بسم الله الرحمن الرحيم

مبادئ البرمجة بلغة البايثون

احمد سالم الصاعدي

أهــــــــــــــداء

الى والدي رحمة الله عليه ووالدتي وزوجتي واولادي واخوتي واخواتي

**المحويات**

اهــــــداء

الفهــرس

[مقـــــدمة](#مقدمة)

[الفصل الأول: نبذة عن بايثون](#الفصل_الأول)

[الفصل الثاني: المبادئ الاساسية](#الفصل_الثاني)

[الفصل الثالث: القوائم](#الفصل_الثالث)

[الفصل الرابع: الحلقات التكرارية](#الفصل_الرابع)

[الفصل الخامس: اتخاذ القرارات](#الفصل_الخامس)

[الفصل السادس: الدوال في بايثون](#الفصل_السادس)

[الفصل السابع: التعامل مع الم](#الفصل_السابع)لفات

[الفصل الثامن: التعامل مع المكتبات في بايثون](#الفصل_الثامن)

[الفصل التاسع: البرمجة الشيئية في بايثون](#الفصل_التاسع)

[الفصل العاشر: برمجة واجهة المستخدم](#الفصل_العاشر)

[الفصل الحادي عشر](#الفصل_الحادي_عشر):

[الفصل الثاني عشر](#الفصل_الثاني_عشر):

[المراجع](#المراجع)

مقدمة

الحمد لله الذي تتم بفضله الصالحات واصلي وأسلم على خير البشر نبينا محمد صلى الله عليه وسلم وعلى آله وصحبه الطيبين الطاهرين. اما بعد:

فلقد لاحظت خلال فترة تعلمي للغة بايثون افتقار المكتبة العربية الى كتاب مكتمل يشرح مبادي هذه اللغة بشكل منظم وبسيط. فعقدت العزم على تأليف هذه الكتاب وذلك لما رايته من الأهمية بمكان ان يتعلم القارئ العربي هذه اللغة والتي أصبحت اللغة البرمجية المحبوبة لدي الكثير من العلماء والباحثين والمهندسين. فمعظم جامعات العالم اليوم أصبحت تدرسها لطلابها فهي لغة برمجة مشهورة , قوية ويمكن استخدامها في مجالات عدة. لذلك اردت ان يكون هذا الكتاب لبنة أولى للمساهمة في تعليم هذه اللغة. الذي امل ان يتم تدريسها في مدارسنا الحكومية في مراحل مبكرة كالمتوسطة والثانوية وذلك لان الأجيال الحالية على اطلاع واسع بالتقنية وخصوصا الكمبيوترية منها. فهي اصحبت قدرة على تعلمها والاستفادة منها دون مشقة. وبما ان هذا العمل بشري المصدر فانه لا يصل الى درجة الكمال لذلك ارجو ممن سنحت له الفرصة لقراءة هذا الكتاب ان يساهم في تحسين هذا الكتاب بارسال ملاحظاته الى ايميل المؤلف mrahmad.alsaadi@gmail.com. الذي يعدكم على انه سوف يعمل على اخذها في الاعتبار متى ما سنحت الفرصة لاصدار طبعة جديدة لهذا الكتاب.

الفصل الأول:نبذة عن لغة بايثون (python)

"اعتقد انه من الاهمية بمكان ان يتعلم كل واحد منا لغة البرمجة وذلك لأنها تساعدنا على تعلم طريقة التفكير الصحيح" ستيفن جوبز

تعريف لغة بايثون:

بايثون (python)لغة برمجية مفتوحة المصدر سهلة التعلم يمكن الاعتماد عليها في كتابة الكثير من التطبيقات البرمجية القوية. وأكبر دليل على ذلك هو استخدام وكالة الارصاد الامريكية ناسا وشركتا قوقل وياهو وغيرها من الشركات الكبرى لهذه اللغة في بناء برامجهم المختلفة.

نشئت لغة بايثون:

كانت بدايات نشئة هذه اللغة في هولندا على يد شخص يدعي جويدو فان روزم

Guido van Rossum)) في نهاية الثمانيات الميلادية من القرن العشرين. حيث تم الاعلان عنها في عام ١٩٩١م. كما يعتبرفتح مصدرهذه اللغة من اهم الاسباب التي ادت الى زيادة شهرتها من خلال تكوين مجتمع برمجي نشط حولها اسهم في انشاء مكتبات كثيرة سهلت على المطورين الاخرين بناء تطبيقاتهم بسرعة و سهوله فائقة مقارنة باللغات البرمجية الاخري.

مزايا لغة بايثون:

للغة بايثون مزايا عدة جعلت منها اللغة المفضلة الاولى لدى كثير من المبرمجين ومن بين اهم هذه المزايا نذكر:

1. سهولة تراكيبها اللغوية: فاكوادها البرمجية تكتب بطريقة قريبة جدا من اللغة الانجليزية. لذلك نجدها لاتشكل اي عائق امام أي مبرمج ان يفهم الأكواد المكتبوبة من قبل مبرمجين اخرين عندما يستدعي الامر صيانة تلك الاكواد اوتحديثها.
2. المرونة: يمكن تشغيل وتطوير البرامج المكتوبة بلغة بايثون على معظم انظمة التشغيل المعروفة. فالأكواد التي تم تطويرها على نظام ويندوز يمكن تشغيلها على نظام ماك ولينكس والعكس صحيح دون الحاجة الى اعادة بناء الأكواد (compiling).
3. كثرة المكتبات: يعتبر توفر المكتبات من اهم المزايا التي تقدمها اللغة للمبرمجين لتزيد من فعاليتهم في بناء التطبيقات. لذلك عند تنصيب اصدارة بايثون نجد انها تحتوي على مكتبات قياسية كثيرة بعضها يعتبر جزء لا يتجزء من تراكيب اللغة كمكتبة الارقام والقوائم وبعضها الاخر يعمل على تسهيل التعامل مع انظمة التشغيل اما الجزء الاكبر من هذه المكتبات فهو اختياري يتم استيراده متى ما دعت الحاجة لذلك. كما ان هناك مكتبات اخري تحتاج الى تنصيب قبل ان يتمكن المبرمج من استيرادها واستخدامها في برنامجه. وهذه المكتبات مجانية ويمكن تحميلها وتنصيبها اما من الموقع الخاص بالمطورين لهذه المكتبة او من موقع <http://pypi.python.org> والذي يحتوي حتى وقت كتابة هذا الكتاب على 69478 مكتبة مجانية جاهزة يمكن استخدامها في بناء التطبيقات المختلقة.
4. التكامل مع لغات برمجية اخرى: يمكن استخدام بايثون كلغة مساندة تمكن المستخدم لبرنامج مكتوب بلغة سي (C) او سي بلس بلس (C++) مثلا من زيادة او تعديل خصائص ذلك البرنامج ليتناسب مع احتياج المستخدم. ومن أقرب الامثلة على ذلك هو استخدام لغة بايثون في برنامج فري كاد (FreeCAD) كلغة برمجة نصية لتحكم بكافة خصائص البرنامج ووظائفه.

