جامعة الموصل

كلية التربية للعلوم الصرفة

قسم علوم الحاسوب

محاضرات مادة

الذكاء الاصطناعي

المرحلة الثالثة

نظري + عملي

م.د. هناء فتحي محمود

م. الاء سعد احمد

علم الذكاء الاصطناعي Artificial Intelligence: هو أحد علوم الحاسب الآلي الحديثة التي تبحث عن أساليب متطورة للقيام بأعمال واستنتاجات تشابه ولو في حدود ضيقة تلك الأساليب التي تنسب لذكاء الإنســـان ، فهو بذلك علم يبحث أولاً في تعريف الذكاء الإنســـاني وتحديد أبعاده ، ومن ثم محاكاة بعض خواصه ، وهنا يجب توضيح أن هذا العلم لا يهدف إلى مقارنة أو مشابهة العقل البشري الذي خلقه الله جلت قدرته وعظمته بالآلة التي هي من صـــنع المخلوق ، بل يهدف هذا العلم الجديد إلى فهم العمليات الذهنية المعقدة التي يقوم بها العقل البشــري أثناء ممارســته (التفكير) ومن ثم ترجمة هذه العمليات الذهنية إلى ما يوازيها من عمليات محاسبية تزيد من قدرة الحاسب على حل المشاكل المعقدة.

ومع أن الذكاء هو من أهم العمليات أو الأنشطة التي يقوم بها عقل الإنسان فإنه يصعب تعريفه بدقة : أهو القدرة على الاستنتاج ؟ أم هو القدرة على تحصيل العلم وتطبيقه ؟ أم هو القدرة على استيعاب الأشياء وتصورها والتأثير عليها في العالم الحسي ؟ فقد اختلف العلماء على تعريفه مثلا عرفه افلاطون أنّه النشاط الذي يكسب صاحبه العلم والتعلّم. اما ارسطو فقد قال انه مجموعة من الأحاسيس والمشاعر التي تشكّل العقل والمنطق.

الذكاء يمكن تعريفه بكل ما تقدم ويزيد ، فهو في نطاقه الواسع قد يشمل جميع العمليات الذهنية من نبوغ وابتكار وتحكم في الحركة والحواس والعواطف ، أما في نطاق دراسة علم الذكاء الاصطناعي للحاسبات الآلية فيمكن تعريفه في نطاق قدرة الإنسان على تصور الأشياء وتحليل خواصها والخروج باستنتاجات منها ، فهو بذلك يمثل قدرة الإنسان على تطوير نموذج ذهني لمجال من مجالات الحياة وتحديد عناصره واستخلاص العلاقات الموجودة بينها ، ومن ثم استحداث ردود الفعل التي تتناسب مع أحداث ومواقف هذا المجال

و يمكن تعريف الذكاء الاصطناعي للحاسب الآلي بأنه القدرة على تمثيل نماذج حاسوبية (Computer Models) لمجال من مجالات الحياة وتحديد العلاقات الأساسية بين عناصره، ومن ثم استحداث ردود الفعل

التي تتناسب مع أحداث ومواقف هذا المجال. ويتضح أن الفرق بين تعريفي الذكاء الاصطناعي والإنساني المذكورين أعلاه هو أولاً القدرة على استحداث النموذج فالإنسان قادر على اختراع وابتكار هذا النموذج ، في حين أن النموذج الحاسوبي هو تمثيل لنموذج سبق استحداثه في ذهن الإنسان ، وثانياً في أنواع الاستنتاجات التي يمكن استخلاصها من النموذج فالإنسان قادر على استعمال أنواع مختلفة من العمليات الذهنية مثل الابتكار (Innovation) والاختراع (Creativity) والاستنتاج بأنواعه (Reasoning) في حين أن العمليات الحاسوبية تقتصر على استنتاجات محدودة طبقاً لبديهيات وقوانين متعارف عليها يتم برمجتها في البرامج نفسها.

فهو بذلك يمثل قدرة الإنسان على تطوير نموذج ذهني لمجال من مجالات الحياة وتحديد عناصره واستخلاص العلاقات الموجودة بينها ، ومن ثم استحدث ردود الفعل التي تتناسب مع أحداث ومواقف هذا المجال.

الذكاء الاصطناعي والسلوك الإنساني

إن أساس علم الذكاء الاصطناعي والذي يعني بجعل الحاسبة تفكر بذكاء هو دراسة السلوك الإنساني في التفكير عند اتخاذه لقرار حل مشكلة معينة وذلك بتجزئة هذه الأفكار إلى خطوات أساسية ثم يبدأ التفكير لحل مشكلة وذلك باستخدام تلك الخطوات واختيار الأنسب منها وبأقصر الطرق للوصول إلى الحل أيضا.كذلك التعلم في السلوك الإنساني إذ تخزن المعلومات في أي جزء من الدماغ وتضاف إليها أية معلومات جديدة قد يكتسبها الإنسان في حياته وتعتبر مرجعاً مستقبليا ،إذ يستطيع الإنسان استخدام هذه الحقائق والمعلومات بدون تغيير في طريقة التفكير.

لذا انتهج علم الذكاء الاصطناعي وبرامجه نفس الأسلوب الإنساني في صنع واتخاذ القرار وذلك بتجزئة المشكلة قيد البحث إلى خطوات محددة والاستفادة من تلك الخطوات في بناء الأنظمة والبرامج ، أما اكتساب المعرفة فتمثل الحقائق والقوانين في قاعدة المعرفة وتستخدم طرق الاستنتاج والاستدلال للوصول إلى الحل أو إضافة معلومات جديدة إلى قاعدة المعرفة وهي بذلك تشابه السلوك الإنساني في اكتسابه المعرفة.

تطبيقات الذكاء الأصطناعي :

يستخدم الذكاء الأصطناعي في مجالات متنوعة مثل: النظم الخبيرة، والتشخيص الطبي، ومحركات البحث على الإنترنت، ومعالجة اللغات الطبيعية، وألعاب الفيديو، وتداول الأسهم، والقانون، وتمييز وتحليل الصور، ولعب الأطفال، والاكتشافات العلمية، والتحكم الآلي، وتمييز الأصوات.

- **الجانب الصحي**

استخدم الذكاء الاصطناعي في تحسين نتائج المرضى وخفض التكاليف. تقوم الشركات بتطبيق التعلم الآلي لإجراء تشخيصات أفضل وأسرع من البشر. مثال على ذلك chatbots الذي يعتبر من أفضل تقنيات الرعاية الصحية المعروفة، وهو برنامج كمبيوتر يستخدم عبر الإنترنت للإجابة على الأسئلة ومساعدة العملاء ، كذلك يساعد في جدولة مواعيد المتابعة أو مساعدة المرضى من خلال عملية إعداد الفواتير ، والمساعدين الصحيين الظاهريين الذين يقدمون ملاحظات طبية أساسية.

- **الجانب التعليمي**

يستطيع معلمي الذكاء الاصطناعي أتمتة الدرجات ، تقييم الطلاب والتكيف مع احتياجاتهم ، ومساعدتهم على العمل بشكل صحيح. كما يمكن لمعلمي الذكاء الاصطناعي تقديم دعم إضافي للطلاب ، مما يضمن بقاءهم على المسار الصحيح.

- **الجانب المالي**

تستخدم البنوك أنظمة الذكاء الاصطناعي لتنظيم العمليات ، والاستثمار في الأسهم ، وإدارة الممتلكات. في أغسطس ٢٠٠١ ، فازت الروبوتات على البشر في مسابقة محاكاة تداول مالية.

- **مواقع التواصل الاجتماعي**

Facebook يعمل حاليا في استخدام الذكاء الاصطناعي لتحليل الطريقة التي يتواصل بها الأشخاص مع بعضهم البعض حتى يتمكن من إضافة ميزات جديدة إلى خدماته أو حتى إزالة المشاركات المسيئة تلقائيًا التي قد تحدث عندما ينشر أحد المشاهير البارزين.

- **الجانب الترفيهي:**

تلعب الذكاء الاصطناعي دورًا مهمًا في التفكير في عدد كبير من المواقف المحتملة بناءً على المعرفة العميقة في الألعاب الإستراتيجية. على سبيل المثال ، لعبة الشطرنج، الداما، وطاولة الزهر وغيرها.

لغات برمجة الذكاء الأصطناعي

في مجال الذكاء الاصطناعي استخدمت عدة لغات برمجية مثل Lisp، Python، C++، Prolog، Java، ومن اشهرها هما:

- **لغة lisp**

وهي اختصار لـ list of processing وتعني معالجة القوائم والتي تم تصميمها عام ١٩٨٤ في الولايات المتحدة وكان الغرض منها تحقيق الأغراض البرمجية للذكاء الاصطناعي.

- **لغة prolog**

وهي اختصار لـ programming in logic وتعني البرمجة بالمنطق والتي تم تصميمها عام ١٩٧٠ بجامعة مرسيليا بفرنسا بغرض برمجة المسائل المنطقية قبل ظهور علم الذكاء الاصطناعي

تمتاز لغات الذكاء الاصطناعي بخصائص تناسب طبيعة أنظمة الذكاء الاصطناعي والخصائص هي :

أ- قابلية تمثيل المعرفة Knowledge Representation

ويقصد بها استخدام قواعد خاصة لوصف المعرفة (حقائق Facts،علاقات Relations، قواعد Rules، اطر Frames) وهي التي تشكل قاعدة المعرفة Knowledge Base.

ب- معالجة الرموز والأشكال Symbolic Processing

تمتاز لغات الذكاء الاصطناعي بإمكانية معالجة الرموز والأشكال.

ج- مرونة في التحكم Flexibility of Control

اللغات التقليدية مثل Pascal و C تقوم بمعالجة المشكلة من خلال تتبع تسلسلي لتعليمات البرنامج فهي دائما ما تكون عاجزة عن علاج مشاكل الذكاء الاصطناعي لذلك أتت لغات الذكاء الاصطناعي بإمكانية تحكم أكثر مرونة.

وبشكل عام:

تعتبر لغات الذكاء الاصطناعي أكثر كفاءة من اللغات التقليدية ونعني بالكفاءة زمن تنفيذ البرنامج وتقليل حجم التخزين في الذاكرة ولكن نحتاج الى مجهود من قبل المبرمج في تحديد كل الحقائق وربطها يبعضها البعض وتوجيهها لاستخلاص النتائج والأهداف المطلوبة.

أنواع الذكاء الاصطناعي

في الوقت الحالي، أصبح الناس مهووسين بالذكاء الإصطناعي خاصة بعد تطوير الروبوت صوفيا في أكتوبر ٢٠١٧ . سنُناقش هنا أنواع الذكاء الإصطناعي الأربعة الرئيسية وكيف تم تطويرهم بمرور الوقت:

النوع التفاعلي Reactive machines

هذه هي أقدم أشكال أنظمة الذكاء الاصطناعي التي لديها قدرة محدودة للغاية. إنها تحاكي قدرة العقل البشري على الاستجابة لأنواع مختلفة من المحفزات. لا تملك هذه الأجهزة وظيفة تستند إلى الذاكرة. وهذا يعني أن هذه الآلات لا يمكنها استخدام الخبرات المكتسبة مسبقًا لإبلاغ أفعالها الحالية ، أي أن هذه الأجهزة لا تملك القدرة على "التعلم". لا يمكن استخدام هذه الأجهزة إلا للاستجابة تلقائيًا لمجموعة محدودة أو مجموعة من المدخلات. لا يمكن استخدامها للاعتماد على الذاكرة لتحسين عملياتها بناءً على نفس الشيء. ومن الأمثلة الشائعة على جهاز AI التفاعلي ، جهاز Deep Blue التابع لشركة IBM ، وهو الجهاز الذي فاز في لعبة الشطرنج على بطل العالم Grandmaster Garry Kasparov في عام ١٩٩٧.

نوع الذاكرة المحدودة Limited memory

تتكون الذاكرة المحدودة من نماذج للتعلم الآلي تستمد المعرفة من المعلومات أو البيانات المخزنة أو الأحداث التي تم تعلمها مسبقًا. على عكس الأجهزة التفاعلية ، تتعلم الذاكرة المحدودة من الماضي من خلال مراقبة الإجراءات أو البيانات التي يتم توفيرها لها من أجل بناء المعرفة التجريبية.

على الرغم من أن الذاكرة المحدودة تعتمد على بيانات الرصد بالاقتران مع البيانات المبرمجة مسبقًا التي تحتويها الأجهزة بالفعل ، إلا أن هذه العينات من المعلومات تنتقل سريعًا. من الامثلة على الذاكرة محدودة هي المركبات ذاتية الحكم.

نوع نظرية العقل Theory of mind

نظرية العقل هي القدرة على صنع القرار على قدم المساواة مع مدى العقل البشري ، ولكن عن طريق الآلات. في حين أن هناك بعض الآلات التي تظهر حاليًا قدرات إنسانية (مساعدين صوتيين ، على سبيل المثال) ، لا يوجد أي منهم قادر تمامًا على إجراء محادثات تتعلق بالمعايير الإنسانية. أحد مكونات المحادثة البشرية هو القدرة العاطفية ، أو السبر والتصرف مثلما يفعل الشخص في الاتفاقيات القياسية للمحادثة.

قد تتضمن هذه الفئة المستقبلية من قدرة الماكينة فهم أن الأشخاص لديهم أفكار وعواطف تؤثر على المخرجات السلوكية وبالتالي تؤثر على عملية التفكير في آلة "نظرية العقل". التفاعل الاجتماعي هو أحد الجوانب الرئيسية

للتفاعل البشري ، ومن أجل جعل نظرية آلات العقل ملموسة ، فإن أنظمة الذكاء الاصطناعي التي تتحكم في الآلات الافتراضية الآن يجب أن تحدد ، تفهم ، تحافظ على ، وتذكر المخرجات والسلوكيات العاطفية .

من هذا المنطلق ، يجب أن تكون آلات نظرية العقل المذكورة قادرة على استخدام المعلومات المستمدة من الأشخاص وتكييفها في مراكز التعلم الخاصة بهم لمعرفة كيفية التواصل مع المواقف المختلفة ومعالجتها. نظرية العقل هي شكل متقدم للغاية من الذكاء الاصطناعي المقترح والذي سيتطلب من الآلات الاعتراف بالتحولات السريعة في الأنماط العاطفية والسلوكية لدى البشر ، وفهم أيضًا أن السلوك البشري سائل ؛ وبالتالي ، يجب أن تكون نظرية آلات العقل قادرة على التعلم بسرعة في أي لحظة.

بعض عناصر نظرية العقل موجودة حاليًا أو موجودة في الماضي القريب. مثالان بارزان هما روبوت Kismet و Sophia ، اللذان تم إنشاؤهما في عامي ٢٠٠٠ و ٢٠١٦ ، على التوالي.

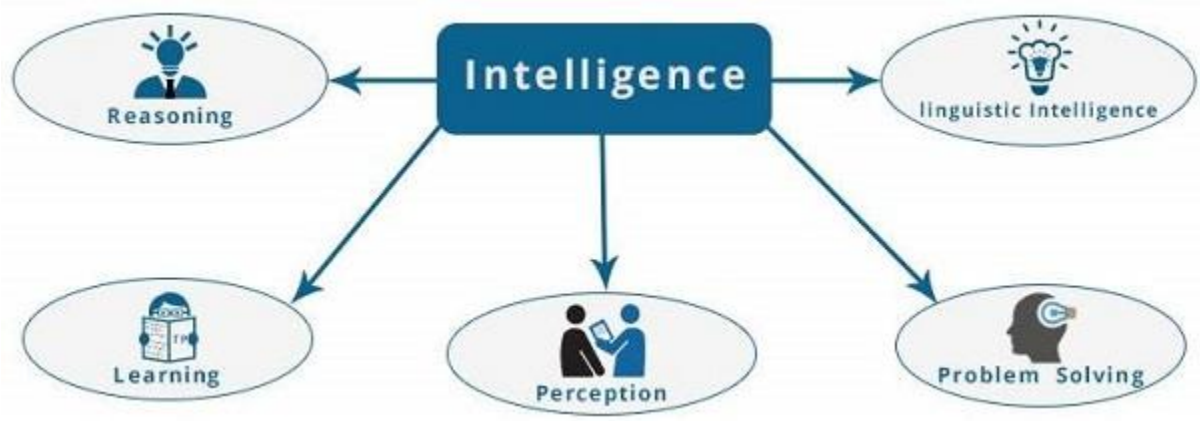
النوع الذاتي الادراك Self-Awareness

الذكاء الاصطناعى المدرك ذاتيا يتضمن الآلات التى لها وعي على مستوى الانسان. هذا الشكل من الذكاء الاصطناعي ليس موجودًا حاليًا ، لكنه يُعتبر أكثر أشكال الذكاء الاصطناعي تقدمًا.

لا تشمل جوانب الذكاء الاصطناعي الواعي القدرة على التعرف على الأعمال الإنسانية وتكرارها فحسب ، بل أيضًا على التفكير لنفسها ولديها رغبات وفهم مشاعرها. الذكاء الاصطناعي الذاتي ، في جوهره ، هو تقدم وتوسيع لنظرية العقل الذكاء الاصطناعى. عندما تركز نظرية العقل فقط على جوانب فهم الممارسات الإنسانية وتكرارها ، فإن الذكاء الاصطناعي الذي يدرك نفسه يأخذها خطوة إلى الأمام عن طريق الإيحاء بأنه يمكن أن يكون له أفكار وردود فعل موجهة ذاتيا.

مكونات الذكاء الاصطناعي

علم الذكاء الاصطناعي يشمل ما يلي:



المنطق Reasoning

إنها مجموعة العمليات التي تمكننا من توفير أساس للحكم واتخاذ القرارات والتنبؤ. يوجد على نطاق واسع نوعان:

الاستدلال الاستقرائي Inductive Reasoning

يقوم باجراء ملاحظات محددة للإدلاء ببيانات عامة واسعة. حتى لو كانت جميع الجمل المنطقية صحيحة ، فإن التفكير الاستقرائي يسمح بأن تكون النتيجة خاطئة.

مثال:

"سها هي مدرسة.

جميع المدرسات مجتهدات.

لذلك ، سها هي مجتهدة ".

المنطق الاستنتاجي Deductive Reasoning

يبدأ ببيان عام ويفحص إمكانيات الوصول إلى استنتاج منطقي محدد.

إذا كان هناك شيء صحيح بالنسبة لفئة من الأشياء بشكل عام ، فإن هذا ينطبق أيضًا على جميع أعضاء تلك الفئة.

مثال:

"جميع النساء فوق سن ٦٠ سنة هم الجدات.

سها ٦٥ سنة.

لذلك ، سها هي جدة.

التعلم Learning

هو نشاط اكتساب المعرفة أو المهارة من خلال دراسة أو ممارسة أو تدريس أو تجربة شيء ما. التعلم يعزز الوعي بموضوعات الدراسة.

يمتلك الإنسان ، وبعض الحيوانات ، والأنظمة الذكية القدرة على التعلم. يتم تصنيف التعلم على النحو التالي:

التعلم السمعي: إنه التعلم من خلال الاستماع والسمع. على سبيل المثال ، يستمع الطلاب إلى المحاضرات الصوتية المسجلة.

التعلم العرضي: للتعلم عن طريق تذكر تسلسل الأحداث التي شهدها المرء أو مر بها.

التعلم الحركي: هو التعلم بحركة دقيقة للعضلات. على سبيل المثال ، التقاط الاشياء ، الكتابة ، إلخ.

التعلم بالملاحظة: التعلم من خلال مشاهدة وتقليد الآخرين. على سبيل المثال ، يحاول الطفل التعلم من خلال محاكاة والديه.

التعلم الإدراكي: هو تعلم التعرف على المحفزات التي شاهدها المرء من قبل.

على سبيل المثال ، تحديد وتصنيف الاشياء والمواقف.

التعلم المكاني: هو التعلم من خلال المنبهات البصرية مثل الصور والألوان والخرائط وما إلى ذلك. على سبيل المثال ، يمكن للشخص إنشاء خريطة طريق في الذهن قبل اتباع الطريق فعليا.

تعلم التحفيز – الاستجابة: إنه تعلم أداء سلوك معين عند وجود حافز معين.

حل المشكلات Problem Solving

هي العملية التي يفكر فيها الشخص ويحاول التوصل إلى حل مرغوب فيه من الموقف الحالي من خلال اتخاذ بعض المسارات، الوصول الى الحل قد يواجه عقبات معروفة أو غير معروفة.

يشمل حل المشكلات أيضًا اتخاذ القرارات ، وهي عملية اختيار أفضل بديل مناسب من بين البدائل المتعددة للوصول إلى الهدف المنشود.

