان بالقيمة الموجودة بداخله نستخدم :

```
check_var.get()
```

- لكى نقوم بعمل القائمة الموجودة فى النافذة المقابلة سنحتاج الى معرفت الاتى :
  - القائمة الخاصة بالwindow تتكون من مجموعة من القوائم مثل file و edit كما فى المثال
  - كل قائمة من القوائم لديها قائمة من الcommands مثل open و save الموجودة فى file



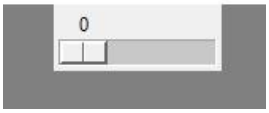
```
menubar=tkinter.Menu(window)
menu_file=tkinter.Menu(menubar,tearoff=0)
menu_file.add_command(label="open")
menu_file.add_command(label="save")
menu_file.add_separator()
menu_file.add_command(label="exit")
menubar.add_cascade(label="file",menu=menu_file)
```

- فى الكود المقابل :
  1. قمنا بانشاء قائمة وهى التى ستحتوى على كل القوائم وقمنا بتسميتها menu bar
  2. قمنا بانشاء menu من القائمة الرئيسية وهى التى ستمثل القوائم التى ستوضع فيها الcommand
  3. قمنا بانشاء مجموعة من الcommands مثل open و save وقمنا باضافة خط فاصل بينهم وبين exit
  4. قمنا بارفاق الmenu file الى الmenu الاساسية
  5. يتم انشاء باقى الmenus كما فى هذا المثال ونقوم فى النهاية باستعمال هذا الامر وهو يقوم باضافة الmenu الاساسية التى قمنا بانشائها الى الwindow
  6. بطبع عن اضافة command باستخام add command فانه يوجد parameter ياخذ اسم الfunction التى سيقوم الcommand بعملها عند اختياره ويسمى هذا الparameter ب command

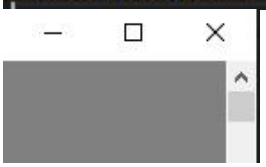
```
window.config(menu=menubar)
```

- لعمل شريط scale فاننا نستخدم :

```
bar=tkinter.Scale(window,from_=0,to=100,orient="horizontal")
bar.pack()
```




```
scroll= tkinter.Scrollbar(window,orient="vertical")
scroll.pack(side="right",fill="y")
```



- لانشاء scroll bar لتقليب الصفحة فاننا نستخدم :
- قمنا بتحديد نوعه من خلال orient parameter وقمنا بجعله يظهر ناحية اليمين من خلال side و جعلناه ياخذ ارتفاع الجانب الايمن كله من خلال fill

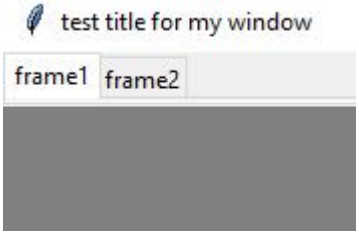
Notebook

- هو جزء يشبه الmenu ولاكنه يستخدم لتبديل بين مجموعة من الصفح او تحديدا مجموعة من الframe على سبيل المثال :

- برنامج الword يحتوى على menu واحدة وهى الخاصة بfile بينما باقى الخيرات هى الnotebook ويمكنك ان تلاحظ ان home هى احداهم ومسؤلة عن اظهار ادوات معينة واذا قمنا بتغير الnotebook الى insert مثلا سنجد ان الادوات تغيرت معنا وهنا نجد ان الجزء الموضوع فيه الادوات هو الframe الذى يتغير من notebook الى اخرى
- فى الكود المقابل قمنا بانشاء notebook وقمنا بعمل لها اضافة الى window ثم قمنا بانشاء اكثر من frame واستخدمنا add function لكى نضيف الالframe الى الnotebook
- لكى نجعل frame معين هو من يتم عرضه فى بداية تشغيل البرنامج فاننا نستخدم ()select. ونضع بداخلها الframe

```
nb=ttk.Notebook(window)
nb.pack()

f1=tkinter.Frame(nb)
nb.add(f1,text="frame1")
f2=tkinter.Frame(nb)
nb.add(f2,text="frame2")
```




```
ttk.Spinbox(window,from_=0,to=100).pack()
```

- Spinbox هو مربع ادخال ولاكن لاياخذ الا الارقام

- لارفاق صورة بذر على سبيل المثال فاننا نقوم بعمل الاتى :

```
image=tkinter.PhotoImage(file="downloads.png").subsample(3,3)
butt=tkinter.Button(window,image=image,text="click",compound="top")
butt.pack()
```