**أصدارت بايثون:**

**هناك اصدارتان لبايثون. الإصدارة الأولى تعرف ببايثون 2 وهي الاقدم والاصدارة الاخرى تدعى بايثون 3 وهي الاحدث.**

**طريقة التنصيب:**

تنصيب بايثون على نظام ويندوز " ":

اذا كنت تستخدم اي من انظمة ويندوز سوا القديم منها او الجديد فانك تحتاج الى تنصيب مفسر لغة بايثون على هذا النظام. واليك الخطوات التالية التي تساعدك على عمل ذلك:

1. حمل مفسر بايثون من الموقع التالي:

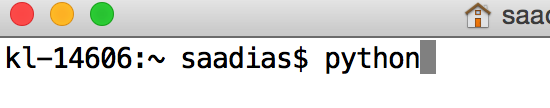
[www.python.gov](http://www.python.gov)

1. نصب مفسر بايثون باتباع التعليمات الموجودة في برنامج التنصيب
2. جهز بيئة ويندوز للتعرف على مفسر بايثون باتباع الخطوات التالية:

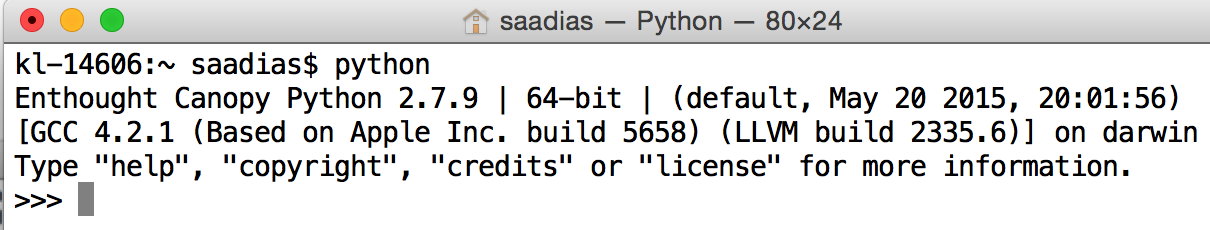
**تنصيب بايثون على ابل ماك:**

يأتي نظام تشغيل ماك وقد نصب عليه مفسر بايثون ذو الاصادرة رقم ٢. ويمكن التأكد من ذلك باتباع الخطوات التالية:

1. قم بتشغيل محرر الاوامر في الماك "تيرمينال"
2. اكتب في محرر الاوامر الامر التالي بعد رمز الدولار "$" :



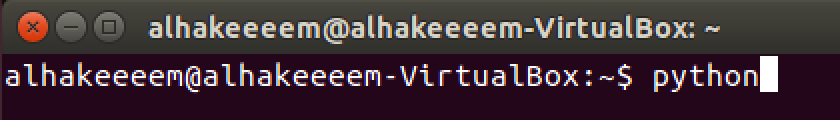
بعد الضغط على زر الادخال سوف تحصل على الاتي:

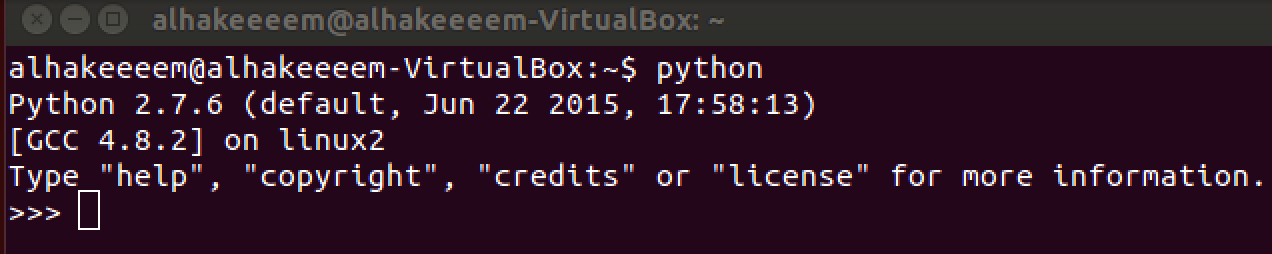


وهويعني ان بايثون منصب على هذا الجهاز وهو من الاصدارة رقم ٢.٧.٩ كما ان المفسر يبدا مباشرة بالوضع التفاعلي والذي يبدأ بمؤشر خاص " ".

**تنصيب بايثون على نظام لينكس (اوبينتو):**

كما هو الحال مع نظام ماك فان نظام اوبينتو لينكس يحتوي على مفسر بايثون وهو كذلك من الاصدارة رقم ٢. لتأكد من وجود بايثون على ابيونتو يمكن اتباع نفس الخطوات التي عملنها مع نظام الماك.





هنا ياتي بايثون باصدارة رقم ٢.٧.٦ وكذلك يبدأ بالوضع التفاعلي.

**طريقة كتابة الكود البرمجي لبايثون:**

هناك طريقتان يمكن بهما كتابة الاكواد البرمجية لبايثون:

1. الطريقة التفاعلية:

وهي الطريقة التي تمكن المستخدم من رؤية نتاتج الامر البرمجي مباشرة بعد تنفيذه. وهذه الطريقة تجعل من بايثون اشبه بالالة الحاسبة حيث يتم الحصول على النتائج مباشرة بعد معالجتها من قبل المفسر.لتاكد من ادراكك لهذه الطريقة يمكنك تجربة كتابة الاكود الاتية وضغظ زر الادخال بعد كل سطر.