الإدراك Perception

هي عملية الحصول على المعلومات الحسية وتفسيرها واختيارها وتنظيمها. التصور يفترض معنى.

في البشر ، الإدراك الحسي يساعده الأعضاء الحسية. المقصود ان الانسان يستخدم حواسه من اجل ادراك مايدور حوله.

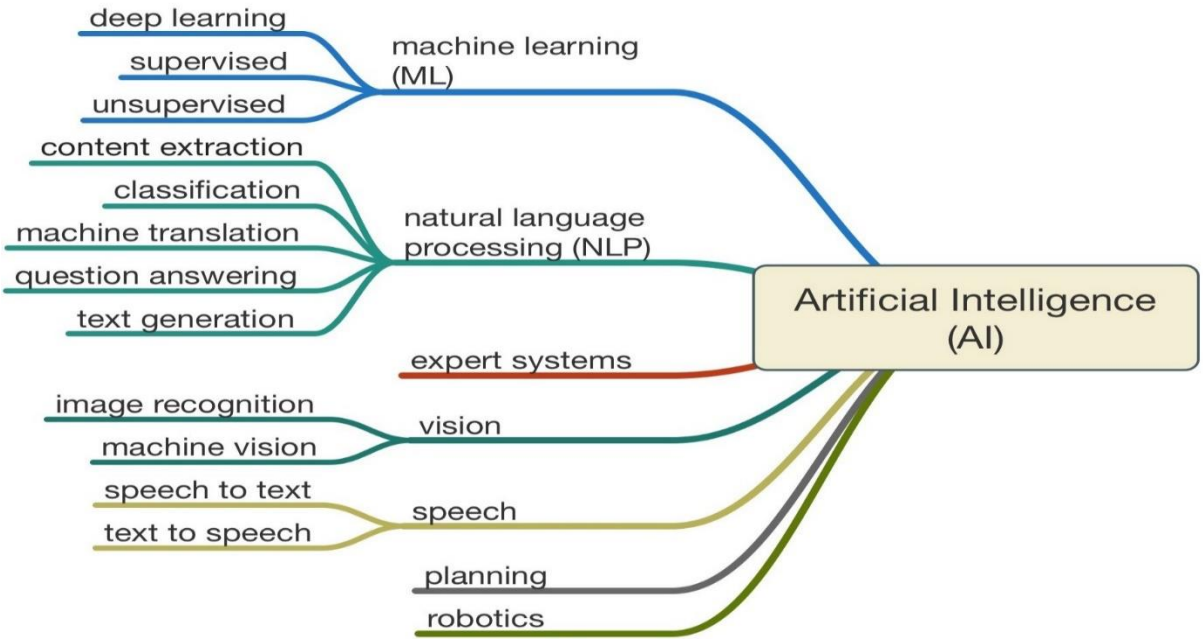
في مجال الذكاء الاصطناعى ، آلية الإدراك تجمع البيانات التي يتم الحصول عليها من خلال المستشعرات بطريقة هادفة. ثم يتم تحليل هذه البيانات من اجل ادراك مالذي يحدث.

الذكاء اللغوي Linguistic Intelligence

إنها قدرة الفرد على استخدام اللغة الشفوية والمكتوبة وفهمها والتحدث بها وكتابتها و تعتبر مهمة في التواصل الشخصي.

١. **<u>مجالات البحث في الذكاء الأصطناعي.</u>**

لقد كان للذكاء الاصطناعي الذي يركز على المعرفة واستخدام الطرق التخمينية في حل المشكلة اثر واسعًا وشاملا في الاختصاصات الأخرى .لذا استخدم في تطبيقات كثيرة منها:



التعلم الآلي Machine Learning: هي طريقة يتم فيها تعريف الهدف ويتم تعلم الخطوات للوصول إلى هذا الهدف من قبل الجهاز نفسه من خلال التدريب (اكتساب الخبرة). على سبيل المثال ، تحديد شيء بسيط مثل تفاحة أو برتقالة. لا يتم تحقيق الهدف من خلال تحديد تفاصيل حوله وترميزه بشكل صريح ، ولكنه تمامًا كما نعلم الطفل من خلال عرض عدة صور مختلفة له وبالتالي السماح للجهاز

بتحديد الخطوات اللازمة لتعريفه مثل تفاحة أو برتقالة.

معالجة اللغة الطبيعية (NLP): تُعرف معالجة اللغة الطبيعية على نطاق واسع على أنها المعالجة التلقائية للغة الطبيعية ، مثل الكلام والنص ، بواسطة البرامج. أحد الأمثلة المعروفة على ذلك هو اكتشاف البريد الإلكتروني العشوائي حيث يمكننا أن نرى كيف تحسنت في نظام البريد لدينا.

الأنظمة الخبيرة (Expert Systems)

لفظ الخبير مشتق من الخبرة ، وهو الشخص المتمرس الذي مر بتجارب عديدة صقلت فهمه لمجال من المجالات وأغنت فكره بمعلومات اختص بها دون غيره ، وميزته عن غيره من المختصين في المجال وبذلك استحق لفظ خبير. وتهدف الأنظمة الخبيرة إلى تطوير برامج حاسوبية تستطيع تحليل الأحداث والمواقف في مجال من المجالات والوصول إلى نفس الاستنتاجات أو النتائج التي يصل لها الخبير. ويتم ذلك عن طريق استحداث نموذج حاسوبي يوازي النموذج الذهني الذي لدى الخبير وخزن المعلومات به ، وقد دلت الأبحاث على أن المعلومات التي يستخدمها الخبير في عمله تنقسم إلى قسمين رئيسيين الأول خاص بالمعلومات الشائعة في هذا المجال مثل الحقائق والقوانين (facts) المتعرف عليها والمقبولة لجميع المختصين والحدس او الاجتهاد (Heuristics) التي يتميز بها الخبير عن غيره والتي قد تكون على شكل علاقة مثلا بين لون البشرة ونسبة الكوليسترول في الدم ، أو الشكل الانسيابي لعينة صخرية ونسبة الترسبات المعدنية فيها.

ومن أوائل الأنظمة الخبيرة التي تطورت حتى الآن نظام مايسن Mycin لتحليل وعلاج أمراض الدم المعدية ، وقد طور هذا النظام في جامعة ستانفورد حيث أحتوت قاعدة معلوماته على نحو (٤٠٠) قانون تربط العوارض المحتملة للمرض بالاستنتاجات الممكنة.

الرؤية Vision: يمكن القول يمكّن الآلات من الرؤية. تلتقط رؤية الماكينة المعلومات المرئية وتحللها باستخدام الكاميرا والتحويل التمثيلي إلى الرقمي ومعالجة الإشارات الرقمية. يمكن مقارنته ببصر الإنسان ولكنه غير مرتبط بالقيود البشرية التي يمكن أن تمكنه من الرؤية من خلال الجدران (الآن سيكون من المثير للاهتمام إذا كان لدينا غرسات يمكن أن تجعلنا نرى من خلال الجدار). يتم تحقيق ذلك عادةً من خلال التعلم الآلي للحصول على أفضل النتائج الممكنة حتى يمكننا القول أن هذين الحقلين مترابطان.

الكلام Speech : تمكين الكمبيوتر على فهم الكلام البشري عن طريق تلقي الأصوات من الخارج وإعادة تجميعها والتعرف عليها ومن ثم الرد عليها .

التخطيط Planning : يدور التخطيط في الذكاء الاصطناعي حول مهام صنع القرار التي تقوم بها الروبوتات أو برامج الكمبيوتر لتحقيق هدف محدد.

الروبوتات Robotics : إنه مجال هندسي يركز على تصميم وتصنيع الروبوتات. غالبًا ما تستخدم الروبوتات لأداء المهام التي يصعب على البشر القيام بها أو القيام بها باستمرار. ومن الأمثلة على ذلك خطوط تجميع السيارات ، والمستشفيات ، ونظافة المكاتب ، وتقديم الأطعمة ، وإعداد الأطعمة في الفنادق ، والقيام بدوريات في مناطق المزارع وحتى كضباط شرطة. تم استخدام التعلم الآلي مؤخرًا لتحقيق بعض النتائج الجيدة في بناء الروبوتات التي تتفاعل اجتماعيًا (صوفيا).

تعريف بعض العلماء للذكاء الاصطناعي

كما اوضحنا سابقا ان الذكاء الاصطناعي : يهدف الى برمجة الحاسبات بحيث يمكنها القدرة على التفكير (Reasoning) والتوصل الى حل المشاكل واتخاذ القرارات بطريقة تحاكي قدرات الإنسان.

تعريف بعض العلماء للذكاء الاصطناعي:

أيلين رج: هو دراسة كيفية توجيه الحاسب لأداء أشياء يؤديها الإنسان بطريقة أفضل.

نيلز نلسن: هدف الذكاء الاصطناعي هو بناء الآلات وتصميمها بحيث تكون قادرة على القيام بالمهام التي تتطلب الذكاء البشري.

مجموعة براتل للأبحاث: ينشأ الذكاء الاصطناعي بواسطة تقنيات مقارنة الصور والتي تساعد في وصف الأشياء او الأحداث او العمليات عن طريق خصائصها النوعية وبهذا يستطيع الحاسب من إيجاد حلول منطقية واستنتاجات ولهذا سميت بحاسبات الجيل الخامس والتي تعتمد على التطور الحاصل فيما يسمى بالمعرفة (Knowledge) وهي أحدى سمات العصر الحالي والذي يسمى بعصر صناعة المعرفة.

و الذكاء الاصطناعي (Artificial Intelligent): واختصاره .AI مصطلح يطلق على علم من أحدث علوم الحاسب الآلي، وينتمي هذا العلم الى الجيل الحديث من أجيال الحاسب الآلي ويهدف إلى أن يقوم الحاسب بمحاكاة عمليات الذكاء التي تتم داخل العقل البشري، بحيث تصبح لدى الحاسوب المقدرة على حل المشكلات واتخاذ القرارات بأسلوب منطقي ومرتب وبنفس طريقة تفكير العقل البشري .

هذه العمليات تتضمن:

- التعليم: اكتساب المعلومات والقواعد التي تستخدم هذه المعلومات .
- التعليل: استخدام القواعد السابقة للوصول إلى استنتاجات تقريبية أو ثابتة .
- التصحيح التلقائي أو الذاتي .

باختصار: هو فرع من فروع علوم الحاسوب يُعنى بميكنة السلوك الذكي عند الإنسان. وفيه نحتاج إلى:

- نظام بيانات :يستخدم لتمثيل المعلومات والمعرفة.

- خوارزميات :نحتاج إليها لرسم طريقة استخدام هذه المعلومات.

لغة برمجة :تستخدم لتمثيل كلاً من المعلومات والخوارزميات.

فهم الذكاء الاصطناعي Understanding AI

من أجل فهم الذكاء الاصطناعي يجب أن نعرف إجابة التساؤلات التالية:

كيف نكتسب المعرفة knowledge ونقوم بتمثيلها represented وتخزينها stored؟

كيف ننتج السلوك الذكي intelligent behavior ونعلمه learned للآخرين؟

كيف نستخدم ونطور ونبرمج خبرات إنسانية مثل الحافز motives والعاطفة emotions وتقدير الاولوية priorities ؟

كيف نستطيع تحويل الإشارات الحسية sensory signals إلى رموز symbols ؟

كيف يتم معالجة الرموز بصورة منطقية محوسبة، من اجل فهم احداث في الماضي والتخطيط plan للمستقبل؟

كيف تستطيع اليات الذكاء انتاج الظواهر الانسانية مثل التوهم illusion والتصديق belief والامل hope والخوف fear والعطف kindness الحلم dreams والحب love ؟

صفات الذكاء الاصطناعي

بالرغم من تنوع المسائل والبحوث والتطبيقات التي تعنون باسم الذكاء الاصطناعي إلا انه توجد صفات مهمة تمتاز فيها كل فروع العلم وهي:

– استخدام الطرق التخمينية بدل الطرق الخوارزمية التقليدية إي أنها تعالج المشاكل التي لا يعرف لها خوارزمية عامة والتي ليس لها ترتيب معروف للخطوات التي تضمن الوصول إلى الحل واختيار طرق تخمينية للمعالجة مع إبقاء المجال مفتوحا لتغييره عندما لا يكون الوصول إلى الحل مضمونا وسريعا .

– استخدام مجال كبير للمعرفة ولكن محدداً لمسالة معينة وهذا أساس النظم الخبيرة.

– هناك احتمالية أن لا تكون الحلول صحيحة لدرجة الكمال وليست على درجة كبيرة من الصدق بل مقنعة.

– التعامل مع الرموز غير العددية ، تكمن أهمية برنامج الذكاء الاصطناعي بصورة عامة في عمليات التفكير التي يقوم فيها البرنامج وهذا يختلف عن الشائع بان الحاسوب يتعامل مع الأرقام ، ولكن يمكن لبرنامج الذكاء الاصطناعي أنجاز الحسابات العددية عند الضرورة.

– إمكانية إعطاء حل ما حتى إن لم تتوفر جميع البيانات المتعلقة بالمشكلة في الوقت الذي يطلب فيه حل لتلك المشكلة .إن عاقبة عدم اكتمال البيانات هو الحصول على حل اقل جودة او تكون الاستنتاجات اقل يقينا.

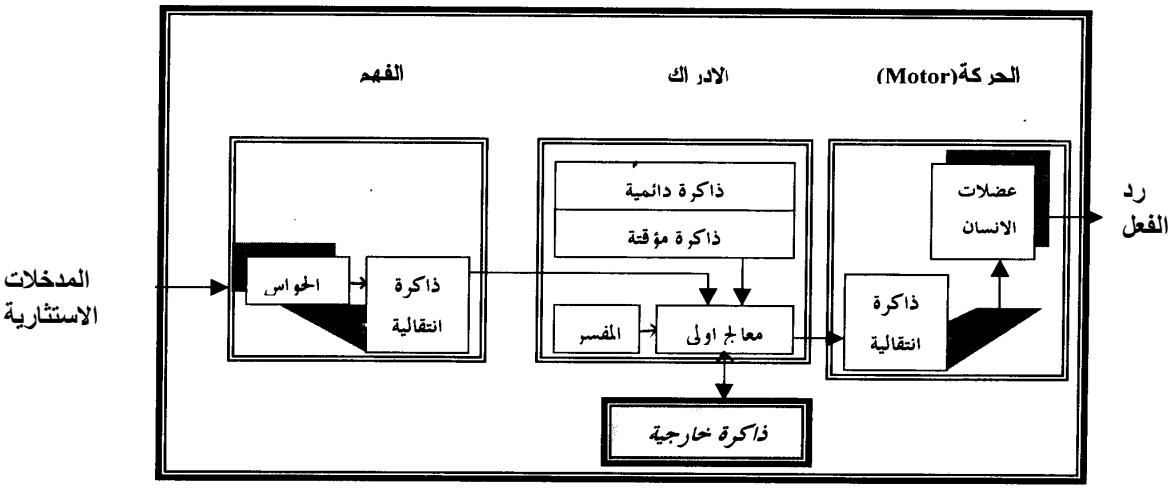
– استخدام مستويات متعددة من المعرفة للحصول على إستراتيجية لحل المسالة وهذه الحالة صعبة ومستخدمة في معظم الأنظمة الحالية.

أسلوب حل المسائل:

تهدف النظم الذكية الى محاولة محاكاة ذكاء الإنسان وذلك عن طريق استخدام الحاسبات. لقد تم وضع العديد من النماذج الخاصة بالمعالجة البشرية للمعلومات ومنها نموذج سايمون_نويل يستخدم هذا النموذج طريقة تناظرية بين المعالجة بالحاسب ومعالجة الإنسان للمعلومات وذلك للوصول الى نموذج قياسي قد يعتمد طريقة الإنسان في حل المشاكل كذلك يساعد على فهم طرق عمل نظم الذكاء الاصطناعي والقيود المفروضة على هذه النظم أثناء التشغيل(الحد من فعاليتها) ويتكون نموذج سايمون نويل للنظام البشري لمعالجة المعرفة من نظم فرعية أخرى هي:

١. نظام فرعي للفهم.
٢. نظام فرعي للإدراك.
٣. نظام فرعي محرك وذاكرة خارجية.

والشكل التالي يوضح معمارية هذا النظام ويوضح كل من الذاكرة والمعالجة المستخدمة في كل نظام فرعي:



معمارية سايمون – نويل

تتمثل المدخلات في اســـتثارة الحواس من العين مثلا وتظل في ذاكرة انتقائية انتظارا للمعالجة عن طريق النظام الفرعي للإدراك. وتســـاعد هذه الذاكرة الانتقائية في الاحتفاظ بكمية هائلة من المعلومات حيث يقوم النظام الفرعي للإدراك باستخدام المناسب منها لحل مسألة معينة أو اتخاذ قرار.

وتماما كما يحدث في وحدة المعالجة المركزية (CPU) فان المعالج المبدئي يقوم باســـتدعاء هذه البيانات عند الحاجة لها لاتخاذ القرار ويقوم بتحويلها الى الذاكرة القصيرة او المؤقتة وتكون هذه العملية بشكل مكرر وبشـكل حلقات(Cycles) وفي كل حلقة يقوم المعالج باسـتدعاء البيانات من الذاكرة الانتقالية ثم يتم خزنها وتقييمها في الذاكرة (باســتخدام نظم خاصـــة) ويحتوي النظام الفرعي للإدراك على المعالج المبدئي ، الذاكرة قصيرة المدى، الذاكرة طويلة المدى، المفسر الذي يقوم بتفسير أجزاء او كل تعليمات البرنامج الخاص بحل المشـــكلة وهذا البرنامج بدوره يعتمد على عدد من المتغيرات مثل المهام المطلوب انجازها ودرجة ذكاء النظام الذي يســاعد في حل المشـكلة وتكون هذه الأنظمة مخزونة وتســحب عند الحاجة . في حالة انجاز بعض المهام المعقدة والتي تحتاج الى قدر كبير من المعلومات يتم الاستعانة بالذاكرة طويلة المدى بالإضافة الى الاستعانة بذاكرة خارجية أضافية (كما هو الحال في المعلومات الموجودة داخل الكتب).

يتفوق الحاسب على الإنسان في سرعة استرجاع هذه المعلومات من الذاكرة الخارجية وفي تخزينها ومعالجتها كذلك في تكامل البيانات أثناء حل المشكلة ، ويمكن في الوقت الحاضر استخدام أسلوب المعالجة المتوازية

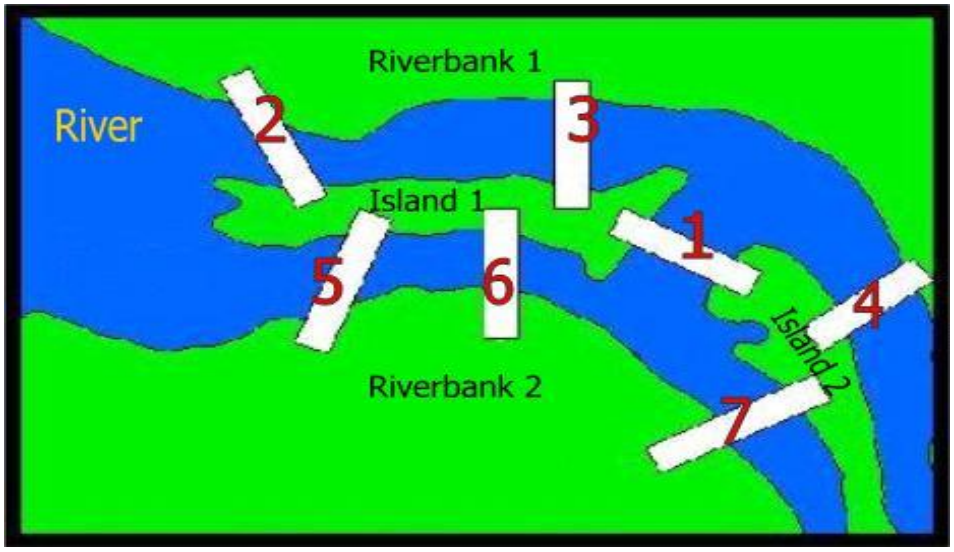
(Parallel Processing) في نهاية عملية البحث خلال الذاكرة يقوم المعالج بإرسال المعلومات المطلوبة الى الجزء الفرعي المحرك الذي يقابل الجهاز العضلي للجسم والذي يقوم بتحريك العضلات معطيا بذلك رد الفعل العضلي للاستثارة التي استقبلتها الحواس.