- هنا قمنا باستدعاء الصورة من خلال photoimage وقمنا بوضع object اسمها او المسار فى file parameter وللتحكم فى حجمها قمنا باستعمال sumsample function عليها وكلما ذات الرقم الخاص بها قلة حجم الصورة

- بعد اضافة الصورة قمنا بانشاء الذر وقمنا بارفاق الصورة له من خلال الimage parameter ويلاحظ من المثال اننا قمنا باضافة نص ايضا الى الذر ويمكننا التحكم فى مكان الصورة والنص فى الذر من خلال compound parameter حيث ان top الموجودة فى المثال تدل على ان الصورة ستكون فى الاعلى والنص فى الاسفل
- يمكننا اضافة صورة الى البرنامج ايضا من خلال وضعها كا label وهى ايضا تحدث من خلال استخدام image parameter فى الlabel بدل text

```
texterea=tkinter.Text(window)
texterea.pack()
```

- يمكن عمل text area كتالى
- الtext area هى مثل الentry ولاكن اكبر حيث تتيح عدد اكبر من الاسطر
- يمكن التحكم فى عدد الاسطر من خلال height و يمكن التكم فى عدد الاحرف للسطر الواحد من خلال الwidth

```
t="""this is first line
i am writing this
"""
texterea.insert(tkinter.END,t)
```

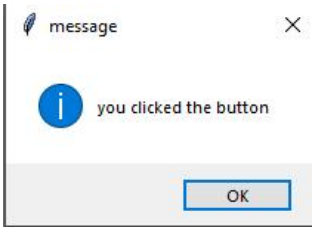
```
le for my window
this is first line
i am writing this
```

- يمكن اضافة قيمة افتراضية تظهر فى ال text area من خلال insert function ويوضع فيها الindex كا اول parameter كما فى المثال المقابل والذى يعنى نهاية النص والparameter الثانى هو قيمة النص الذى سيوضع

Message box

- لكى نقوم باظهار رسالة للuser تظهر له خطاء ما او تحذير او مجرد رسالة بسيطة فاننا نستخدم message box

```
from tkinter import messagebox
def submit():
    messagebox.showinfo("message","you clicked the button")
```



- فى المثال المقابل قمنا باستخدام function وهى showinfo وهى تستخدم لاظهار معلومة بسيطة
- يوجد العديد من الmessage boxes كتالى :
  1. Showinfo
  2. Showwarring
  3. Showerror
  4. Askokcancel
  5. Askretrycancel
  6. Askokcancel
  7. Askyesno
  8. Askyesnocancel
- جميعهم ياخذو 2 parameter الاول وهو title النافذة والثانى وهو الرسالة المنبثقة

Active background & active foreground

- هم خاصيتين فى كثير من العناصر ويقومو على تغير خلفية العنصر او لون نصه فى حالة اصبع العنصر active
- يصبح العنصر active فى حالة تم التعامل معه مثل عند الضغط على الذر فانه يصبح active

Tree view

- ال tree view هو عنصر موجود فى الTTK ويستخدم لانشاء العناصر ذالت الهياكل مثل الtable

```
tree = ttk.Treeview(window, columns=("Name", "Email"), show="headings")
tree.heading("Name", text="Name")
tree.heading("Email", text="Email")
tree.pack(fill="both", expand=True)
```

- فى الكود المقابل قمنا بانشاء tree وقمنا باضافة لها ال Columns وهم الاعمدة الخاصة بالجدول ثم قمنا باضافة show وهى التى تحدد شكل الهيكل والقيمة الخاصة بها والتى هى heading هى ما تحدت انه سيكون table
- يمكن انشاء scroll bar واضافته الى الtree من خلال xscrollcomand و yscrollcomand
- من خلال heading function نقوم بتحديد الخصائص الخاصة بالcolumn من text لتحديد اسمها عندما تظهر
- لتحكم فى حجم الcolumn نستخدم :

```
tree.column("Name",width=20)
```

- عملية اضافة عنصر الى الtable تتم من خلال Insert function

```
self.tree.insert("", "end", values=(name, email))
```

My Notes

- فى sys module يوجد function هامة وهى sys.stdin وتستخدم لاستقبال ملف او لاخذ input من خلال عمل cat مثلا ويتم استخدامها على for loop لتنقل بين الاسطر والتنفيذ عليهم
-