1. rntee "مرحبا بك في عالم بايثون" print
2. enter 2+2
3. x=4 enter
4. print 4\*x enter

اذا لم تزل غير متاكد من استيعابك لهذه الطريقة يمكنك زيارة موقع الكتاب ومشاهدة الفديو التوضيحي لهذه الطريقة.

استخدام محرر النصوص:

يأتي مع مفسر بايثون محرر نصوص يدعى "ايدل" ويمكن تشغيلة بكاتبة الامر idle من خلال محرر الاوامر في ماك ولينكس.

او من خلال قائمة البرامج في نظام ويندوز كما في الرسم التوضيحي التالي:

|  |
| --- |
| students=[“Ahmad”,”Ali”,”Omar”]  print students[0]  print students[1]  print students[2] |

كما يمكن مشاهدة الفيديو التوضيحي لطريقة تشغيل محرر النصوص ايدل بزيارة موقع الكتاب.

الفصل الثاني: اساسيات كتابة الكود البرمجي

بعد ان تأكدنا من ان نظام التشغيل الذي نعمل علية يحتوي على احدى اصدارات لغة بايثون يمكننا الان ان نبدأ رحلة التعلم والتى اتمنى ان تكون حافة بالمتعة والفائدة.

الاحرف الكبيرة = الاحرف الصغيرة:

لغتنا العربية الجميلة لاتحوي على مفهوم الحروف الصغيرة والكبيرة بعكس ما هو موجود في اللغة لانجليزية. وبما ان لغة بايثون مكتوبة باللغة الانجليزية فان هذه اللغة تهتم بما اذا كان الكود البرمجي او جزء منه مكتوب بالاحرف الصغيرة او الكبيرة. فالامر..... مثلا يكتب بالاحرف الصغيرة كما في المثال التالي:

ولكن عندما نحاول كتابته بالاحرف الكبيرة فان مفسر لغة بايثون يعطينا خطأ مخبرا انه لايوجد امر معروف لديه باسم .... كما في المثال التالي:

ربما قد تبادر الى اذهان بعض القراء الان ان تذكر وضع حالة الاحرف ما اذا كانت صغيرة ام كبيرة يعد امرا شاقا. لكن علينا ان نتادرك الموقف بسرعة ونبين لهم ان هناك ضوابط وضغت من قبل مطويرين لغة بايثون تجعل من تذكر وضع حالة الاحرف غاية في السهولة. فالضابط الاول هو ان جميع اوامر لغة بايثون دائما يكون مكتوبة بالاحرف الصغيرة. اما الضابط الثاني فان الاحرف الكبيرة فانها فقط تستخدم عند كتابة الثوابت. قد لا يكون هذا الامرواضحا بما فية الكفاية الان لدى القراء ولكن أعدك ان أوضح هذه المسألة بعد ان أتطرق الى المفاهيم الاساسية في لغة بايثون.

طباعة نص على شاشة الكمبيوتر:

كما هو المعتاد في تعلم اي لغة برمجة جديدة فان البدء دايما ما يكون بتعلم كيفية طباعة نص على شاشة الكمبيوتر. وسوف نسير على هذا العرف نحن هنا ايضا. فامر الطباعة على الشاشة في بايثون يكون باسيتخدام الامر ..... متبوعا بما يراد طباعتة. وبما اننا في بداية المشوار ولم نتطرق لمفوهم المتغيرات فسوف نبدأ بطباعة النصوص اولا. فالسطر البرمجي التالي يقوم بطباعة "مرحبا بك في عالم البرمجة" على الشاشة:

لاحظ ان هناك قواعد يجب معرفتها عند طباعة النصوص ومن بين هذه القواعد مايلي:

اولا: يمكن استخدام علامات التنصيص الاحادية (’) او الثنائية (") او الثلاثية (’") لتحيط بالنص المراد طباعته على الشاشة. مع ملاحظة انه يجب ان تسخدم نفس علامة التنصيص في بداية النص واخره وعدم الخلط بينها. كما هو موضح في المثال التالي:

ثانيا: علامة التنصيص الاحادية والثنائية تستخدم لكاتبة نصوص من سطر واحد بينما علامة التنصيص الثلاثية تسمح بطباعة اكثر من سطر. كما في المثال التالي:

ثالثا: يمكن استخدام رمز التجاهل "\" لتمكين علامة التنصيص الاحادية والثنائية من طباعة اكثر من سطر على الشاشة. كما في المثال التالي:

رابعا: يمكن استخدام علامة تنصيص او اكثر داخل علامتي تنصيص ولكن بعد التأكد من ان علامة التنصيص الداخلية مختلفة عن علامة التنصيص المستخدمة في بداية ونهاية النص. كما يمكن استخدام رمز التجاهل المشار اليه سابقا لاداء نفس الوظيفة. كما ماهو موضح في الامثلة التالية:

تدوين الملاحظات:

تسمح لغة بايثون كغيرها من لغات البرمجة للمبرمج بان يكتب ملاحظاته داخل الكود البرمجي من اجل ان تساعدة على تذكر وظيفة الكود البرمجي او من اجل اعطاء شروحات وافيه للمبرمجين الاخرين الذين قد يعملون على صيانة وتطوير الكود البرمجي في المستقبل. ويمكن كتابة ملاحظة من سطر واحد في اي مكان من الكود البرمجي ولكن بعد ان يسبق الملاحظة علامات الهاشتاق "#" كما في المثال التالي:

كما يمكن كتابة ملاحظة متعددة السطور باستخادام علامة التنصيص الثلاثية كما في المثال التالي:

العمليات الرياضية:

كما هو المعتاد مع لغات البرمجة الاخرى فان العمليات الحسابية الاساسية يمكن القيام بها باستخدام الرموز الاتية:

1. "+" للجمع
2. "-" للطرح
3. "\*" للضرب
4. "/" للقسمة
5. "\*\*" للاس

والامثلة التالية توضح استخدام هذه العمليات:

وكما هومتعارف عليه في علم الرياضيات فان عمليتي القسمة والضرب تسبق عملية الطرح والجمع وعملية الاس تسبق الضرب والجمع الا اذا استخدمت الاقواس لتحديد اسبقية العمليات الحسابية. وهذه امثلة اخرى توضح هذا المفهوم:

انواع البيانات في بايثون:

تنقسم انواع البيانات في بايثون لثلاثة اقسام. بيانات رقمية وبيانات منطقية وبيانات نصية. البيانات الرقمية تشمل الاعداد الاعداد الصحيحة "....." والاعداد ذات الفاصلة والاعداد الثنائية التى تتكون قيمة صفر وواحد وبيانات ست عشرية والتي تاخذ القيم من صفر لتسعة بالاضافة الى الخمسة الحروف الاولى من اللغة الانجليزية. البيانات النصية تشمل كل ماهو مكتوب بين علامة تنصيص سواء كانت ارقام او احرف او رموز. اما البيانات المنطقية فهي الاتي تأخد احدى القيمتين صح "...." او خطأ "......". ويمكن التعرف على نوعية البيانات باستخدام الامر..... كما في الامثلة التالية:

المتغيرات:

المتغيرات هي اسماء تستخدم لدلالة على قيم بيانات موجود في ذاكرة الكمبيوترواستخدامها في كتابة الاكواد البرمجية ذو اهمية لايمكن الاستغناء عنها وذلك لاومور التالية:

* المتغيرات تسهل عملية التعديل على الكود البرمجي اذا ماحتاج الامر الى ذلك في المستقبل والمثال التالي يوضح هذا الامر:
* المتغيرات تساعد المبرمج على تذكرالبيانات باسماء يسهل تذكرها بدلا من استخدام قيم البيانات ذاتها.
* المتغيرات وسيلة رياضية تساعد على ايجاد حلول المعادلات الرياضية.

وهناك قواعد يجب اتباعها عند كتابة المتغيرات وتتلخض هذه القواعد فيما يلي:

1. المتغيرات يجب ان تبدأ بحرف او شرطة سفلية.
2. يمكن استخدام الارقام عند كتابة المتغيرات ولكن لايمكن استخدامها في بداية المتغير.
3. المتغيرات يجب ان تكون كلمة واحدة
4. الكلمات المتسخدمة في المتغيرات يجب ان تكون مختلفة عن الكلمات المستخدمة في التركيب اللغوي لبيايثون والجدول التالي يبين الكلمات المحجوزة من قبل لغة بايثون:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| and | def | from | nonlocal | True |
| as | elif | global | None | try |
| assert | else | if | not | while |
| break | except | import | or | with |
| class | False | in | pass | yield |
| continue | finally | is | raise |  |
| del | for | lambda | return |  |

1. لايمكن استخدام اي رمز في كتابة المتغيرات عدا الاحرف والارقام والشرطة السفلية.
2. المتغيرات المكتوبة بالاحرف الكبيرة يتعامل معها مفسر بايثون على انها مختلفة عن المتغيرات المكتوبة بالاحرف الصغيرة.

في معظم لغات البرمجة المعروفه لايمكن استخدام المتغيرات الا بعد تعريفها مسبقا وتحديد نوع البيانات التي تشير اليها. لكن الامر مختلف تماما في لغة بايثون. فالمبرمج لايجتاج لاي من الخطوتان السابقتان. حيث يمكن استخدام المتغيرات مباشرة دون الحاجة لتعريفها مسبقا كما ان مفسر بايثون يتعرف على نوع البيانات التي تشير اليها المتغيرات بمجرد كتابتها. وهذه الخاصية تعطي المبرمج بلغة بايثون سهوله غير مسبوقة في كتابة الاكواد البرمجية.

الفصل الثالث: التعامل مع القوائم في بايثون

القوائم في لغة بايثون هي عبارة عن مجموع من البيانات التي يشار اليها بمتغير واحد. وهي تعتبر وسيلة سهلة لتخزين البينات قبل معالجتها وتحليلها. واقرب مثال يوضح اهمية القوائم هو ان يكون هناك عدة قيم لمتغيرما كدرجات الحرارة خلال السنة مثلا. فالطريقة التي تعلمناها في الفصل السابق تجبرنا على كتابة متغير لكل قيمة كالاتي:

هذه الطريقة طبعا متعبة ومملة. لذلك فان الطريقة الاسهل تتمثل في تخصيص قائمة بمتغير واحد حيث يتم الاشارة الى كل قيمة في هذه القائمة برقم يدل على موقعها. ومن الامثلة على هذه الطريقة مايلي:

["x=["ahmad","ali","omar

في هذا التركيب اللغوي للقائمة هناك قواعد يجب اتباعها لكتابة الكود البرمجي:

اولا: استخدام الاقوام المربعة ( [ ] ) لتحيط بالبيانات واستخدام الفاصلة بين كل بيان واخر.

ثانيا: يمكن تعيين موقع بيان في قائمة باستخدام الامر.... كما في المثال التالي:

ثالثا: لايحتاج الى معرفة عدد القيم قبل ادخالها ويتم تحديد سعة القائمة تلقائيا من قبل مفسرلغة بايثون.

رابعا: يمكن انشاء قائمة فارغة بدون قيم باستخدام قوسين مربعين فارغين كما في المثال التالي:

خامسا: يشار الى مكان البيانات في القائمة بالارقام ٠,١,٢,٣,٤ الى أخ حيث يعطى الرقم صفر لاول قيمة من اليسار وذلك بعد وضعة بين قوسين مربعين بجانب المتغير الذي يدل على اسم القائمة كما في المثال التالي:

سادسا: يمكن معرفة عدد المرات التي تتكرر فيها قيمة بيان باستخدام الامر.... كما في المثال التالي:

سابعا: يمكن تحديد البيانات في القائمة بشكل عكسي من اليمين لليسار باستخدام اشارة السالب والبدء بالرقم واحد ليدل على اخر قيمة في القائمة كما في المثال التالي:

ثامنا: يمكن تجزئة البيانات في قائمة باستخدام نقطتين فوق بعض توضغ بين مكان اول واخر قيمة يراد تجزءتها كما في المثال التالي:

تاسعا: لا تحتاج البيانات داخل القائمة لأن تكون من نفس النوع. فيمكن ادخال قيم رقمية ونصية وكذلك قائمة فرعية داخل قائمة رئيسية كما في المثال التالي:

عاشرا: يمكن معرفة عدد البيانات في قائمة باستخدام الامر.... كما في المثال التالي:

الحادي عشر: يمكن اضافة قيمة الي نهاية قائمة باستخدام الامر... كما في المثال التالي:

او استخدام الامر..... كما في المثال التالي:

الثاني عشر: يمكن اضافة قيمة في مكان محدد الى قائمة عن طريق الامر ... وذلك بتحدين مكان القيمة التي تسبق القيمة الجديدة كما في المثال التالي:

الثالث عشر: يمكن حذف اخر قيمة من قائمة باستخدام الامر .... كما في المثال التالي:

الرابع عشر:يمكن حذف قيمة معينة من قائمة باستخدام الامر ..... او الامر.... كما في المثال التالي:

الخامس عشر: يمكن دمج قائمتين باستخدام علامة الجمع (+) بينهما كما في المثال التالي:

السادس عشر: يمكن تعيين قيمة بيانات من قائمة فرعية موجودة داخل قائمة رئيسية باستخدام الطريقة التالية:

السابع عشر:يمكن تغيير قيمة البيانات في قائمة ببيانات اخرى كما يلي:

الثامن عشر: يمكن عكس ترتيب البيانات في قائمة باستخدام الامر.... كما في المثال التالي:

التاسع عشر: يمكن اضافة بيانات الى قائمة باستخدام لامر..... كما في المثال التالي:

تمارين:

ان الاستفادة القصوى من المعلومات التى طرحناها في هذا الفصل عن القائمة لن تؤتي ثمارها الا بعد التمرن على استخدامها. وها نحن هنا نطرح بعض التمارين التي نأمل ان تساعد القارئ على تثبيت ماقد تعلمه في هذا الفصل. الحلول المقترحة لهذه التمارين موجودة في موقع الكاتب مع مولاحظة انه قد تكون هناك اكثر من طريقة للحصول على نفس النتائج.

الفصل الرابع: حلقات التكرار

تخيل أنك تريد طباعة اسماء ٣ طلاب موجودين عندك في قائمة. ان ما تعلمته حتى الآن يخولك للقيام بهذه المهمة بالطريقة التالية:

|  |
| --- |
| **Code:**  >>> students=[“Ahmad”,”Ali”,”Omar”]  print students[0]  print students[1]  print students[2]  Ahmad  Ali  Omar  >>> |

قد تكون هذه الطريقة مقنعة لك حتى الآن وذلك لأن القائمة قصيرة ولا تحتوي على عدد كبير من الطلاب. لكن من المؤكد أنك سوف تغير رأيك عندما يصبح عدد الطلاب ١٠٠ طالب مثلا. ومن المحتمل ايضا أنك سوف تبحث عن طريقة أسهل تمكنك من اداء مهمتك. إذا كان بحثك مجديا فإنه من المؤكد أنك قد وجدت ما يسمى بالحلقات التكرارية. فهذه الحلقات التكرارية تسمح للمبرمج باداء مهمة معينة لمرات عدة بستخدام كود برمجي بسيط ومختصر. وهذه الحلقات التكرارية تعتبر من التراكيب اللغوية المهمة في لغة البرمجة بشكل عام وفي لغة بايثون بشكل خاص ويوجد نوعين منها في بايثون:

اولا: حلقة for:

**الغرض**: تستخدم هذه الحلقة التكرارية عندما يريد المبرمج تكرار عملية ما عددا معينا من المرات.

**التركيب اللغوي**:

**for** number of iterations:

Statements inside iteration loop

4

2

3

1

5

1. يجب أن تبدأ الحلقة التكرارية بكلمة for.
2. عدد مرات التكرار.
3. النقطتان بعد شرط التحقق ضرورية وعدم كتابتها يؤدي لحدوث خطأ عند تنفيذ البرنامج.
4. الأوامر المطلوب تنفيذها داخل الحلقة التكرارية.
5. ترك مسافة قبل كتابة الأوامر داخل الحلقة التكرارية ضروري ولو بمقدار مسافة حرف وذلك لإخبار مفسر بايثون ان هذه الأوامر موجودة داخل الحلقة التكرارية. الفشل في ترك مسافة يؤدي لحدوث خطأ عند تنفيذ البرنامج. من المتعارف عليه بين المبرمجين ترك مسافة أربع أحرف قبل كتابة الأوامر داخل الحلقة التكرارية.

**مثال ١**:

|  |
| --- |
| **code:**  >>> for x in "python":  ... Print x  ...  p  y  t  h  o  n  >>> |

في هذه الحلقة التكرارية يقوم البرنامج بطباعة حروف كلمة "python" حرفا تلو اخر. ففي كل مرة تأخد x قيمة من المتسلسلة الحرفية لكلمة "python" ليتم طباعتها.

يعتبر نسيان وضع النقطتان ":" بعد سلسلة حلقة التكرار من أكثر الأخطاء الشائعة لدى المبرمجين المبتدئين لذلك أحرص على تذكر كتابتها.

**مثال ٢:**

|  |
| --- |
| **code:**  >>> days=[“Saturday”,”Sunday”,”Monday”,  ”Tuesday”,”Wednesday”]  >>> for i in days:  ... print i  ...  Saturday  Sunday  Monday  Tuesday  Wednesday  >>> |

يقوم هذا البرنامج بطباعة البيانات الموجود في قائمة days واحدا تلو الاخر حيث يأخذ i قيمة البيان الاول ويقوم الامر print بطباعته على الشاشة. تتكرر هذه العملية حتى تنتهي جميع البيانات الموجود في قائمة days.