أكد العالمان سايمون ونويل على ان الفعالية الذكائية سواء كانت للإنسان او الآلة تتم من خلال مايلي :

١. نماذج بسيطة لتمثيل الجوانب المهمة لمجال المسألة.
٢. العمليات على تلك النماذج لتوليد الحلول للمسائل.
٣. البحث لاختيار حل من بين الحلول الممكنة.

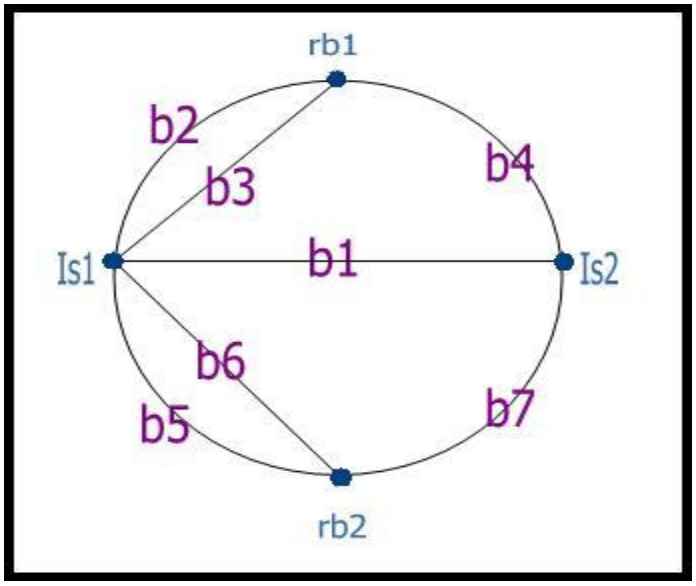
نظرية التخطيط البياني Graph Theory

كان عالم رياضيات نمساوي اسمه Leonhard Eular السبب بعد إرادة الله سبحانه، في اختراع نظرية اسمها Graph Theory في أوائل القرن الثامن عشر، اخترع هذه النظرية لإيجاد حل للمشكلة التي واجهته حينما زار مدينة Königsberg في ألمانيا، هذه المدينة يخترقها نهر river عليه جزيرتين صغيرتين Island1 & Island2. ترتبط هاتان الجزيرتان بضفتي النهر Riverbank1 & Riverbank2 وببعضهما البعض عن طريق سبعة من الجسور Bridges ، كما توضح الصورة التالية:



أراد هذا العالم التجول في إرجاء هذه المدينة بكاملها بدون المرور على جسر أكثر من مرة! لدراسة إمكانية ذلك؛ قام العالم برسم خريطة توضيحية بسيطة للمدينة، مثّل فيها الجهات التي يريد التنقل فيما بينها (كلاً من Is1, Is2, rb1 and rb2) كنقاط أو أطراف nodes ، ثم مثَّل فيها كل جسر كوصلة link تربط بين هذه الأطراف كما هي موجودة في أرض الواقع، هذا التمثيل سمِّيَ Graph أو تمثيل بياني :

-حيث أن **Is= Island, rb= Riverbank and b=bridge**



بعد ذلك أوجد مفهوم جديد يسمى Degree of the Node of the Graph أو درجة الطرف في التخطيط، حيث أن لكل طرف درجة هذه الدرجة هي عدد الوصلات التي تصل هذا الطرف مع الأطراف الأخرى المجاورة له، أي عدد الـ links الداخلة أو الخارجة من هذا الطرف، من الممكن أن يكون هذا العدد فردي أو زوجي بطبيعة الحال. لنوجد معاً درجة كل طرف في التخطيط:

Node	Degree
Is1	5
Is2	3
rb1	3
rb2	3

توصل بعد ذلك إلى النظرية التي تقول:

يمكنك حل المشكلة والمشي في أرجاء المدينة مع العبور على كل جسر مرة واحدة فقط في حالتين:

١. إذا كان لديك طرفين فقط يحملون درجة فردية .two odd degree nodes
٢. إذا لم يكن لديك ولا طرف من درجة فردية zero odd degree node بمعنى أن جميع الأطراف لديك من درجة زوجية.

عدا ذلك فإن المشكلة مستحيلة الحل ولا يمكنك المشي في أرجاء المدينة دون العبور على أحد الجسور أكثر من مرة!

ثم حدد مسار السير في الحالات التي يمكن حل المشكلة فيها، كالتالي:

- إذا كان لديك طرفين من درجة فردية، فإن السير سيبدأ من الطرف ذو الدرجة الفردية الأول، وينتهي عند الطرف ذو الدرجة الفردية الثاني .
- إذا لم يكن لديك ولا طرف من درجة فردية، بمعنى أن كل الأطراف من درجة زوجية، فإن المشي سيبدأ من أحد هذه الأطراف وينتهي عند نفس الطرف!

ثم أقرّ بأنه لا يمكنه التجوال في أرجاء مدينة Königsberg بدون العبور على جسر أكثر من مرة، لأنه يوجد أربعة أطراف من درجة فردية في graph هذه المدينة!!

عرفت هذه المشكلة باسم "Bridge of Königsberg problem" ومؤخراً أصبحت معروفة باسم العالم : "Finding an Eular path through a graph"، وهي تعتبر حجر الأساس وأول نظرية في عالم الـ Graph .

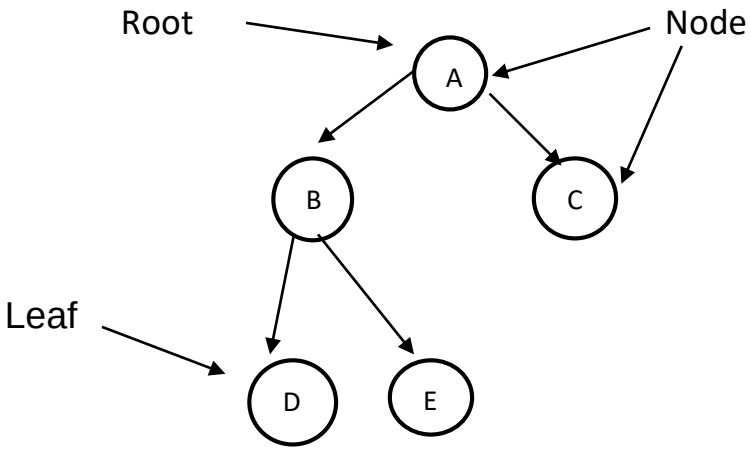
وهذا يقودنا للسؤال :ما معنى كلمة Graph ؟!

الـ Graph كما رأينا هو مجموعة من الأطراف nodes تربط ما بينهما مجموعة من الوصلات links، من الممكن أن نعتبر كل طرف يمثّل حالة، وللإنتقال من حالة إلى أخرى نستخدم الوصلة التي تصل بينهما.

اذا كان هناك اتجاه مرافق لـ Graph هذا الـGraph يسمى Graph متجه .

الجذر Root: هي النقطة التي ليس لها اب .

الورقة Leaf: هي النقطة التي ليس لها ابناء .



Path: هو تسلسل مرتب من النقاط [N1,N2,…..,Ni] حيث Ni هي الاب لـ Ni+1 ويطلق عليه مسار ذو طول = i

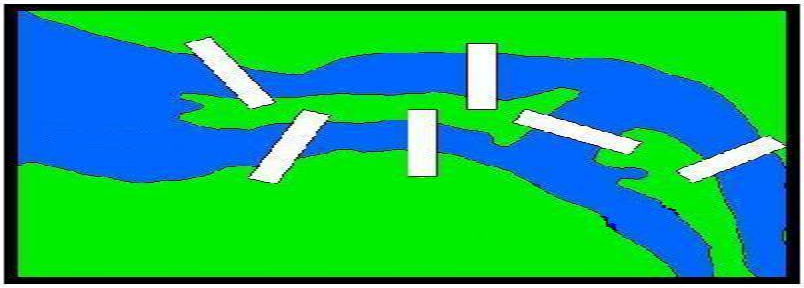
Cyclic Path: هو المسار الذي يحوي نفس النقطة اكثر من مرة .

Tree الشجرة : هي نوع من الـ Graph لا يحتوي على مسار فيه دارة .

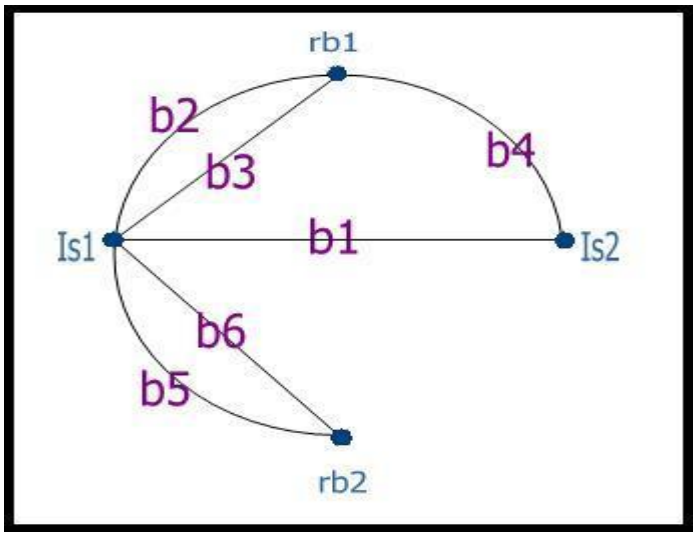
- A tree is a connected graph without cycles
- A connected graph is a tree if it has N vertices and N-1 edges
- A graph is a tree if there is one and only one path joining any two of its vertices

يفيد تمثيل المشـاكل بهذه الطريقة في اختزال واحتواء المشـكلة، وزيادة فهمها مما يسـهل الطريق إلى حلها، كما تعتبر Graph Theory أفضل أداة للتعليل reasoning في أي تركيب structure يحوي مجموعة من الكائنات objects بينهما مجموعة من العلاقاتrelations . في علوم الذكاء الاصـطناعي، تستخدم هذه النظرية في تقنيات البحث وخصـوصـاً في State-space search بنوعيها Depth-first and Breadth-first.

المثال التالي نطبق عليه النظرية ونرى هل يمكننا حل مشكلته أم لا؟! انظرهذه الخريطة:



أول خطوة، نمثلها كـ:Graph



ثم نوجد درجة كل node فيها:

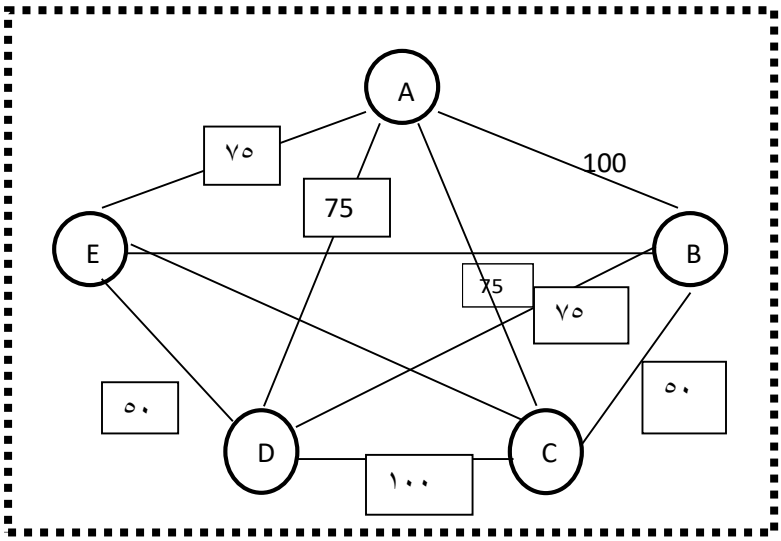
Node	Degree
Is1	5
Is2	2
rb1	3
rb2	2

ثم نحدد عدد الأطراف من الدرجة الفردية، يوجد لدينا طرفين من درجة فردية. إذن المشكلة ممكنة الحل والمشي بدون العبور أكثر من مرة على كل جسر ممكن. لنحدد معاً مسار المشي الذي لابد من أن يبدأ من أحد الأطراف ذات الدرجة الفردية (Is1 or rb1)وينتهي عند الطرف الآخر، من الممكن أن يكون أحد المسارات التالية:

1. Is1(throughb5)→rb2(b6)→Is1(b2)→rb1(b3)→Is1(b1)→ Is2(b4)→rb1
2. Is1(b1)→Is2(b4)→rb1(b3)→Is1(b5) →rb2(b6) →Is1(b2)→ rb1
3. Is1(b2)→rb1(b3)→Is1(b5) →rb2(b6) →Is1(b1)→Is2(b4)→ rb1
4. rb1(b2) →Is1(b5)→rb2(b6)→Is1(b3)→rb1(b4)→Is2(b1) → Is1
5. rb1(b4)→Is2(b1)→Is1(b5)→rb2(b6)→Is1(b2) →rb1(b3) → Is1

مسالة البائع المتجول: Traveling Sale's man problem

تعرف مسالة البائع المتجول Traveling Sale's man problem على أن البائع المتجول يرغب بالقيام بجولة مبيعات يزور فيها عدد معين من المدن وذلك لعرض نماذج تلك المبيعات ،وذلك انطلاقا من مدينة معينة وانتهاء بنفس المدينة على شـــرط أن لا يزور أي مدينة أكثر من مرة واحدة ،وهو يأمل لأسباب اقتصادية ان تكون المسافة التي يقطعها اقل ما يمكن .



GRAPH

1- Path 1 : A → E → D → B →C → A

A → E =75
E → D =50
D → B =75
B → C = 50
C → A = 75

الكلفة الكلية = المجموع = ٣٢٥

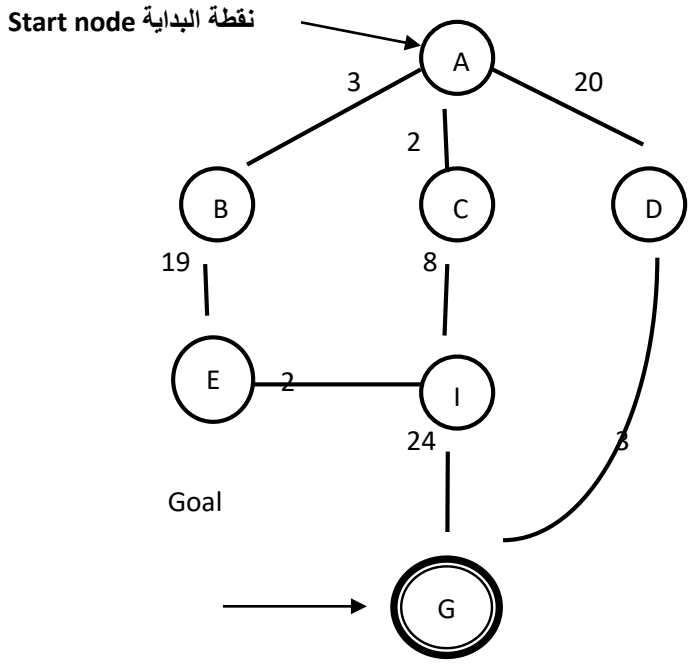
2- Path 2 : A → E → D → C →B → A

A → E =75
E → D =50
D → C =100
C → B = 50
B → A = 100

} الكلفة الكلية = المجموع = ٣٧٥

عدد الاحتمالات = مفكوك عدد الـ Node في هذا المثال عدد الاحتمالات = !5

أذا تم استخدام الطرق التقليدية في حل مسالة البائع المتجول مثل طريقة الـ(Simples) أو طريقة السرد الكلي (Total Enumeration) وباستخدام الحواسيب العملاقة فان العملية سوف تأخذ وقت طويل جدا وخاصـــة عندما يكون عدد المدن كبير ولكن يمكن معالجة هذه المســـألة باســـتخدام الطرق الذكية مثل (Heuristics) والتي تعطي الحل لأكثر من ١٠٠ مدينة في وقت قصـير جدا وبالاعتماد طبعا على نوعية المعالج المستخدم.



Path 1: A→ B →E →I → G =3+19+2+24=48

Path 2: A→ D→ G= 20+3=23

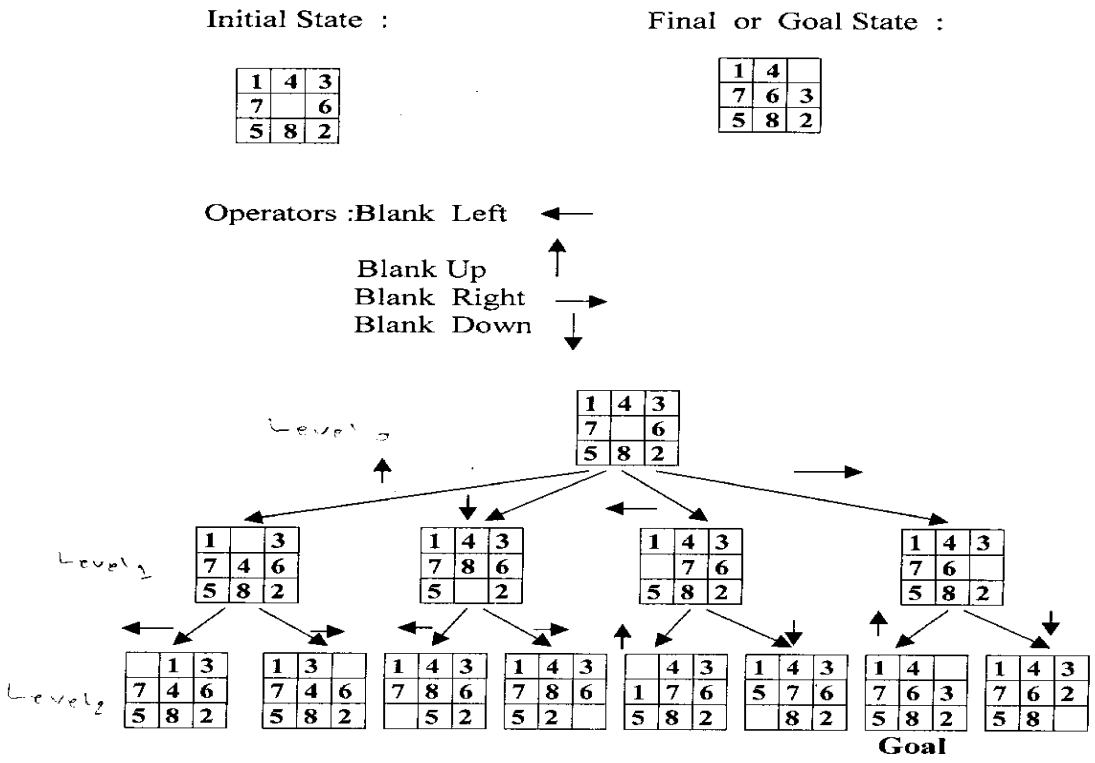
Path 2: A→ C → I → G =2+8+24=34

لعبة الأحجار الثمانية 8-puzzle problem

استخدمت خوارزميات الذكاء اصطناعي في حل الكثير من المسائل التي تحتاج الى وقت وجهد كبير في الحل باستخدام الطرق التقليدية من هذه المسائل هي لعبة الاحجار الثمانية وهي عبارة عن لوح مقسم الى تسع مربعات صغيرة Puzzlesيضم كل منها رقم معين وهناك مربع فارغ لايحمل اي رقم يستخدم للتبديل مع المربعات الاخرى والمطلوب ترتيب هذه الارقام المبعثرة داخل اللوح (نقطة البداية) بشكل معين للوصول الى (نقطة الهدف) او الشكل النهائي.

(في حالات اخرى يكون اللوح على شكل صورة مقسمة على شكل مربعات وتكون مبعثرة – نقطة البداية – والمطلوب اعادة ترتيبها للحصول على الصورة الاصلية – نقطة الهدف–).

لايجاد اقصر مسار بين نقطة البداية ونقطة الهدف يتم تحريك المربع الفارغ (الابيض) وتبديله مع المربعات المجاورة حسب نظام تسلسل حركة معين موضح في الشكل التالي (يسار ، اعلى، يمين ، اسفل)، ويمكن استخدام نظام حركة مغايير حسب متطلبات المسألة كما موضح في الحل في الشكل التالي اذ تم استخدام نظام حركة مغايير للنظام الافتراضي من اجل التوضيح وكما في المثال التالي في الشكل اذا تم اعتماد نظام حركة للمربع الخالي وهو (اعلى ، اسفل، يسار، يمين)



ان معظم مشاكل الذكاء الاصطناعي يمكن ان تولد هياكل معرفة كبيرة جدا قد لا تسعها ذاكرة الحاسبة بالاضافة الى كون تلك الطرق بطيئة جدا حتى عند استيعاب ذاكرة الحاسبة لها .عملية توليد الهياكل تتم بصورة تدريجية بمستويات يتم البحث في كل مستوى (Level) عن الهدف. طرق البحث التوليدية المستخدمة في الذكاء الاصطناعي تقسم إلى الأنواع التالية .