لابد أنك قد لاحظت ان عدد مرات التكرار قد حددناه في **المثال ١** بعدد الاحرف الموجودة في متسلسلة نصية. وفي **المثال ٢** تم تحديد عدد مرات التكرار بعدد البيانات النصية الموجودة في قائمة. وهتان الطريقتان ليستا الوحيدتان لتحديد عدد مرات التكرار فهناك طرق عدة منها على سبيل المثال استخدام range:

الامر range يقوم بعمل قائمة من ارقام صحيحة وذلك عن طريق استخدام التركيب اللغوي التالي:

|  |
| --- |
| >>> range(start, end, step-size) |

في التركيب اللغوي السابق:

* start تمثل عدد يحدد بداية سلسة الاعداد الصحيحة.
* end تمثل عدد يحدد نهاية سلسلة الاعداد الصحيحة ولا يكون مشمولا فيها .
* step-size تمثل عدد يحدد الخطوات بين عدد واخر في المتسلسلة العددية.
* end عدد ضروي لإنشاء المتسلسلة العددية بينما start وstep-size فهما اختياريان.
* عند حذف العدد الذي يمثل الخطوة بين الاعداد "step-size" سوف يفترض مفسر بايثون ان المبرمج يريد تكوين سلسلة عددية تزيد بمقدار ١.

فمثلا:

|  |
| --- |
| >>> range(2,8)  2  3  4  5  6  7  >>> |

وعند حذف العدد الذي يمثل بدية المتسلسلة العددية والعدد الذي يمثل الخطوة بين الاعداد فان مفسر بايثون يفترض ان المبرمج يريد بداية المتسلسلة العددية من العدد صفر وتزيد بمقدار 1 فمثلا:

|  |
| --- |
| >>> range(6)  0  1  2  3  4  5  >>> |

ويمكن استخدام range في حلقة for التكرارية كما يلي:

|  |
| --- |
| **Code:**  >>> for n in range(0,11):  ... print n  0  2  4  6  8  10  >>> |

هذه الحلقة تطبع على الشاشة الارقام الزوجية من 0 الى 10.

**تمارين:**

|  |
| --- |
| 1. اكتب برنامج يطبع على الشاشة الاعداد التالية: (10,20,30,40,50,60,60,70,80,90,100) 2. اكتب برنامج يطبع الأعداد من صفر لعشرة تنازليا. 3. اكتب برنامج يطبع على الشاشة الأعداد الفردية من 1 الى 9. |

ثانيا: حلقة while

**الغرض**: الهدف من استخدام هذه الحلقة هو تكرار عملية ما عددا غير معروف من المرات من قبل المبرمج. حيث يتم تحديده لاحقا داخل الحلقة التكرارية.

**التركيب اللغوي**:

**While** logical expression:

Statements inside iteration loop

4

2

3

1

5

1. يجب أن تبدأ الحلقة التكرارية بكلمة while.
2. Logical expression هو شرط التحقق الذي بمقتضاه يتم تنفيذ الأوامر داخل الحلقة التكرارية من عدم تنفيذها.
3. النقطتان بعد شرط التحقق ضرورية وعدم كتابتها يؤدي لحدوث خطأ عند تنفيذ البرنامج.
4. الأوامر المطلوب تنفيذها داخل الحلقة التكرارية.
5. ترك مسافة قبل كتابة الأوامر داخل الحلقة التكرارية ضروري ولو بمقدار مسافة حرف وذلك لإخبار مفسر بايثون ان هذه الأوامر موجودة داخل الحلقة التكرارية. الفشل في ترك مسافة يؤدي لحدوث خطأ عند تنفيذ البرنامج. من المتعارف عليه بين المبرمجين ترك مسافة أربع أحرف قبل كتابة الأوامر داخل الحلقة التكرارية.

**مثال 1:**

|  |
| --- |
| **Code:**  i=0  >>> while i < 5:  ... print i  ... i=i+1  ...  0  1  2  3  4  >>> |

يقوم البرنامج باعطاء i قيمة 0 ومن ثم يتحقق من كون i أصغر من 5 أو لا بما ان i تبدأ بقيمة 0 فان الشرط يتحقق ويتم تنفيذ الأوامر داخل حلقة while والتي تتلخص في مهمتين: طباعة قيمة i على الشاشة وزيادة قيمتها بـ 1. تتكرر العملية حتى ينتفي شرط التحقق فيقوم المفسر بالخروج من حلقة while.

تمرين استكشافي: جرب ان تحذف الامر الذي يزيد قيمة i. ماذا تلاحظ؟

الجواب: سوف تلاحظ ان البرنامج سوف يقوم بطباعة العدد 0 بعدد لا نهائي من المرات. وذلك لان قيمة i لا تتغير داخل حلقة while وفي كل مرة يتم التحقق من الشرط فتكون النتيجة إيجابية فيتم امر الطباعة داخل الحلقة التكرارية. ولإنهاء هذه العملية يجب التدخل باستخدام الامر "ctr+c" (يجب الضغط على زر ctr وزر c في نفس الوقت) من لوحة المفاتيح لإنهاء البرنامج عن طريق نظام التشغيل.

**مثال 2:**

|  |
| --- |
|  |

استخدام الامر break:

أحيانا يحتاج المبرمج الى ان يكون شرط التحقق لحلقة while التكرارية دائما متحقق وذلك باستخدام True او 1 كشرط تحقق كما في المثال التالي :

|  |
| --- |
| >>> while True:  ... statements inside the loop |

|  |
| --- |
| >>> while 1:  ... statements inside the loop |

ثم يقوم باستخدام الامر break للخروج من الحلقة التكرارية متى ما أراد ذلك. فعندما يصادف مفسر بايثون كلمة break عند تنفيذه لأوامر داخل الحلقة التكرارية يخرج من الحلقة ويقوم بتنفيذ الأوامر التي تقع مباشرة بعد الحلقة التكرارية. والمثال التالي يوضح كيف يمكن استخدام الامر break:

|  |
| --- |
| Code:  >>> while true:  ... password=12345  ... x=input("please enter password") |

استخدام الامر continue:

استخدام الامر pass :

الحلقات التكرارية الفرعية:

الفصل الخامس: اتخاذ القرارات

ان من الاشياء الجميلة في لغة البرمجة ان المبرمج يستطيع ان يعطي الكمبيوتر قدرة على التفكير "ان صح التعبير" والتي من خلالها يستطيع الكمبيوتراتخاذ قراراته. لكن في حقيقة الامر هذه القرارات ماهي الا شروط يفرضها المبرمج على البرنامج ليحقق . فاتخاذ القرارات في بايثون يتم عن طريق استخدام التركيب اللغوي التالي:

فمتى ماتحقق الشرط بعد كلمة if يتم تنفيذ الاوامر الواقعة داخل اداة الشرط.