١. طرق بحث نظامية (Systematic Methods).
٢. طرق بحث أمثلية (Optimal Methods).
٣. طرق بحث تنقيبية (حدسية) (Heuristic Methods).
٤. طرق بحث جينية (Genetics Methods).
٥. طرق بحث موجهه (Adversary Methods).

طرق البحث النظامية تقسم إلى عدة أقسام من أشهرها :

١. بحث عمودي (Depth – First Search).
٢. بحث أفقي (Breadth – First Search).

وتعتمد طريقة اختيار احد النوعين أعلاه على توقع موقع الهدف في الشجرة ، فإذا كان الهدف قريب من نقطة البداية فان الاختيار الأفضل هو البحث الأفقي. وإذا كان توقع الهدف بعيدا عن نقطة البداية فان الاختيار الأفضل هو البحث العمودي. ان عملية توقع موقع الهدف تحتاج الى دراسة المشكلة والى خبرة بسيطة.

خوارزمية البحث العمودي Depth-First Search Algorithm

Depth-first search (**DFS**) is an algorithm for traversing or searching a tree, tree structure, or graph. Intuitively, one starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking.

Formally, DFS is an uninformed search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hadn't finished exploring. In a non-recursive implementation, all freshly expanded nodes are added to a LIFO stack for expansion

تعتبر من اهم طرق البحث الاعمى Blind Search فكرتها هي البحث باتجاه العمق بمسار واحد ومن جهة اليسار الى ان نصل الى نقطة الهدف او نصل الى نقطة ليس لها ابن، في هذه الحالة تبدا عملية العودة Backtracks الى الاعلى وذلك لاختبار النقاط الاخرى تستخدم في الحل مجموعتين هما الـ Open وClose كلاهما يتم اضافة وحذف العناصر منهما من جهة اليسار. وهي توصل للحل بسرعة اذا كان الهدف بالعمق حيث انها كفوءة للبحث في تفرعات كثيرة .

```
Procedure Depth-First Search
Initialize : open=[start node], close[ ];
While open <> [ ] Do
        Remove the first state from left of open, call it x.
    If x is the goal then return (path)
    Generate all children of x.
    Put x in close.
    Remove from open any children of x already in open.
```

Discard any children of x already in close.

Add the remaining children of x to the left of open.

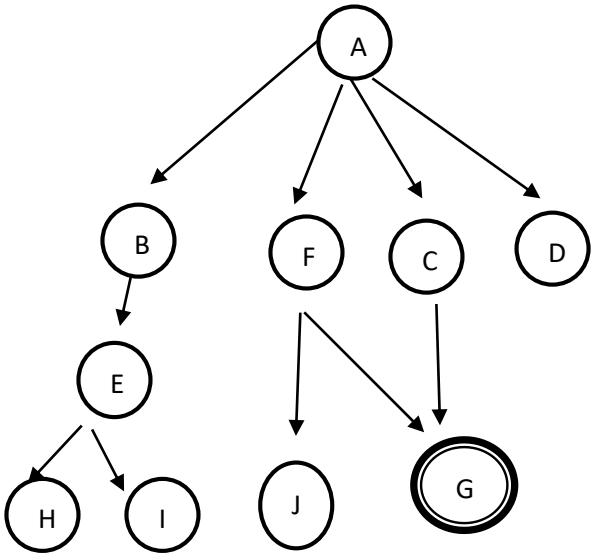
Return(fail).

End.

Example:

Start state=A

Goal state=G



Iteration	Open	Close
0	[(A,0)]	[]
1	[(B,A), (F,A), (C,A), (D,A)]	[(A,0)]
2	[(E,B),(F,A),(C,A), (D,A)]	[(B,A), (A,0)]
3	[(H,E),(I,E),(F,A),(C,A),(D,A)]	[(E,B),(B,A), (A,0)]
4	[(I,E),(F,A),(C,A),(D,A)]	[(H,E), (E,B),(B,A), (A,0)]
5	[(F,A),(C,A),(D,A)]	[(I,E),(H,E),(E,B),(B,A),(A,0)]
6	[(J,F),(G,F), (C,A),(D,A)]	[(F,A),(I,E), (H,E), (E,B),(B,A),(A,0)]
7	[(G,F), (C,A),(D,A)]	[(J,F),(F,A),(I,E),(H,E),(E,B),(B,A),(A,0)]
8	G is the Goal	[(G,F),(J,F),(F,A),(I,E),(H,E),(E,B),(B,A), (A,0)]

State Space:- All the node of tree

فضاء الحالة: كل النقاط التي دخلت في مصفوفة الـ OPEN

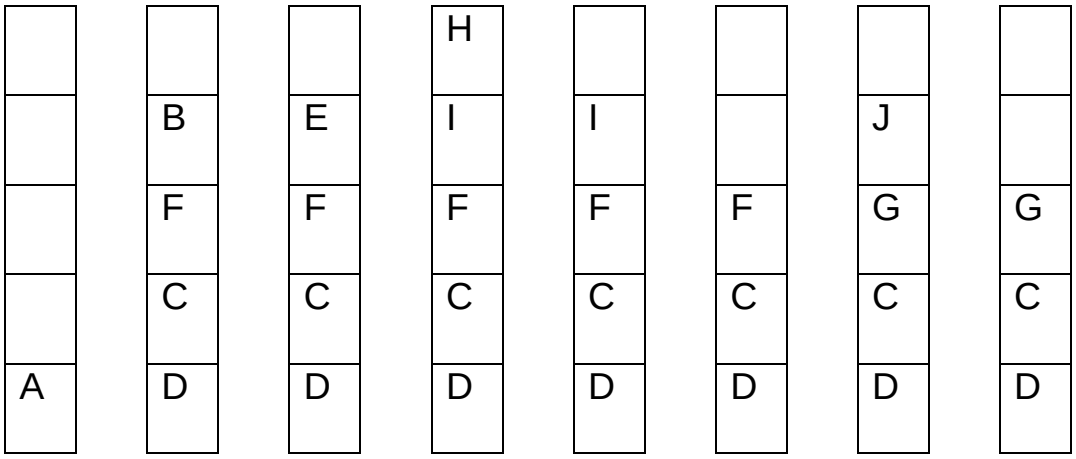
Search Space:- A,B,E,H,I,F,J,G

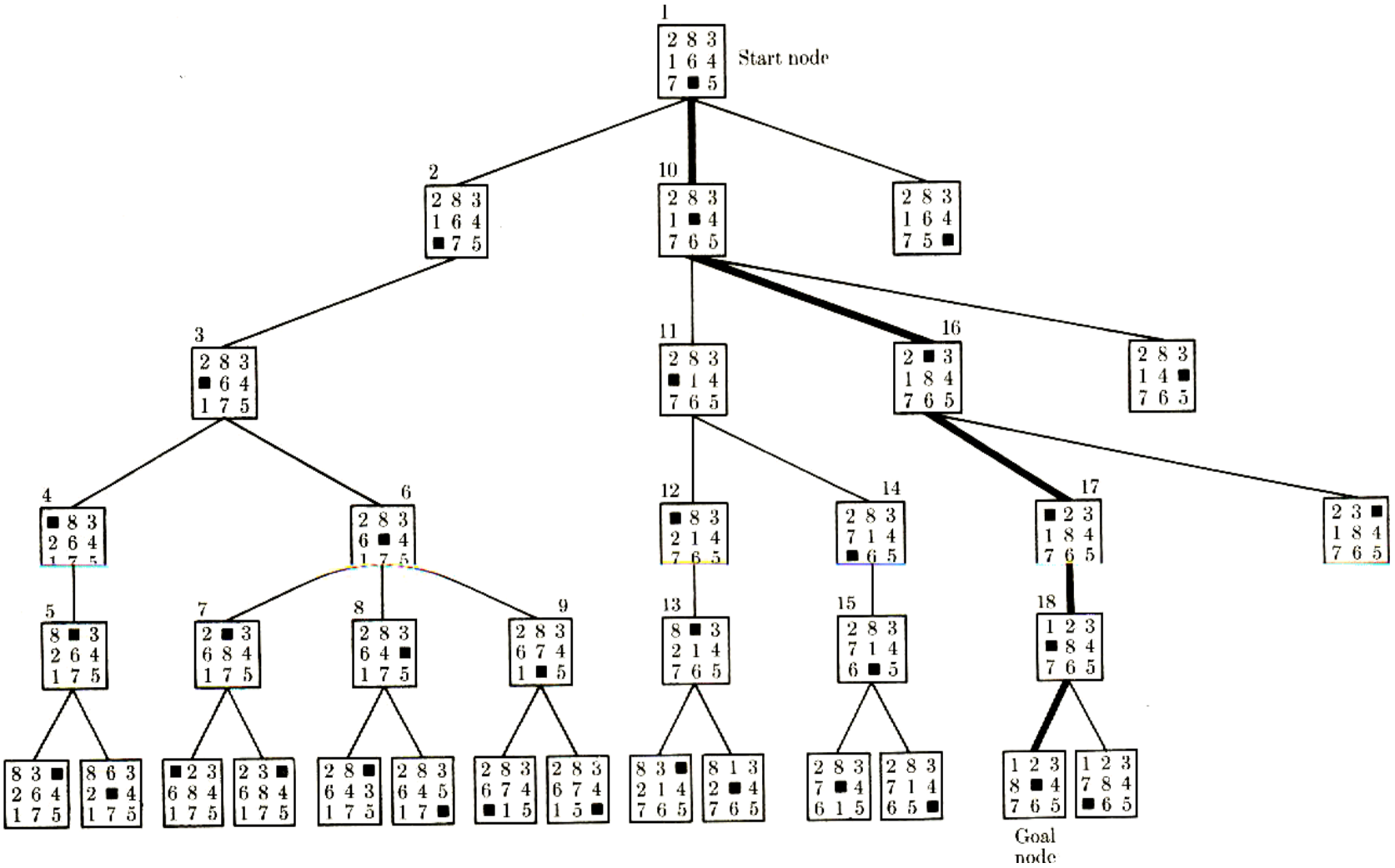
فضاء البحث :كل النقاط التي تم زيارتها واختبارها هل هي نقطة الهدف ام لا

Solution Path:- A→ F → G

اقصر طريق او مسار من نقطة الجذر الى نقطة الهدف

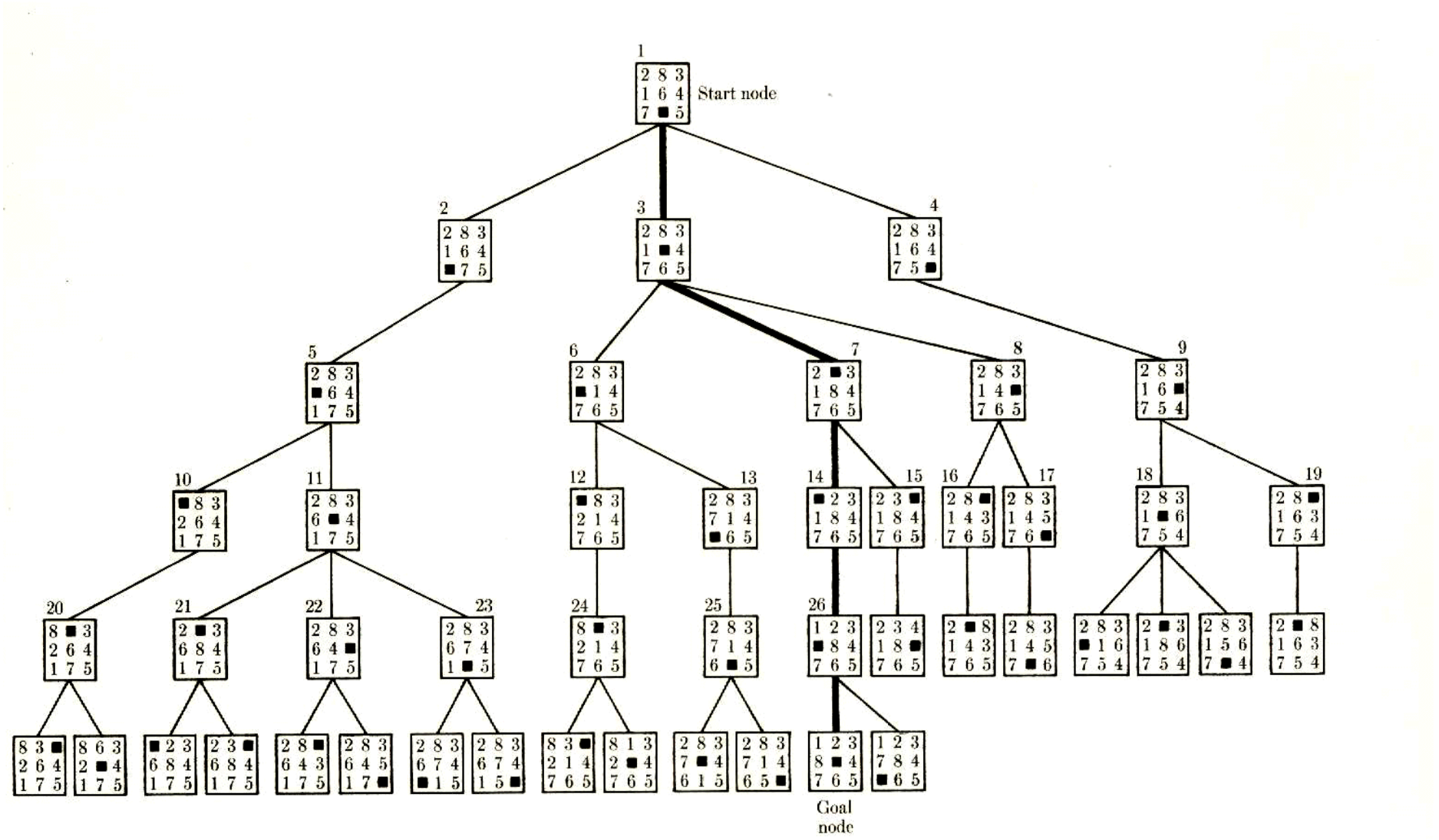
هذه الخوارزمية تشبه مبدا الـ stack أي انها تعتمد على مبدا LIFO العنصر الداخل اخيرا يخرج اولا





ايجاد اقصر مسار ل 8- Puzzle باستخدام طريقة البحث العمودي التوقف عند المستوى الخامس

ايجاد اقصر مسار باستخدام طريقة البحث الافقي


Start node
Goal node

Breadth-First Search Algorithm خوارزمية البحث الأفقي

Breadth-first search (BFS) is a graph or tree search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

تعتمد هذه الخوارزمية في عملية البحث على Blind Search ايضا لانها من طرق البحث الاعمى و السير في مستويات افقية حيث تنتقل من مستوى الى الذي يليه الى ان تصل الى الهدف.

Procedure Breadth-First Search

Initialize : open=[start node], close[ ];

While open <> [ ] Do

Remove the first state from left of open, call it x.

If x is the goal then return (path)

Generate all children of x.

Put x in close.

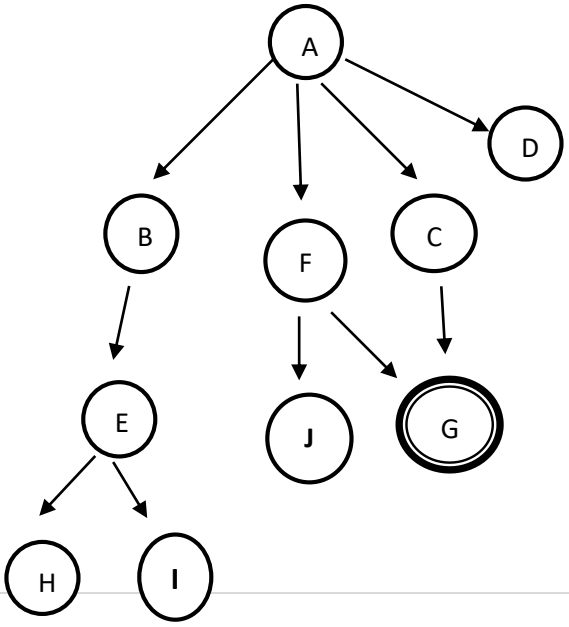
Remove from open any children of x already in open.

Discard any children of x already in close.

Add the remaining children of x to the right of open .

Return(fail).

End.



ايجاد اقصر مسار في الشجرة السابقة باستخدام طريقة البحث الافقي الـ Breadth-first search

Iter	Open	Close
0	[(A,0)]	[]
1	[(B,A), (F,A), (C,A), (D,A)]	[(A,0)]
2	[(F,A), (C,A), (D,A) (E,B)]	[(B,A), (A,0)]
3	[(C,A),(D,A),(E,B),(J,F),(G,F)]	[(F,A),(B,A), (A,0)]
4	[(D,A),(E,B),(J,F),(G,F),(G,C),(F,C)]	[(C,A), (F,A),(B,A), (A,0)]
5	[(E,B),(J,F),(G,F),(G,C),(F,C)]	[(D,A),(C,A), (F,A),(B,A), (A,0)]
6	[(J,F),(G,F),(G,C),(F,C),(H,E),(I,E)]	[(E,B),(D,A),(C,A), (F,A),(B,A), (A,0)]
7	[(G,F),(G,C),(F,C),(H,E),(I,E)]	[(J,F),(E,B),(D,A),(C,A), (F,A),(B,A), (A,0)]
8	G is the Goal	[(G,F)(J,F),(E,B),(D,A),(C,A), (F,A),(B,A), (A,0)]

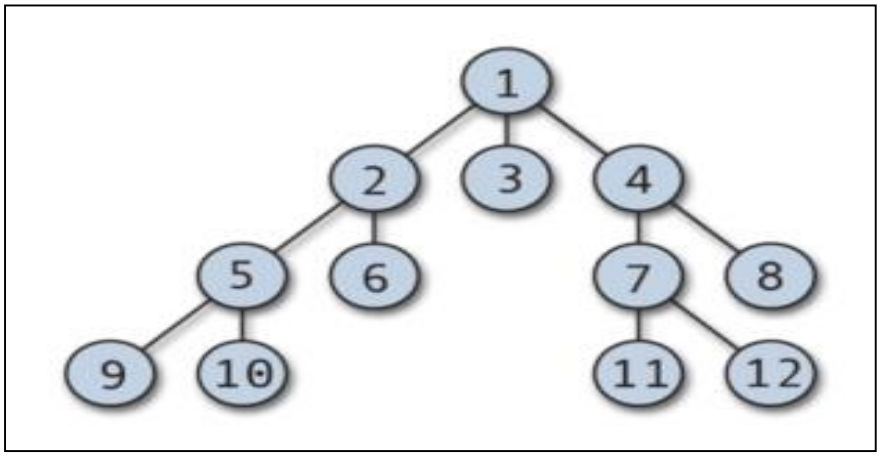
State Space: All the nodes of Graph

Search Space: A,B,F,C,D,E,J,G.

Solution path: A→F→G.

واجب

جد اقصـــر مســـار من النقطـة ١ الى النقطـة ١٢ بـاســـتخـدام خوارزميتي البحث العمودي و الافقي



2- طرق البحث التنقيبية (الحدسية) (Heuristic Methods)

الحدس (الاجتهاد)(التخمين) (Heuristic) هو الحكم على الأشـــياء بالخبرة التجريبية وهو يســـاعد الإنســـان على اتخاذ القرار فيما سـيفعله مسـتقبلا لذا يعتبر الحدس هو احد عناصـر الذكاء الاصـطناعي الأساسية . حيث يمكن تعريفه وفق مفهوم الذكاء الاصطناعي الأتي:-

الذكاء الاصطناعي: هو ذلك النوع من علوم الحاسبات الذي يتخذ أسلوب المعالجة المرمزة (Symbolic Processing) لتمثيل المعرفة وليس المعالجة العددية (Numerical Processing) كذلك الحدس يحاكي (simulate) أسلوب الحدس عند الإنسان في معالجة المعلومات وكمثال على ذلك.

١- استخدام الحدس في توقع الأحداث المستقبلية .

مثلا **(يبدو أن السماء ستمطر --> يجب اخذ مظلة)**

٢- يمكن استخدام الحدس في تحديد الأفعال المستقبلية

مثلا **(سأضبط المنبه على الساعة السادسة حيث أني احتاج ساعة للإفطار وارتداء الملابس والذهاب الى العمل في السابعة)**

وللأسباب أعلاه فان طرق الحدس تستخدم للتغلب على مشكلة الزيادة في عدد المرات التكرارية اللازمة للبحث في فضاء الحالات للطرق العشوائية السابقة قبل الوصول الى الحل أي تقليل وقت المعالجة والخزن.

ميزات طرق الحدس:

١- مرونة كبيرة في التعامل مع التراكيب الصعبة والمشاكل المعقدة.
٢- كفاءة عالية في حالة عدم مطابقة البيانات للواقع او قد تكون البيانات المتوفرة غير كافية للاستنتاج واتخاذ القرار.
٣- قدرة على التحليل النوعي لذا فهي تناسب حالات اتخاذ القرار،
٤- يمكن استخدامها كجزء من إجراءات تكرارية لضمان الوصول الى الحل المثل أي انه يمكن دمجها مع طرق أخرى للحصول على أسلوب جيد للحل.
٥- عند تمثيل المشكلة باستخدام الشجرة البحثية فان استخدام أسلوب الحدس يؤدي الى إنقاص حجم الشـجرة وذلك بحذف العقد التي لا نتوقع ان تسـاعد كثيرا في الوصـول للحل المناسـب (Non-Promising nodes).

في هذا النوع من خوارزميات البحث سـوف تجري عملية مفاضـلة بين نقطة وأخرى بالاعتماد على قيمة عددية وهذه القيمة قد تكون كبيرة او صـغيرة حسب نوع المسالة كان نريد إجراء اقل كلفة فنأخذ اقل قيمة او اكبر سعر في هذه الحالة نأخذ اكبر قيمة .

ومن الخوارزميات المستخدمة في هذا النوع من البحث

١- البحث التجميعي او المسمى بطريقة تسلق التلال **Hill–Climbing Search**

٢- البحث اللحظي او المسمى الأفضل أولا **Best – First Search**

خوارزمية تسلق التلال HILL - CLIMBING ALGORITHM

```
Begin
Cs=start, Open=[start], Stop=false.
                    Path=[start]
          While(not stop) do
Begin
                If(Cs=Goal)then return (path);
Empty Open;
     Generate all children of Cs and put them in open;
If (Open [ ])then stop=true
Else
Begin
          X=Cs;
For each state Y in open Do
Begin
```

Compute the heuristic value of Y , h(Y);

If (Y better than X)then X=Y

End

If (X is better than Cx) then Cx=X. add Cx to path

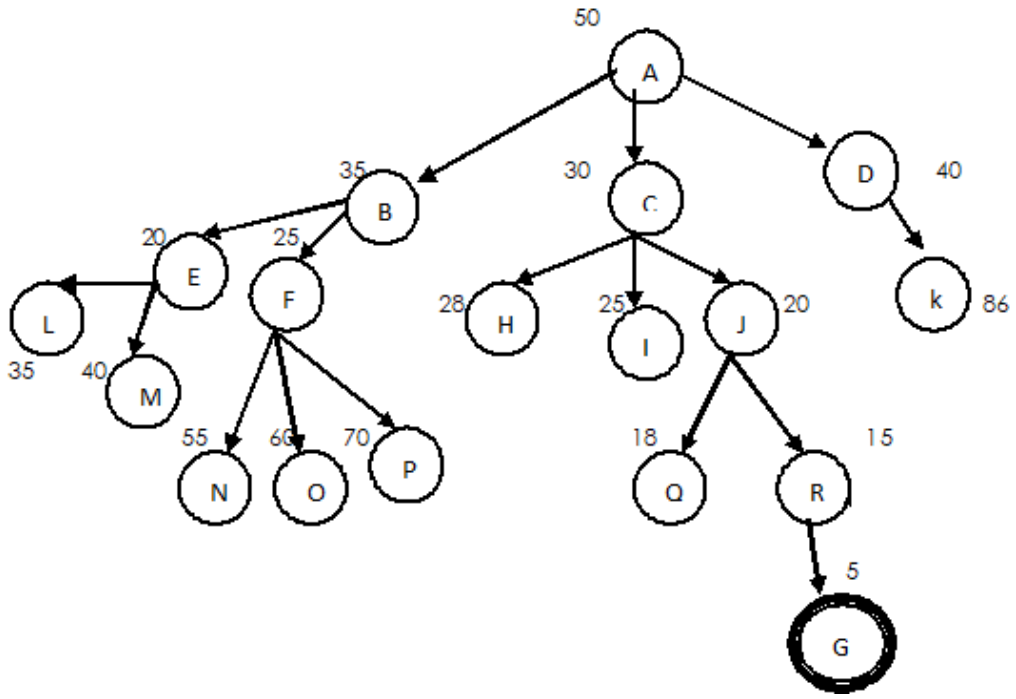
Else stop=true

End

Return(fail)

End

End.



	Cs	Open	Path	
0	A	[A]	[A]	
1	C	[B,C,D]	[A,C]	X=A=50 X=B=35 X=C=30 الاقل كلفة

2	J	[H,I,J]	[A,C,J]	X=C X=H=28 X=I=25 X=J=20
3	R	[Q,R]	[A,C,J,R]	X=J X=Q=18 X=R=15
4	G	[G]	[A,C,J,R]	X=R X=G IT IS THE GOAL

Solution Path :A→ C→ J→R→G

استخدام خوارزمية تسلق التلال في حل مسألة الأحجار الثمانية

Hill–Climbing Search (8-puzzle example)

يتم حساب كلفة كل نقطة بالاعتماد على المعادلة التالية

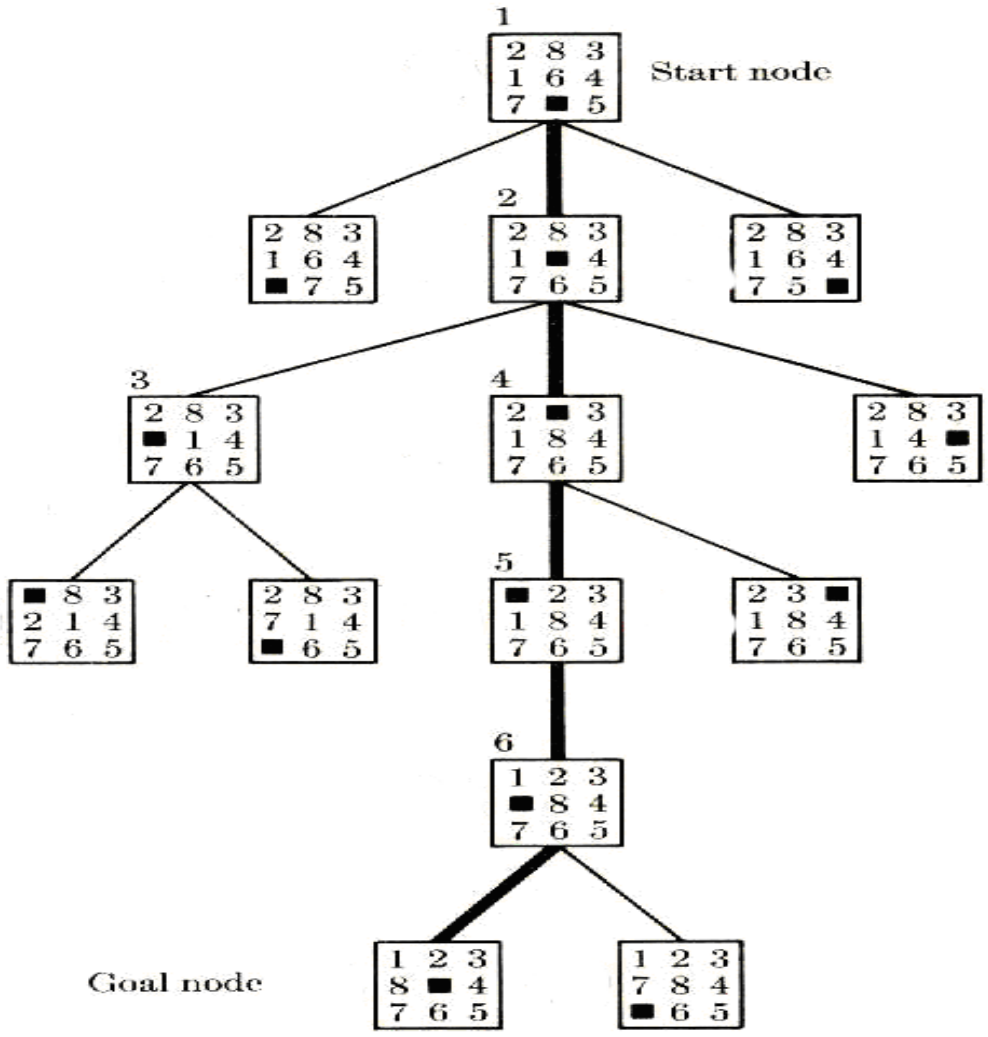
Evaluation Function: **F(n)=g(n)+h'(n),**

Where g(n)=depth of n العمق (المستوى الذي تقع فيه النقطة)

h'(n)= function counts the number of placed tiles in the state description associated with node n.

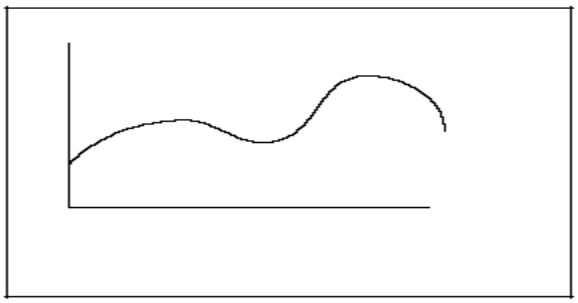
عدد الأحجار التي في مواقعها

الكلفة(للنقطة) = العمق (المستوى الذي تقع فيه النقطة) + عدد الاحجار التي في مواقعها

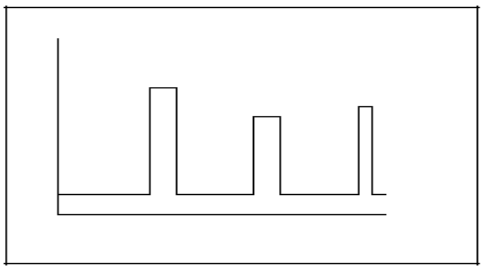


مشاكل الـ HILL CLIMBING

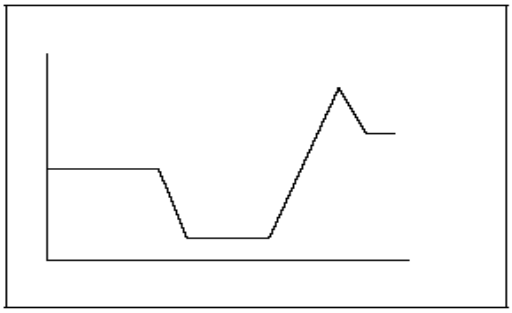
١- احتمال وجود صعود موضعي فيمكن اعتبار (Local Maximum)او يسمى (Foothills).



٢- وجود مساحة مستوية (Plateau) كلما نطبق القاعدة يظهر نفس الحل أي لايوجد صعود.



١- او قد تأتي منطقة نزول (Ridge) فعند تنفيذ الخطوات التالية ســـتؤدي الى تكرار تنفيذ الـ rule مما يؤدي الى انهيار في الدالة التخمينية أي ان باتجاه الحل توجد حافة منخفضـــة كلما تقدمنا في الحل تنقص الدالة التخمينية.



الحلول لمشاكل الـ HILL CLIMBING

١- الرجوع إلى الوراء أي إلى Node سابقة ومحاولة الذهاب في اتجاهات مختلفة.

٢- عمل قفزة طويلة في بعض الاتجاهات للحصول على مجال جديد في البحث.

٣- طبق قاعدة أو قاعدتين أو أكثر قبل عملية الاختيار. أي عمل حركة بعدة اتجاهات قبل عملية الاختيار.

Solutions to problems:

Foothills - backtracking

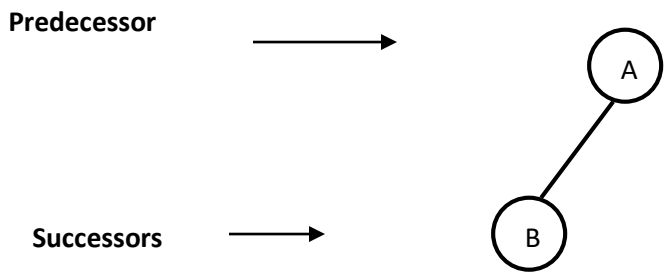
Plateaus - large step size

Ridges - apply multiple moves before testing heuristic

في هذه الخوارزمية قد لا نصل الى الحل لأنه لايوجد رجوع (Backtracking) كذلك لايوجد close

إذا أضفنا الـ backtracking الى الـ Hill Climbing سوف نحصل على طريقة جديدة هي الـ Best-first search.

تسمى الـ node (سلف Predecessor)اذا كانت تسبق النقاط الاخرى مثلا نقطة الجذر هي النقطة الـPredecessor لكل النقاط الموجودة في الشجرة.



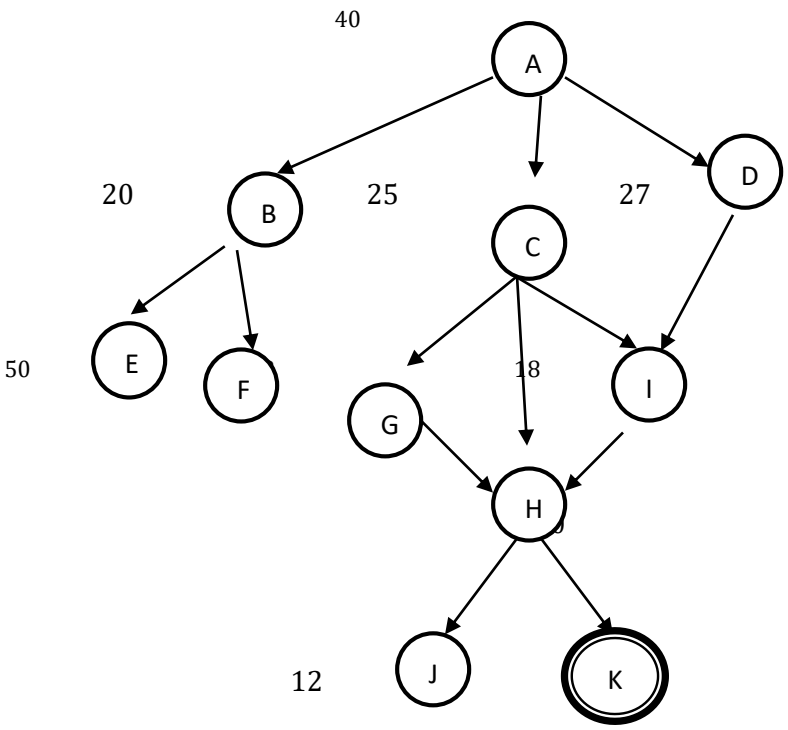
تسمى الـ node (خلف Successors) اذا كانت تلي النقاط الاخرى مثلا النقطة B هي Successors للنقطة A

Best – First Search طريقة البحث اللحظي او الأفضل أولا

Best – First Search Algorithm

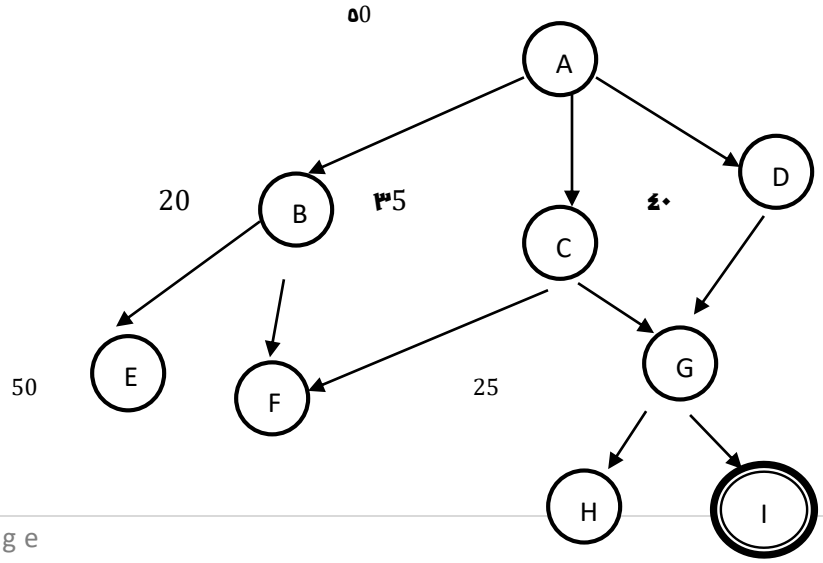
```
Begin
Input the start node S and the set G of goal nodes
Open =[s],close=[ ];
Pred[s]=0,found=false;
Cs=s;
While (open<>[ ])&(found=false) do
Begin
L=the set of nodes on open for which F has the best value;
If(L is singleton)then X = that element
Else
   If there are any goal nodes in L
   Then X= one of them
    Else
        X= the first element of L
        Remove X from open and put it on close
        If (X is a goal node) then found=true
       Else
      Begin
    Generate the set successor of children  of X for each child Y do
      If (Y is not already  on open  or close)then
        Begin
          Compute h[Y] , pred [Y]=X
           Insert Y in open;
```

```
        End
    End
      If found = false then output "failure"
        Else trace the path using pred pointers
  End
End .
```



Cs	Open	Close
A	A	
B	B_{20} C_{25} D_{27}	A
C	E_{50} F_{60} C_{25} D_{27}	B A
H	G_{18} H_{10} I $_{18}$ E_{50} F_{60} D_{27}	C B A
K	J K G I E F D	H C B A
K	The Goal	Stop

Solution Path :A→ C→ H →K Found by using pred



10 5

Cs	Open	Close
A	A	
B	B C D	A
C	E F C D	B A
G	G E F D	C B A
I	H I E F D	H C B A
I	The Goal	Stop

Solution Path :A→ C→ G →I

Best –First search (8-puzzle example)

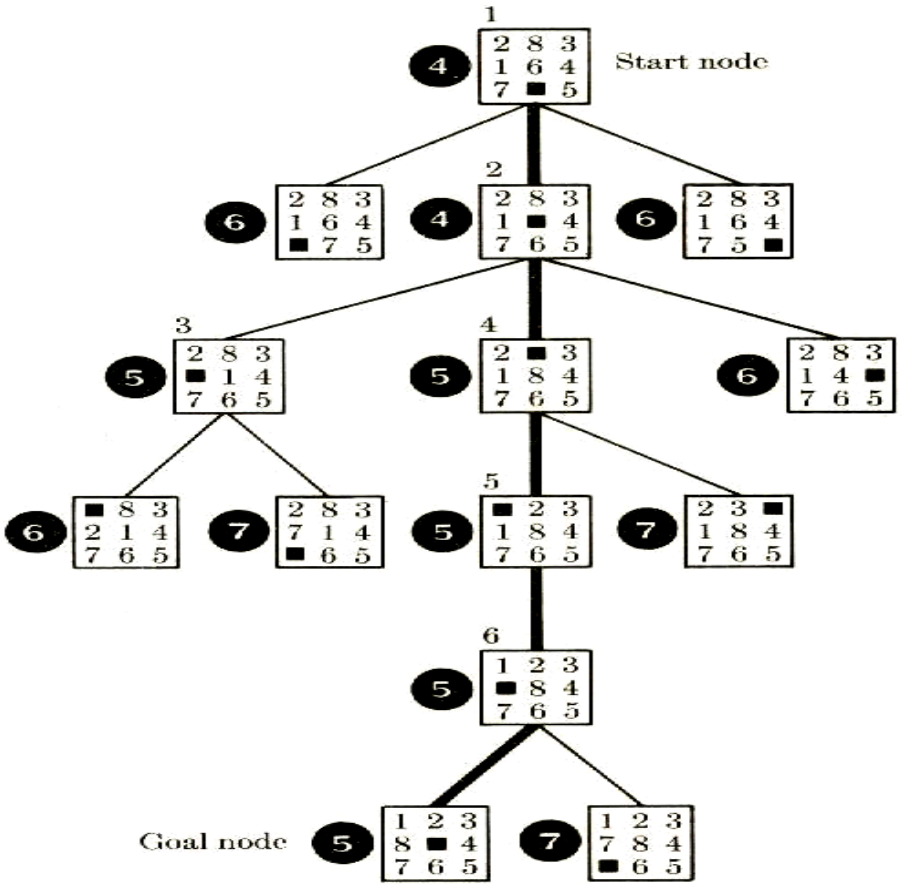
Evaluation Function: F(n)=g(n)+h'(n),

Where g(n)=depth of n. العمق

h'(n)= function counts the number of misplaced tiles in the state description associated with node n.