اذا لم يتحقق الشرط فان مفسر بايثون يتخطى هذا التركيب اللغوي بدون تنفيذ اي امر. يمكن زيادة تعقيد اتخاذ القرار باستخدام كلمة .... فيصبح التركيب اللغوي على النحو التالي:

في هذا التركيب اذا لم يتحقق الشرط فان المفسر يقوم بتنفيذ الاوامر الواقعة تحت كلمة ......

ويمكن استخدام زيادة تعقيد التركيب اللغوي لاتخاذ القرار باستخدام اداة شرطية داخل اداة شرطية. فيصبح التركيب اللغوي على النحو التالي:

امثلة:

1. اكتب برنامج يفرز مجموعة اعداد موجودة في قائمة الى قائمتين. القائمة الاولى تحتوي على اعداد زوجية والاخرى فردية.

الفصل السادس: الدوال في بايثون

الفصل السابع: التعامل مع الملفات

ان جميع البرامج التي قمنا بكتابتها حتى الان كانت تتطلب ادخال البيانات يدويا داخل الكود البرمجي قبل تشغيله. وهي طريقة بدائية الى حد ما وغير مفيدة خاصة عندما نرغب في ادخال بيانات جديدة غير التي ادخلناها قبل تشغيل البرنامج. لذلك استحدث المبرمجون في بايثون تركيب لغوي يمكن من ادخال البيانات بعد تشغيل البرنامج وهو على النحو التالي:

|  |
| --- |
| Code:  >>> data = raw\_input("message") |

فالتركيب اللغوي هذا يطلب من المستخدم بعد تشغيله للبرنامج ان يقوم بادخال قيمة المتغير data وذلك من خلال كتابة ملاحظة او رسالة "message" للمستخدم ترشده الى ما يجب عمله. يجب ان يلاحظ ان كل مايقوم بادخاله المستخدم من بيانات يتم التعامل معه كسلسلة نصية. لنأخذ مثال على هذا التركيب.

لنفرض اننا نريد كتابة برنامج يطلب من المستخدم ادخال قيمة رقمين x وy لكي يقوم بحساب حاصل جمعهما. فمن خلال تعلمنا للتركيب اللغوي السابق يمكن كتابة البرنامج بالطريقة التالية:

|  |
| --- |
| Code:  x = raw\_input("please enter value for x ")  y = raw\_input("please enter value for y ")  print "\n"  print "----------- result ------------"  print "the sum of x + y = ", x+y |

عند تنفيذ البرنامج السابق نجد ان هناك مشكلة. فالبرنامج تعامل مع المدخلات على انها سلسلة نصية وتمت عملية الجمع كما تعلمنا سابقا بتجميع النصوص مع بعضها البعض. يمكن حل المشكلة بإحدى طريقتين. الأولى هي تحويل سلسلة النصوص المدخلة الى ارقام كما في المثال التالي:

|  |
| --- |
| Code:  x = raw\_input("please enter value for x ")  y = raw\_input("please enter value for y ")  print "\n"  print "----------- result ------------"  print "the sum of x + y = ", float(x)+float(y) |

اما الطريقة الثانية في تستخدم الامر input بدلا من الامر raw\_input فالاولى تقوم بالتعرف على نوع البينات المدخلة تحديد نوعها ام الثانية فهي تتعامل مع المدخلات دائما على انها سلسلة نصية. يمكن كتابة المثال السابقة باستخدام الامر input على النحو التالي:

|  |
| --- |
| Code:  x = input("please enter value for x ")  y = input("please enter value for y ")  print "\n"  print "----------- result ------------"  print "the sum of x + y = ", x+y |

بإمكاننا ان نطور البرنامج السابق قليلا بحيث ان نجعل البرنامج يسأل المستخدم عما اذا كان يرغب في اجراء عملية أخرى او يرغب في انهاء البرنامج. لاجراء هذا التطوير نحتاج لاستخدام الحلقة التكرارية while التي تعلمناها سابقا على النحو التالي:

|  |
| --- |
| Code:  user\_choice="y"  While user\_choice=="y":  x = input("please enter value for x ")  y = input("please enter value for y ")  print "\n"  print "----------- result ------------"  print "the sum of x + y = " x+y  print "the multiplication of x\*y = " x\*y  user\_choice=raw\_input("do you want to continue(y/n)? " |

في هذا المثال أسندنا قيمة "y" للمتغير user\_choice من اجل ان نجعل البرنامج يدخل الى الحلقة التكرارية عند بداية تشغيل البرنامج. فيطلب من المستخدم ادخال قيمة لـ x وy ومن ثم يقوم بطباعة النتائج. يقوم البرنامج بعد ذلك بسؤال المستخدم عن رغبة في الاستمرار وذلك بطلبه لإدخال قيمة y للإجابة بنعم او n اذا كانت الإجابة بالرفض. هذا القيمة المدخلة تسند للمتغير user\_choice والذي يتم من خلالها تحديد ما إذا كان البرنامج يحتاج الدخول للحلقة التكرارية ام لا.