عدد الأحجار التي ليس في مواقعها

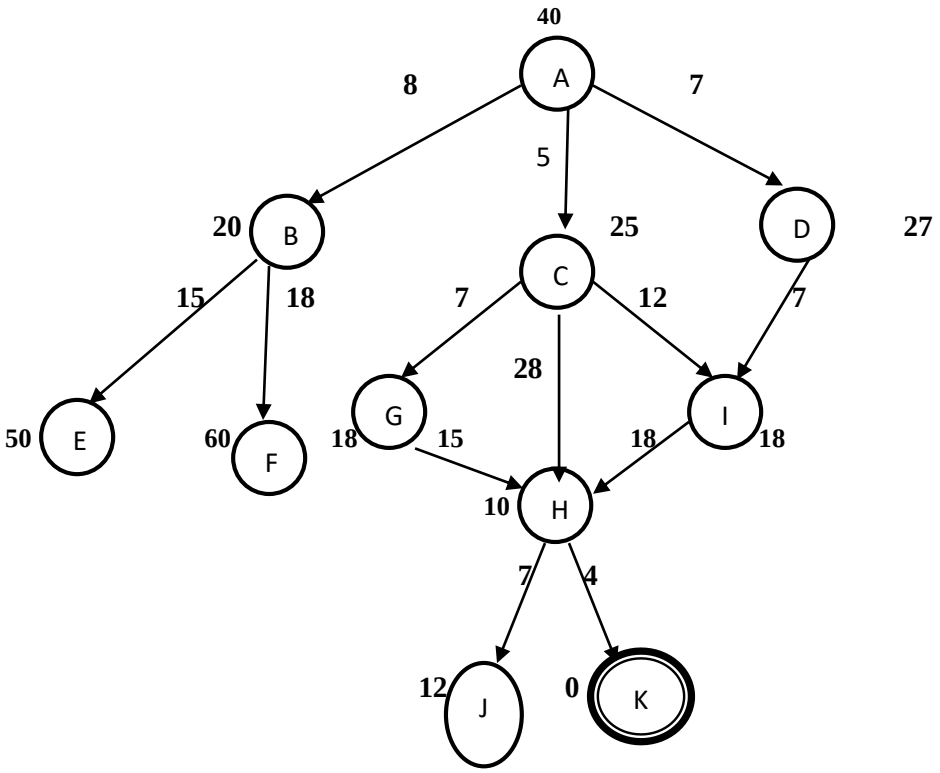
الكلفة(للنقطة) = العمق (المستوى الذي تقع فيه النقطة) + عدد الاحجار التي ليس في مواقعها


1
2 8 3
1 6 4
7 ■ 5
4
Start node
2
2 8 3
1 ■ 4
7 6 5
6
2 8 3
1 6 4
■ 7 5
4
2 8 3
1 6 4
7 5 ■
6
3
2 8 3
■ 1 4
7 6 5
5
4
2 ■ 3
1 8 4
7 6 5
5
2 8 3
1 4 ■
7 6 5
6
5
■ 8 3
2 1 4
7 6 5
6
2 8 3
7 1 4
■ 6 5
7
■ 2 3
1 8 4
7 6 5
5
2 3 ■
1 8 4
7 6 5
7
6
1 2 3
■ 8 4
7 6 5
5
Goal node
1 2 3
8 ■ 4
7 6 5
5
1 2 3
7 8 4
■ 6 5
7

A Star (A*) Algorithm خوارزمية (A*)

```
Begin
Input the start node S and the set G of goal nodes
Open [s]; closed=[ ] ;
G[s] = 0 ; pred[s] = Nil ; found=false;
F[s] = h(s);
While (open <> [ ] ) & (found= false) do
{
    L= the set of nodes on open for which F has the best;
    If (L is singleton) then let X be that element
    Else
    If there are any goal nodes in L
    Then let X be one  of them
    Else
        Let X be any element of L
        Remove X from open and put it into closed
        If ( X is a goal node ) then found = true
        Else
        Begin
            Generate the set successors of X for each Y in successors do
            If(Y is not already on open or closed) then
            Begin
                G[Y]=G[X]+cost(X,Y)
                F[Y]=G[Y]+h(Y)
                Pred[Y] = X
```

```
                    Insert Y in open;
                End
                Else /* Y is on open or closed*/
                Begin Z=pred[Y];
                    Temp =F[Y]-G[Z]-cost(Z,Y)+G[X]+cost(X,Y)
                    If (Temp < F[Y]) then
                Begin
                    G[Y]=G[X]+cost(X,Y)
                    F[Y]= Temp;
                    Pred[Y]= X;
                    if( Y in closed) then insert Y on open and      remove
                    Y from closed
                 end
            end
        end
    end;
}
If found=false then output " Goal not found"
Else trace the pointers in pred fields from X
Back to S to get the path from S to the Goal state
End.
```



STATE	ITER	OPEN	CLOSE	NODE	PRED
	0	A_{40}	[ ]		
A_{40}	1	$[B_{28},C_{30},D_{34}]$	$[A_{40}]$	A	0
B_{28}	2	$[C_{30},D_{34},E_{73},F_{86}]$	$[B_{28},A_{40}]$	B	A
C_{30}	3	$[D_{34},E_{73},F_{86},G_{30},H_{43},I_{35}]$	$[C_{30},B_{28},A_{40}]$	C	A
G_{30}	4	$[D_{34},E_{73},F_{86},H_{37},I_{35}]$	$[G_{30},C_{30},B_{28},A_{40}]$	G	C
				H	C G
D_{34}	5	$[E_{73},F_{86},H_{37},I_{32}]$	[D34,G30,C30,B28,A40]	D	A
				I	C D
I_{32}	6	$[E_{73},F_{86},H_{37}]$	$[I_{32}, D_{34},G_{30},C_{30},B_{28},A_{40}]$	I	D
H_{37}	7	$[E_{73},F_{86},J_{46},K_{31}]$	$[H_{37},I_{32},D_{34},G_{30},C_{30},B_{28},A_{40}]$	H	G
K_{31}	8	Goal found		K	H

Path : A→ C→ G →H→K

STATE	ITER	OPEN	CLOSE	NODE	PRED
	0	A_{40}	[ ]		
A_{40}	1	$[B_{28},C_{30},B_{34}]$ G(y)=G(y)+cost(x,y) F(A)=0+40=40 G(B)=G(A)+cost(A,B) G(B)=0+8=8 F(B)=8+20=28 G(C)=0+5=5 F(C)=5+25=25 G(D)=0+7=7 F(D)=7+27=34	$[A_{40}]$	A	0
B_{28}	2	$[C_{30},D_{34},E_{73},F_{86}]$ G(E)==8+15=23 F(E)=23+50=73 G(F)=8+18=26 F(F)=26+60=86	$[B_{28},A_{40}]$	B	A
C_{30}	3	$[D_{34},E_{73},F_{86},G_{30},H_{43},I_{35}]$ G(G)=5+7=12 F(G)=12+18=30 G(H)=5+28=33 F(H)=33+10=43 G(I)=5+12=17 F(I)=17+18=35	$[C_{30},B_{28},A_{40}]$	C	A
G_{30}	4	$[D_{34},E_{73},F_{86},H_{37},I_{35}]$ Z=pred(y) Temp=F(y)-G(Z)-cost(Z,Y)+G(X)+cost(X,Y) If (temp<F(Y))	$[G_{30},C_{30},B_{28},A_{40}]$	G H	C C G

		Z=pred(C) 43-5-28+12+15=37 If(37<43) then H37			
D_{34}	5	$[E_{73},F_{86},H_{37},I_{32}]$ Z=C Temp=35-5-12+7+7=32 If(35<32) then I32	[D34,G30,C30,B28,A40]	D I	A C D
I_{32}	6	$[E_{73},F_{86},H_{37}]$ Z=G Temp=37-12-15+14+18=42 If(42<37) then H37	$[I_{32}, D_{34},G_{30},C_{30},B_{28},A_{40}]$	I	D
H_{37}	7	$[E_{73},F_{86},J_{46},K_{31}]$ G(Y)=G(X)+COST(X,Y) 27+7=34 F(J)=34+12=46 G(K)=27+4=31 F(H)=31+0=31	$[H_{37},I_{32},D_{34},G_{30},C_{30},B_{28},A_{40}]$	H	G
K_{31}	8	Goal found		K	H

Properties of Heuristic Function:

1- **Admissibility: A heuristic function is admissible if it finds the shortest path to a goal state.**
   Note: All A* algorithms are admissible.

2- **Monotonicy: A heuristic function is monotonic if the first visit to any intermediate node gives the shortest path to that node.**
   Note: each Monotoncity function is admissible but not the complement because it choose the shortest path

3- **Informedness: A heuristic function is h1 is said to be more informed than a heuristic function h2 if h1(n)>=h2(n) for all nodes n. if h1 is**

more informed than h2 then the subsequence searched by h1 will be less than that searched by h2.

الدالة التي تجد اقصر طريق للهدف تعتبر ADMISSIBILITY.

***A هي** ADMISSIBILITY **أي تجد اقصر طريق للحل.**

الدالة التخمينية التي تجد اقصر طريق للهدف باول زيارة للـــ node أي تحصل على اقصر طريق بزيارة واحدة للـ node أي لا يحدث حذف داخل open node.

H1 تكون اكثر حكمة من h2 اذا كانت h1 تزور node اقل من h2 لذلك هي اكثر حكمة.

البحث الموجه Adversary Search

هو بحث مختص يســـتخدم فقط للبحث عن الطريقة للفوز بالعاب معينة (Games). وهذا البحث يكون على بناء شـجري يولد من خلال خطوات اللعبة يسـمى بشـجرة اللعبة (Game tree) حيث العقد فيها تمثل الموقف في اللعبة المعينة (Board configuration) والتفرعات لهذه الشــجرة تمثل النقلات المختلفة التي يقوم بها اللاعبين (Moves) وكذلك فان كل مستوى (Plies) من مستويات الشجرة يعود الى دور لاحد اللاعبين الذي يقوم باختيار خطواته القادمة من اجل الوصـــول لحالة الفوز، أي ان هذا البحث يتدخل في انجازه اللاعبين وليس يعتمد على الخوارزمية فقط كما هو الحال في طرق البحث السـابقة التي تم مناقشـتها. ولذلك يطلق عليه البحث الموجه.

محاضرات العملي:

Python Programming Language

What is Python?

Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming.

Who created Python?

One of the amazing features of Python is the fact that it is actually one person's work. Usually, new programming languages are developed and published by large companies employing lots of professionals, and due to copyright rules, it is very hard to name any of the people involved in the project. Python is an exception.

There are not many languages whose authors are known by name. Python was created by **Guido van Rossum**, born in 1956 in Haarlem, the Netherlands. Of course, Guido van Rossum did not develop and evolve all the Python components himself.

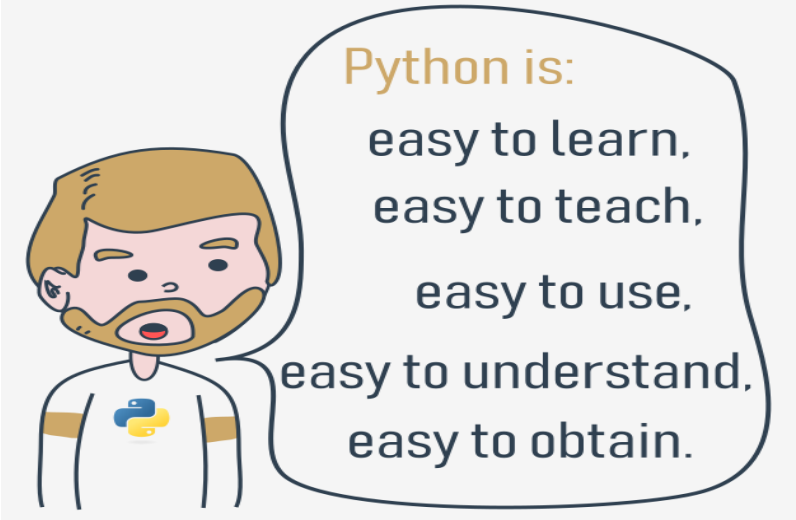
Python goals

In 1999, Guido van Rossum defined his goals for Python:

- an **easy and intuitive** language just as powerful as those of the major competitors;
- **open source**, so anyone can contribute to its development;
- code that is as **understandable** as plain English;
- **suitable for everyday tasks**, allowing for short development times.

About 20 years later, it is clear that all these intentions have been fulfilled. Some sources say that Python is the most popular programming language in the world, while others claim it's the third or the fifth.

Python isn't a young language. It is mature and trustworthy. It's not a one-hit wonder. It's a bright star in the programming firmament, and time spent learning Python is a very good investment.



What Can You Do with Python?

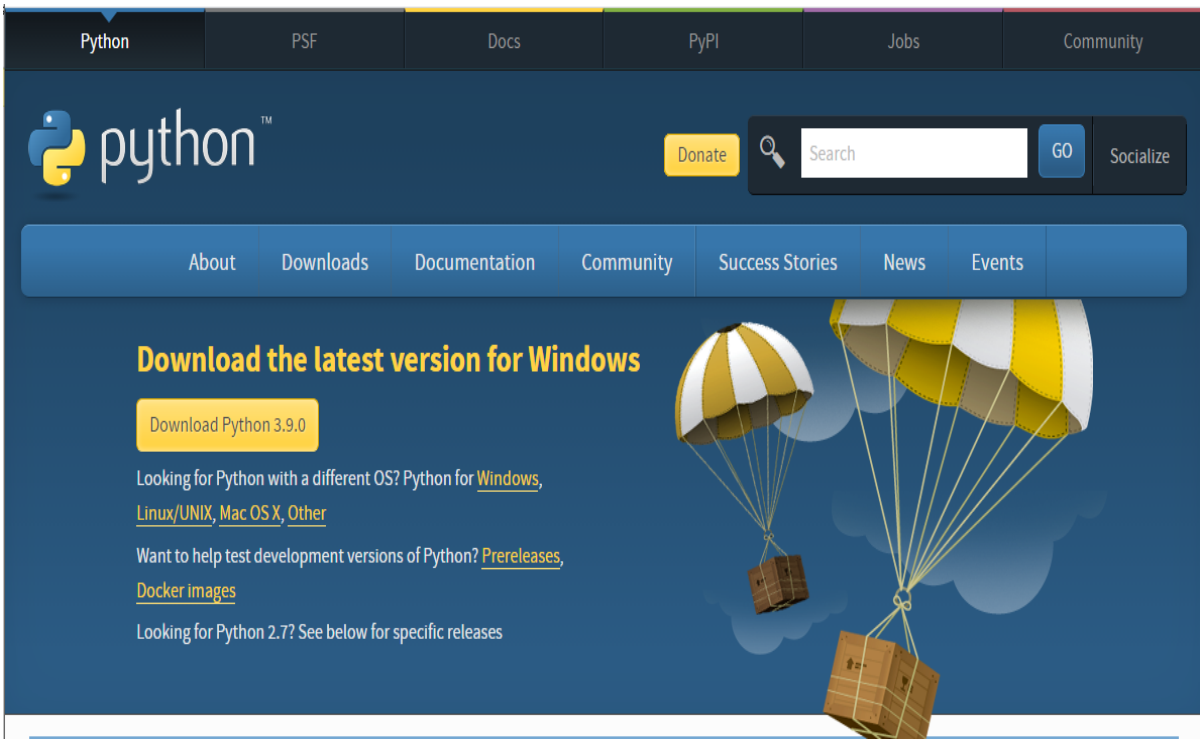
You may be wondering what all are the applications of Python. There are so many applications of Python, here are some of the them.

1. **Web development** – Web framework like Django and Flask are based on Python. They help you write server side code which helps you manage database, write backend programming logic, mapping urls etc.
2. **Machine learning** – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognises user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.
3. **Data Analysis** – Data analysis and data visualisation in form of charts can also be developed using Python.
4. **Scripting** – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.
5. **Game development** – You can develop games using Python.
7. **Desktop applications** – You can develop desktop application in Python using library like TKinter or QT.

How to get Python and how to get to use it

1- How to Install Python

Python installation is pretty simple, you can install it on any operating system such as Windows, Mac OS X, Ubuntu etc.
To install the Python on your operating system, go to this
link: https://www.python.org/downloads/. You will see a screen like this.



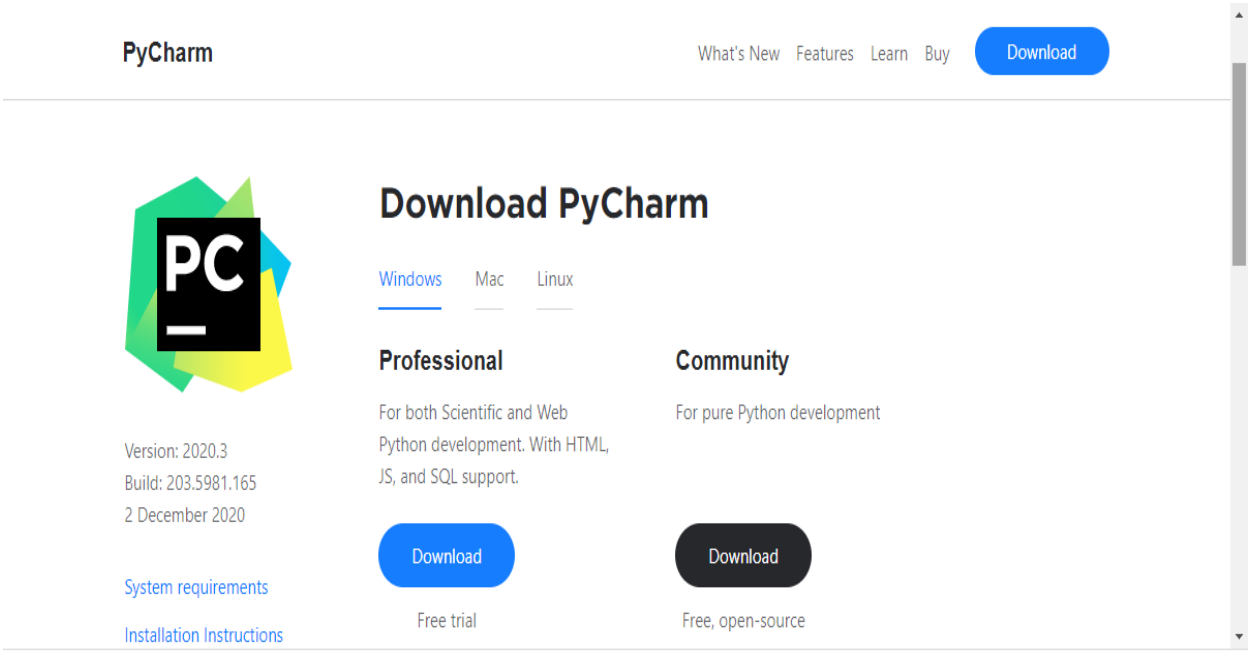
There are many ways of utilizing Python, especially if you're going to be a Python developer, such as IDEL and PyCharm.

Install PyCharm Python IDE

IDE stands for integrated development environment. It is a software that consolidates the basic tools that are required to write and test programs in a certain language. Typically, an IDE contains a code editor, a compiler or interpreter and a debugger that you can access at the same place through IDE GUI.

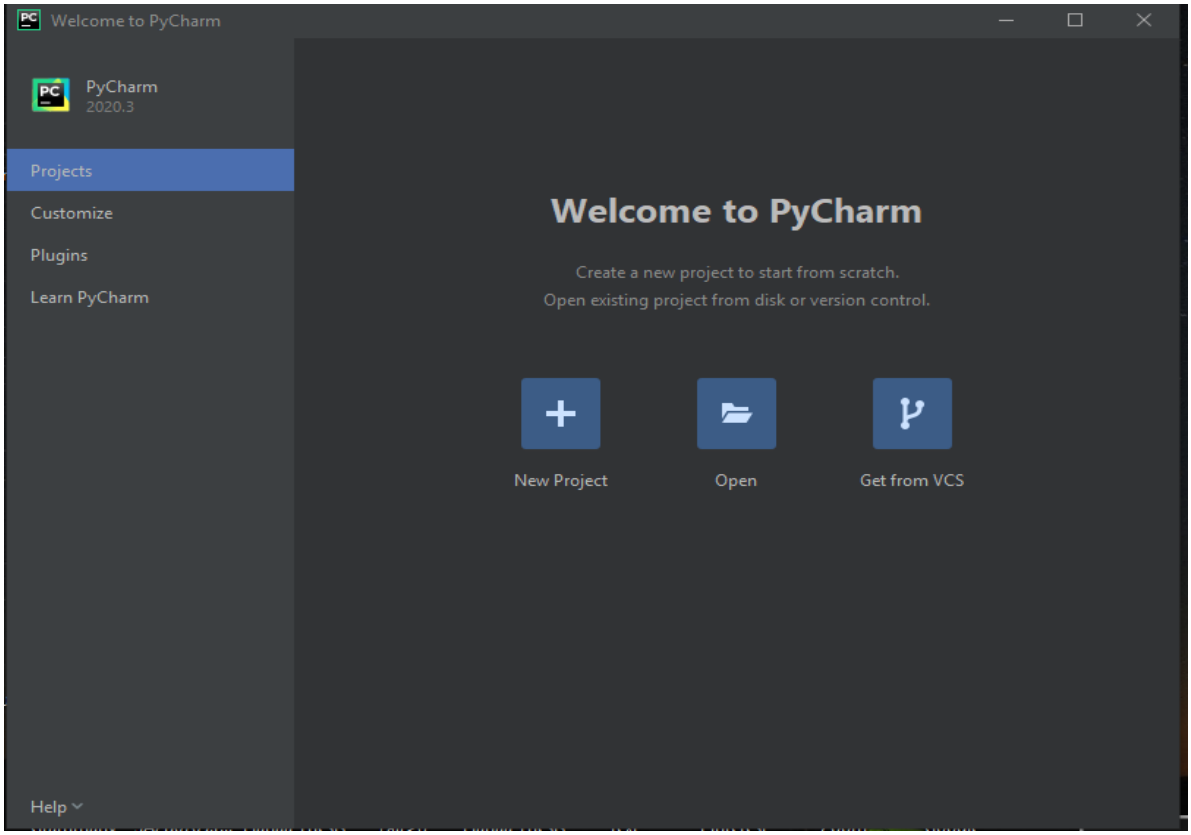
PyCharm Installation

Go to this link: https://www.jetbrains.com/pycharm/download/ and download the community edition. Then Double click the .exe file and follow the installation steps for the default PyCharm installation.

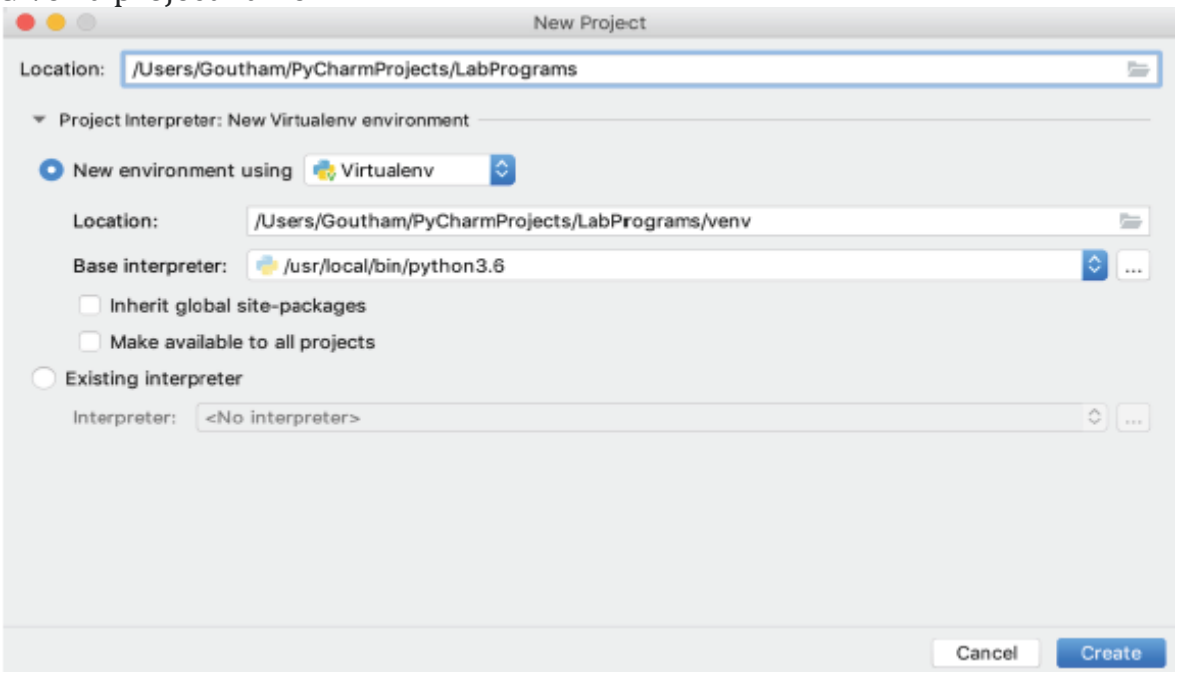


Creating Python Project in Pycharm IDE

1. Click “Create New Project” in the PyCharm welcome screen.

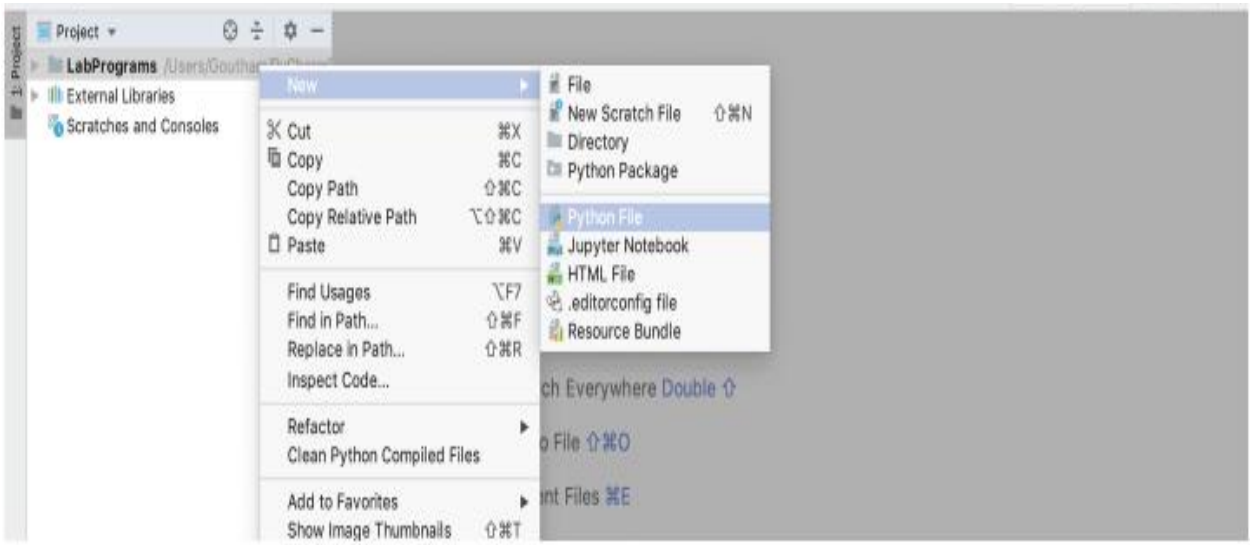


Given a project name



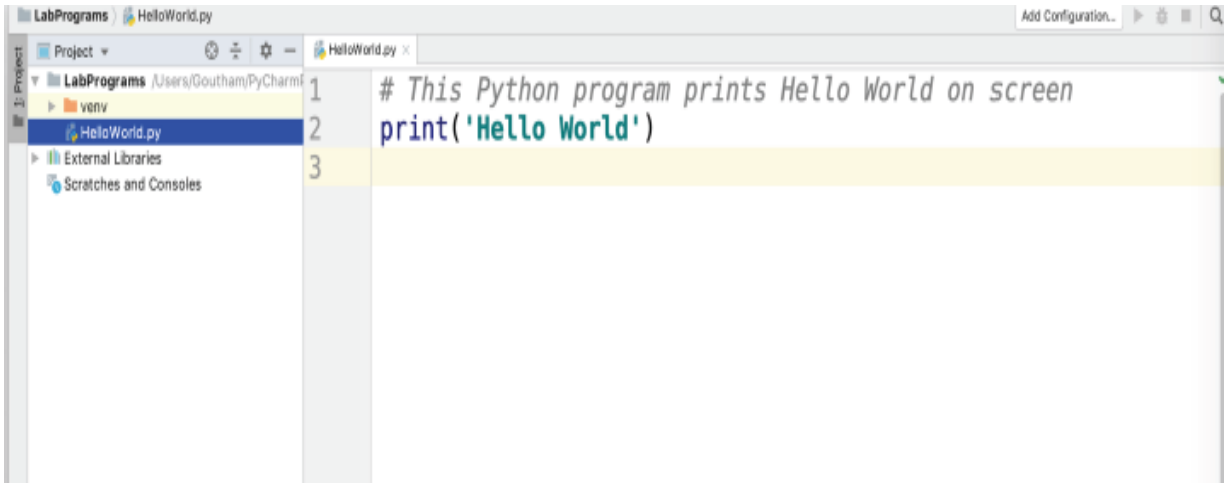
Writing and running your first Python Program

1. Now that we have created a Python project, it's time to create a Python program file to write and run our first Python program. To create a file, right click on the folder name > New > Python File (as shown in the screenshot). Give the file name as "HelloWorld" and click ok

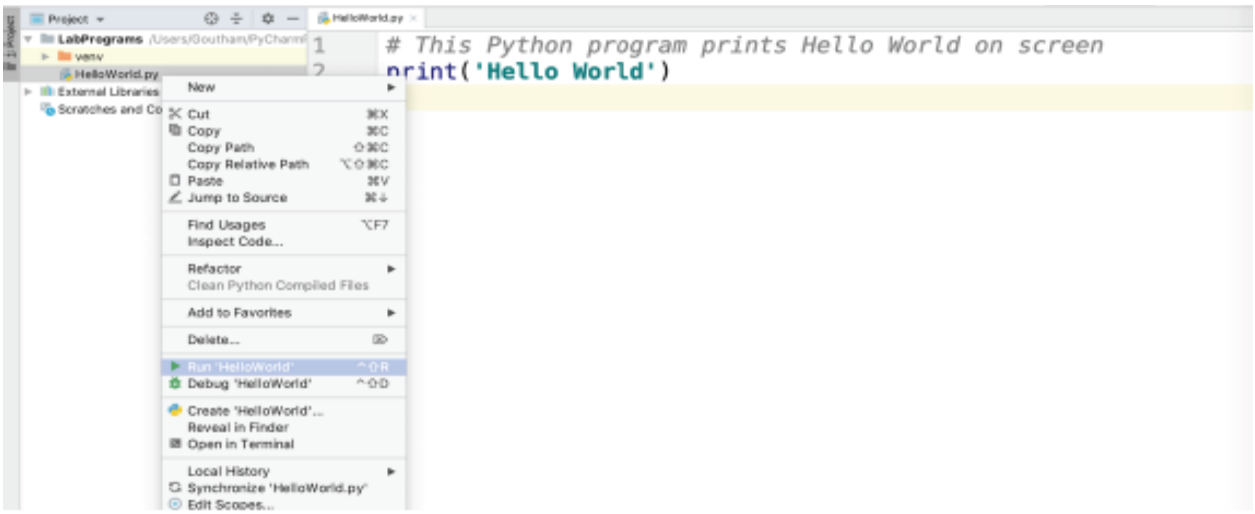


Write the following code in the file.

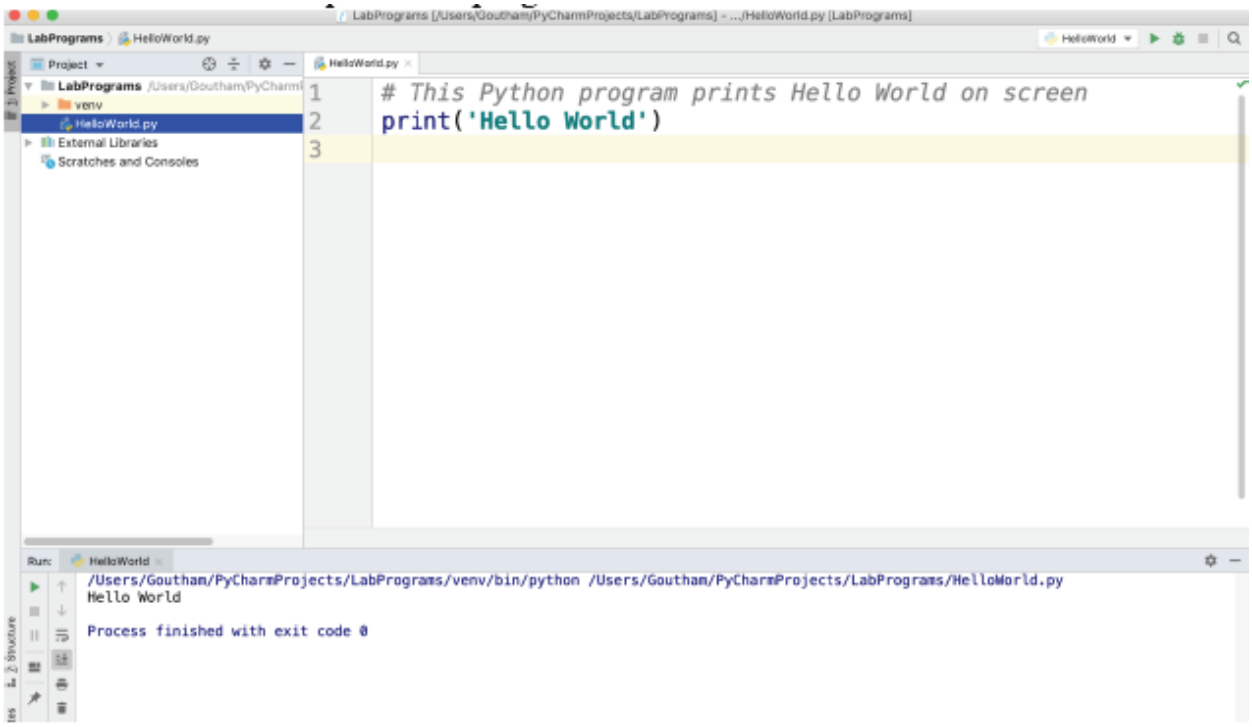
```
# This Python program prints Hello World on screen
    print('Hello World')
```



Let's run the code. Right click on the HelloWorld.py file (or the name you have given while creating Python file) in the left sidebar and click on 'Run HelloWorld'.



You can see the output of the program at the bottom of the screen.



Python Variable with Examples

Variables are used to store data, they take memory space based on the type of value we assigning to them. Creating variables in Python is simple, you just have to write the variable name on the left side of = and the value on the right side, as shown below. You do not have to explicitly mention the type of the variable, python infer the type based on the value we are assigning.

```
num = 100 #num is of type int
str = "Ahmed" #str is of type string
```

Variable name – Identifiers

Variable name is known as identifier. There are few rules that you have to follow while naming the variables in Python.

1. The name of the variable must always start with either a letter or an underscore (_). For example: _str, str, num, _num are all valid name for the variables.

2. The name of the variable cannot start with a number. For example: 9num is not a valid variable name.

3. The name of the variable cannot have special characters such as %, $, # etc, they can only have alphanumeric characters and underscore (A to Z, a to z, 0-9 or _).

4. Variable name is case sensitive in Python which means num and NUM are two different variables in python.

Python Variable Example

```
num = 100
str = "COMP-Lab"
print(num)
print(str)
```

Python multiple assignment

We can assign multiple variables in a single statement like this in Python.

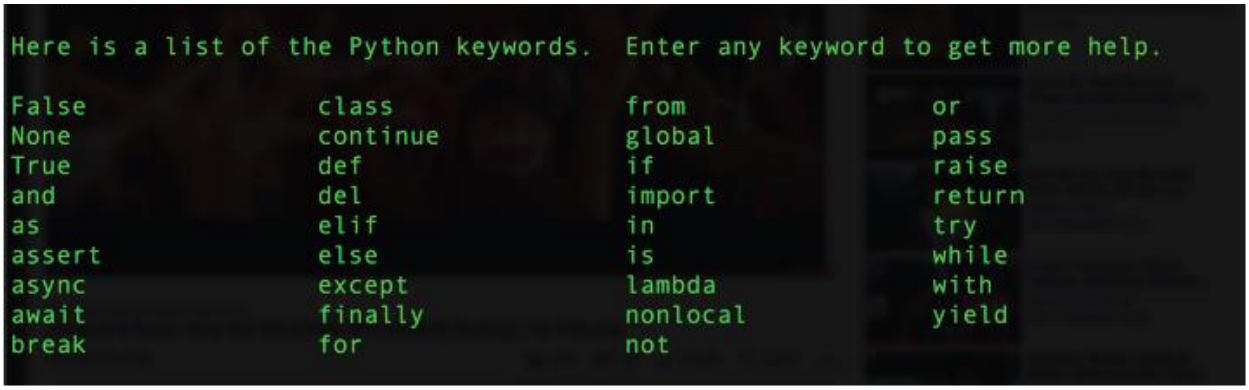
```
x = y = z = 99
print(x)
print(y)
print(z)
```

Another example of multiple assignment

```
a, b, c = 5, 6, 7
print(a)
print(b)
print(c)
```

Python Keywords

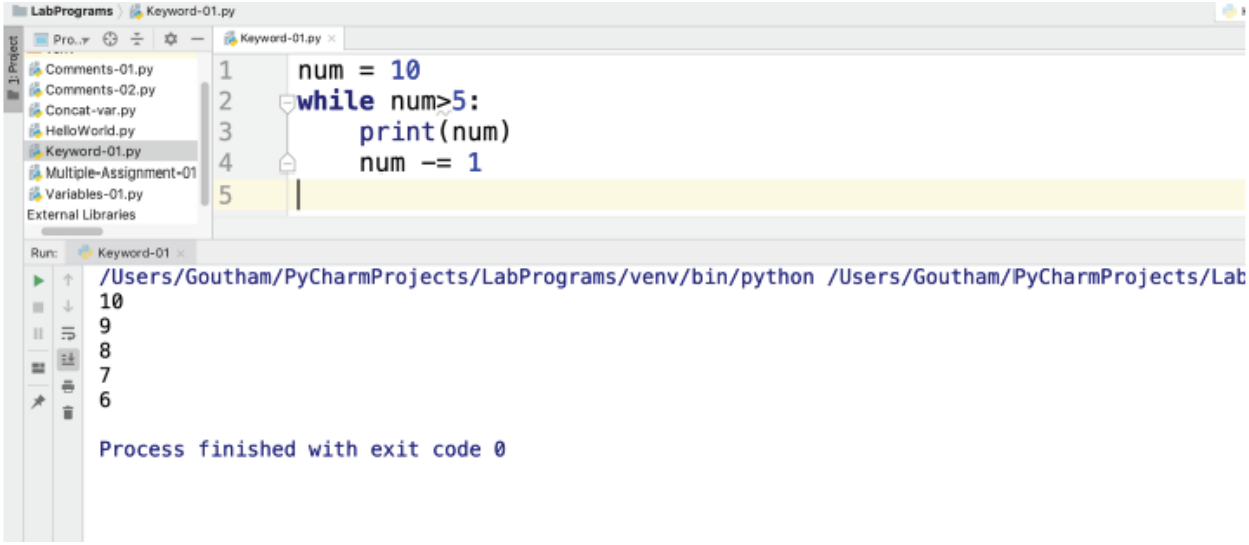
A python keyword is a reserved word which you can't use as a name of your variable, class, function etc. These keywords have a special meaning and they are used for special purposes in Python programming language. For example – Python keyword "while" is used for while loop thus you can't name a variable with the name "while" else it may cause compilation error. There are total 35 keywords in Python 3.6.