تمكن المستخدمين الذين ليس لهم دراية بكيفية التعامل مع الكود البرمجي. كما ان البرامج السابقة لم تكن تمكنا من الاحتفاظ بمخرجات البرنامج. لذلك استحدث المبرمجين طريقة أسرع وأسهل وهي استخدام الملفات للقراءة منها والكتابة اليها. فيمكن قراءة البيانات عن طريق استخدام الامر open على النحو التالي:

|  |
| --- |
| Code:  >>> file = open(filename, mode,buffer) |

فالتركيب اللغوي السابق يتكون من الاتي:

1. File : يمثل كائن برمجي موجود في الذاكرة يمثل قناة بين القرص الصلب والبرنامج لاداء مهام المطلوبة على الملف.
2. filename: يمثل اسم الملف المطلوب التعامل معه. اذا لم يكن اسم الملف موجود في نفس المجلد الذي يتم من خلاله تشغيل البرنامج فلابد من تحديد مسار الملف حتى يتسنى للبرنامج الوصول للملف. اذا كان المراد الكتابة الى ملف جديد غير موجود مسبقا ولم بتم تحديد المسار المراد حفظ الملف فيه فان البرنامج سوف يقوم بكتابة الملف داخل المجلد الذي تم تشغيل البرنامج منه.
3. Mode : يمثل نوع العملية المراد تنفيذها على الملف. والجدول التالي يمثل العمليات المسموح بها:

|  |  |
| --- | --- |
| الرمز | العملية |
| "r" | قراءة من ملف |
| "w" | كتابة الى ملف |
| "br" |  |
|  |  |

1. .وهذه الجزئية السابقة يقوم بفتح ملف اسمه "data.txt" وتعين امر القراءة باستخدام "r" . كما يجب ملاحظة انه يجب تحديد مسار الملف اذا لم يكن الملف موجود في المجلد الذي تم تشغيل البرنامج منه.

وللكتابة الى ملف يستخدم الامر نفسه ولكن باستخدام "w" بدلا من "r" لتحديد امر الكتابة بدلا من القراءة كالتالي:

|  |
| --- |
| Code:  >>> f = open("data.txt","w") |

الفصل الثامن:

الفصل التاسع:

الفصل العاشر:

الفصل الحادي عشر:

الفصل الثاني عشر:

1. أي من العبارات التالية هي عبارة شرطية صحيحة؟
   1. if (a => 22)
   2. if a >= 22
   3. if (a >=22)
   4. if a >= 22:
2. أي من الرموز التالية يستخدم لكتابة ملاحظات على الكود بسطر واحد:
   1. \*
   2. #
   3. //
   4. (comment)
3. ماهي الكلمة المحجوزة في لغة بايثون والتي تستخدم بعد try:
   1. catch(a)
   2. catch
   3. exception
   4. except
4. اذا كان لدينا متغير اسمه x ونريد ان نعرف نوعه نستخدم:
   1. type(x)
   2. getType(x)
   3. x.type
   4. Type(x)
5. يتم فتح ملف للقراءة منه بالطريقة التالية:
   1. f = open("test.txt","read")
   2. f = open("test.txt","r")
   3. f = open("r","test.txt")
   4. f = open("read","test.txt")
6. الامر المستخدم للطباعة على الشاشة هو:
   1. print
   2. echo
   3. console.log
   4. output
7. يمكن كتابة دالة في بايثون بالطريقة التالية:
   1. def someFunction()
   2. function someFunction()
   3. def someFunction():
   4. function someFunction():
8. من مزايا بايثون انها تعمل الكثير من المنصات:
   1. صح
   2. خطأ
9. يمكن ان تتكامل لغة بايثون مع لغات برمجة أخرى مثل الجافا؟
   1. صح
   2. خطأ
10. يمكن كتابة قائمة اسمها x بالطريقة التالية:
    1. ءx = [1,2,3,4]
    2. x = /1,2,3,4/
    3. x = (1,2,3,4)
    4. x = {1,2,3,4}
11. يمكن تحويل متغير من نوع نصي لرقمي صحيح بالشكل التالي:
    1. castToInteger(a)
    2. integer(a)
    3. int(a)
    4. castToInt(a)
12. يجب تحديد نوع المتغير في بايثون عند كتابته:
    1. نعم
    2. لا
13. يمكن كتابة ملاحظات من سطور عدة على الكود باستخدام:
    1. ’’’
    2. (comment)
    3. ###
    4. ///
14. أي من الجمل التالية ليست صحيحة عن بايثون:
    1. لغة بايثون على منصات عدة
    2. لغة بايثون هي مشابهه للغة php
    3. لغة بايثون تستخدم لتطوير تطبيقات الويب
    4. لغة بايثون تستخدم لتوليد صفحات ويب تفاعلية
15. يمكن فتح ملف والكتابة فيه بالطريقة التالية:
    1. f = open("test.txt","write")
    2. f = open("write","test.txt")
    3. f = open("test.txt","w")
    4. f = open("w","test.txt")
16. يمكن كتابة حلقة loop التكرارية بالطريقة التالية:
    1. loop a < 10
    2. while a < 10 :
    3. while (a < 10 )
    4. while loop a < 10 :
17. ماهي الكلمة التي يمكن ان تضيفها لعبـارة if الشرطية:
    1. elif
    2. elseif
    3. جميع الأجوبة خاطئة
    4. else if
18. بايثون لغة مرنة:
    1. صح
    2. خطأ
19. إذا كان لدينا المتسلسلة النصية x="art" يمكن استبدال الحرف r بالحرف n كتالي:
    1. x.match('n','r')
    2. x.replace('n','r')
    3. x.swap('n','r')
    4. x.replace('r','n')
20. أي من المعاملات الحسابية غير صحيح:
    1. -=
    2. +=
    3. \*=
    4. x=
21. اذا اردنا طباعة البيان الثاني في قائمة x نكتب:
    1. print (x[2])
    2. print (x(2))
    3. print (x[1])
    4. echo(x[2])
22. إذا أردنا إعطاء المتغير a القيمة 2 نكتب:
    1. int a = 2
    2. variable a = 2
    3. var a = 2
    4. a = 2
23. يمكن تغيير قيمة Tuples؟
    1. صح
    2. خطأ
24. أي من التالي يكون صيغة صحيحة كتابة قاموس اسمه x:
    1. x=('item1'=>2,'item2'=>20)
    2. x=('item1':2,'item2':20)
    3. x={'item1'=>2,'item2'=>20}
    4. x={'item1':2,'item2':20}
25. أي من حلقة for التكرارية يعتبر صحيحة:
    1. for a in range(3)
    2. for a loop 3:
    3. for (a=0; a<3;++)
    4. for a in range(1,3):
26. ناتج تنفيذ الكود print '%.2f' %123.444 هو:
    1. 44
    2. 123.44
    3. 123.440
    4. 12

المراجع