Example of Python keyword

In the following example we are using while keyword to create a loop for displaying the values
of variables num as long as they are greater than 5.

```
num = 10
while num>5:
print(num)
num -= 1
```



Data Types:

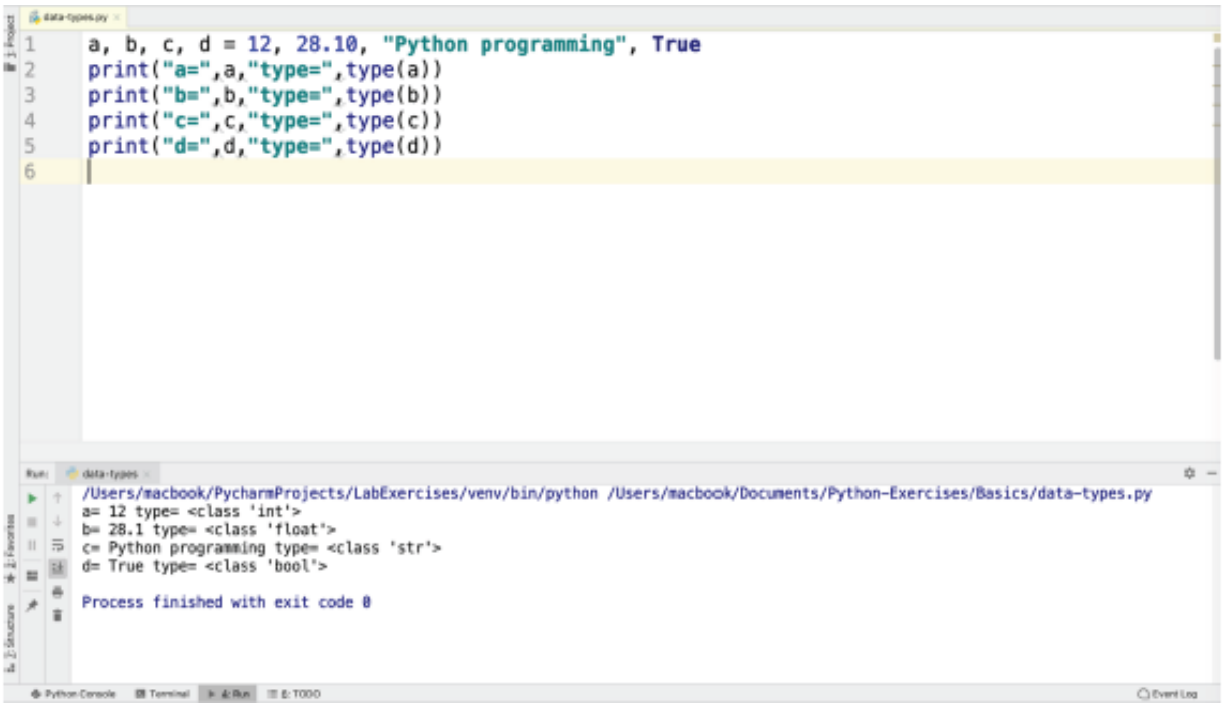
A data type defines the type of data, for example 123 is an integer data while “hello” is a String type of data.

Float: They are the numbers that have (or may have) a fractional part after the decimal point.

The decimal point is essentially important in recognizing floating-point numbers in Python. Look at these two numbers:
4
4.0
You may think that they are exactly the same, but Python sees them in a completely different way.
4 is an integer number, whereas 4.0 is a floating-point number. The point is what makes a float.

Write a Python Program to demonstrate different Data Types Solution:

```
a, b, c, d = 12, 28.10, "Python programming", True
print("a=",a,"type=",type(a))
print("b=",b,"type=",type(b))
print("c=",c,"type=",type(c))
print("d=",d,"type=",type(d))
```

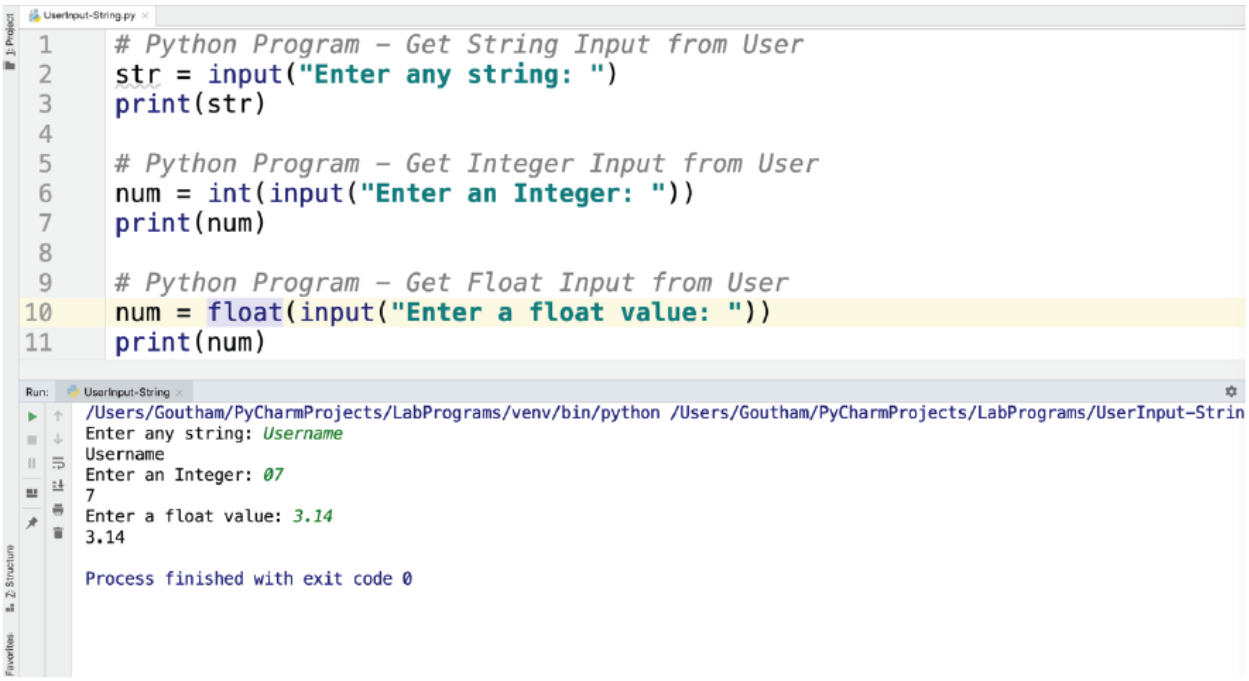


Write a Python Program to demonstrate the Getting Input from User Solution:

Get Different Datatypes input from user

In this program we will see how to receive String, Integer and Float input from user in Python.

```
# Python Program - Get String Input from User
str = input("Enter any string: ")
print(str)
# Python Program - Get Integer Input from User
num = int(input("Enter an Integer: "))
print(num)
# Python Program - Get Float Input from User
num = float(input("Enter a float value: "))
print(num)
```



```
# Python Program - Get String Input from User
str = input("Enter any string: ")
print(str)

# Python Program - Get Integer Input from User
num = int(input("Enter an Integer: "))
print(num)

# Python Program - Get Float Input from User
num = float(input("Enter a float value: "))
print(num)
```

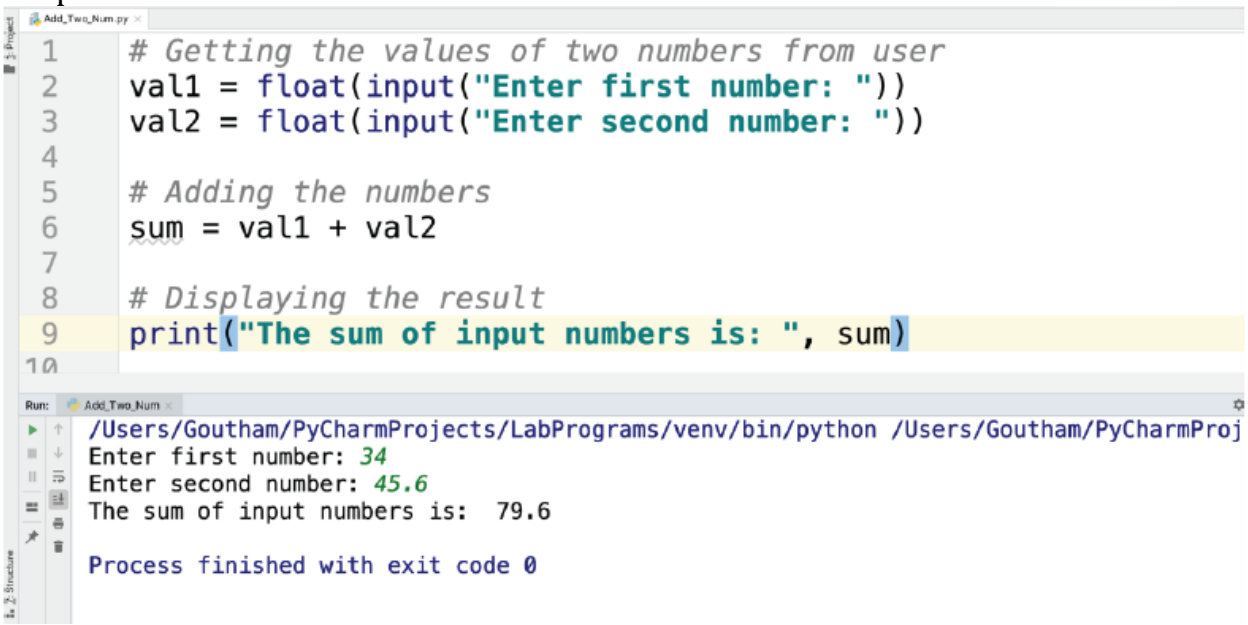
```
/Users/Goutham/PyCharmProjects/LabPrograms/venv/bin/python /Users/Goutham/PyCharmProjects/LabPrograms/UserInput-Strin
Enter any string: Username
Username
Enter an Integer: 07
7
Enter a float value: 3.14
3.14

Process finished with exit code 0
```

Write a Python Program to demonstrate the Addition of two Numbers Solution

```
# Getting the values of two numbers from user
val1 = float(input("Enter first number: "))
val2 = float(input("Enter second number: "))
# Adding the numbers
sum = val1 + val2
# Displaying the result
print("The sum of input numbers is: ", sum)
```

Output:



```
# Getting the values of two numbers from user
val1 = float(input("Enter first number: "))
val2 = float(input("Enter second number: "))

# Adding the numbers
sum = val1 + val2

# Displaying the result
print("The sum of input numbers is: ", sum)
```

```
/Users/Goutham/PyCharmProjects/LabPrograms/venv/bin/python /Users/Goutham/PyCharmProj
Enter first number: 34
Enter second number: 45.6
The sum of input numbers is:  79.6

Process finished with exit code 0
```

Write a Program to Demonstrate Arithmetic Operations using Python Solution:

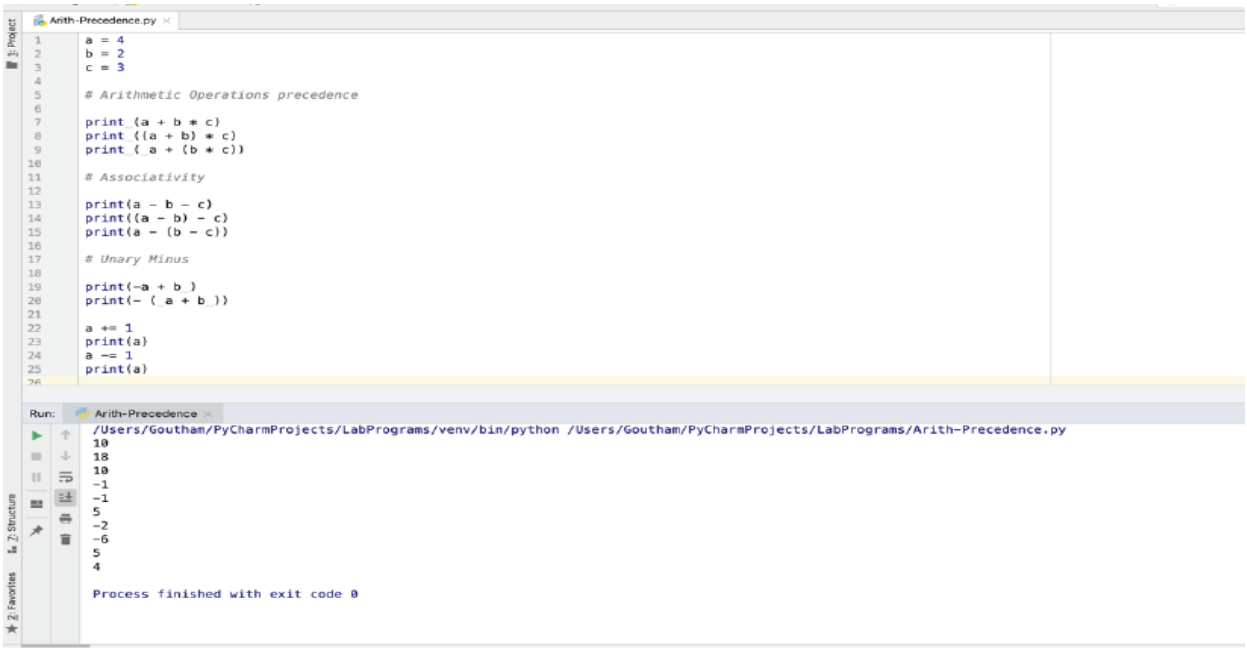
```
# Assignment Statement
a = 4
b = 2
# Arithmetic Operations
c = a + b
print(c)
c = a - b
print(c)
c = a / b
print(c)
c = a * b
print(c)
c = a % b
print(c)
c = a ** b
print(c)
c = - b
print(c)
```

Write a Program to Demonstrate Associativity and Precedence Operations using Python

Solution:

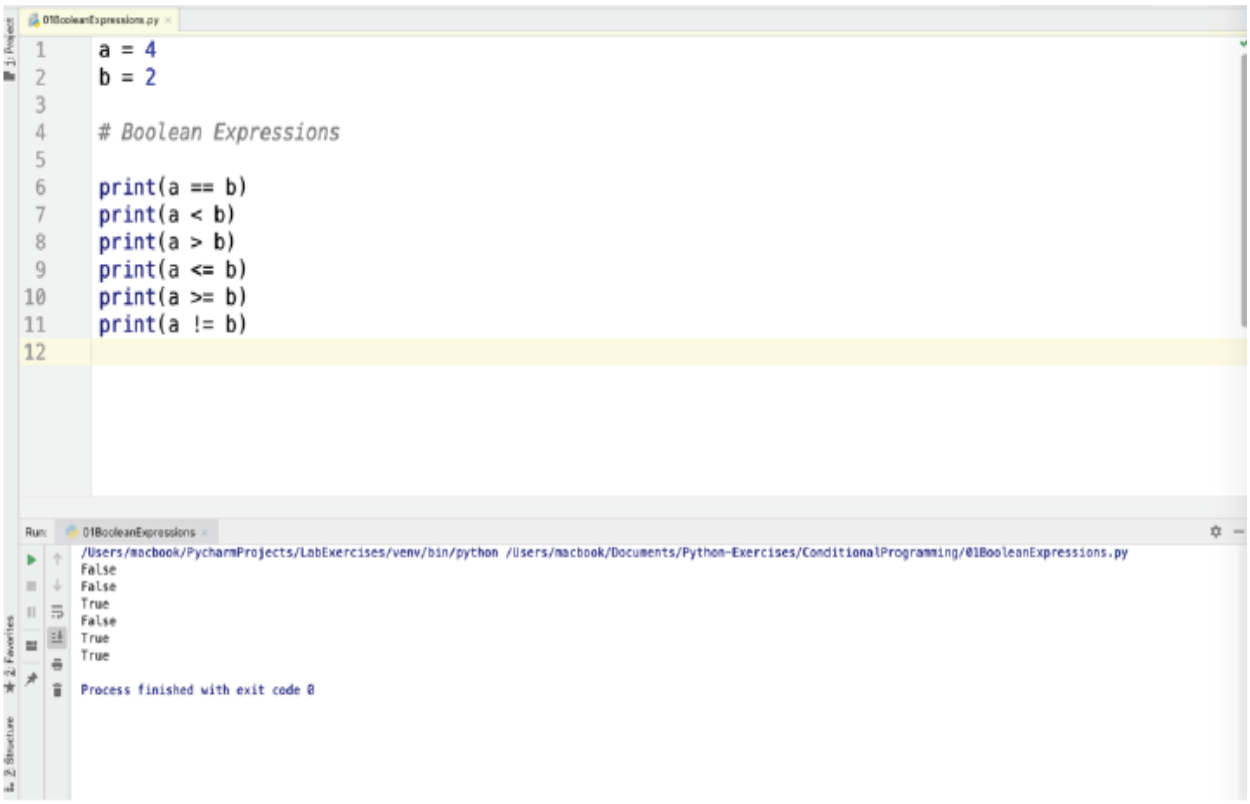
```
a = 4
b = 2
c = 3
# Arithmetic Operations precedence
print (a + b * c)
```

```
print ((a + b) * c)
print ( a + (b * c))
# Associativity
print(a - b - c)
print((a - b) - c)
print(a - (b - c))
# Unary Minus
print(-a + b )
print(- ( a + b ))
a += 1
print(a)
a -= 1
print(a)
```



Write a Program to Demonstrate Boolean Expressions using Python Solution:

```
a = 4
b = 2
# Boolean Expressions
print(a == b)
print(a < b)
print(a > b)
print(a <= b)
print(a >= b)
print(a != b)
```



Strings

Strings are used when you need to process text (like names of all kinds, addresses, novels, etc.), not numbers.
You already know a bit about them, e.g., that strings need quotes the way floats need points.
This is a very typical string: "I am a string."

However, there is a catch. The catch is how to encode a quote inside a string which is already delimited by quotes.
Let's assume that we want to print a very simple message saying:
I like "Python"
How do we do it without generating an error? There are two possible solutions.

The first is based on the concept we already know of the escape character, which you should remember is played by the backslash. The backslash can escape quotes too. A quote preceded by a backslash changes its meaning - it's not a delimiter, but just a quote. This will work as intended:

```
print("I like \" Python\"")
```

Lab(3) : Conditional Statements

If statements are control flow statements which helps us to run a particular code only when a certain condition is satisfied. For example, you want to print a message on the screen only when a condition is true then you can use if statement to accomplish this in programming.
In this Lab Session, we will learn how to use **if statements in Python programming** with the help of examples.

If Statement Syntax
The syntax of if statement in Python is pretty simple.

```
if condition:
block_of_code
```

Write a Program to Demonstrate the if-statement using Python

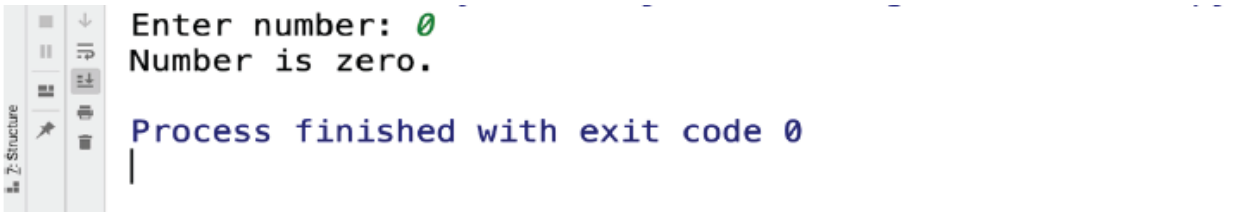
```
a = 1
b = 2
if a > b:
print("a is greater than b")
```

Write a Program to Demonstrate the if – else statement using Python.
Solution:

```
a = 3
b = 2
if a > b:
print("a is greater than b")
else:
print("b is greater than a")
```

Write a Program to demonstrate the if…elif…else statement using Python Solution. The user is asked to enter the number, the input number is stored in a variable number and then we have checked the number using if..elif..else statement.
Solution:

```
# User enters the number
number = int(input("Enter number: "))
# checking the number
if number < 0:
print("The entered number is negative.")
elif number > 0:
print("The entered number is positive.")
elif number == 0:
print("Number is zero.")
else:
print("The input is not a number")
```



Write a Python Program to check if Number is odd or Even.

Solution:

This program takes the input from user and checks whether the entered number is even or odd.

We are receiving the value into a variable num and then we are dividing the num by 2 to see whether it is an even number or odd number.

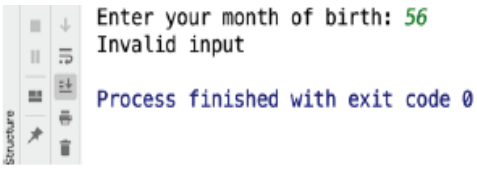
```
num = int(input("Enter any number: "))
flag = num%2
if flag == 0:
print(num, "is an even number")
elif flag == 1:
print(num, "is an odd number")
else:
print("Error, Invalid input")
```

Write a Python program that reads the month of birth from the user and prints the name for that month.

Solution:

```
month = int(input("Enter your month of birth: "))
if month == 1:
print ("January")
elif month == 2:
print("February")
elif month == 3:
print("March")
elif month == 4:
print("April")
elif month == 5:
print("May")
elif month == 6:
print("June")
elif month == 7:
```

```
print("July")
elif month == 8:
print("August")
elif month == 9:
print("September")
elif month == 10:
print("October")
elif month == 11:
print("November")
elif month == 12:
print("December")
else:
print("Invalid input")
```



Logical Operators

We use these operators to evaluate a statement to return either a **True** or a **False** The logical operators not, and, and or can be used to create a composite condition. Sometimes, a combination of several conditions determines whether a statement is executed. You can use logical operators to combine these conditions to form a compound expression.

Operator	Syntax	Description	Example
And	x and y	This returns **True** if both x and y are true	x = 10 x > 5 and x < 15
Or	x or y	This returns **True** if either x or y are true	x = 10 x > 5 or x < 2
Not	not x	Reverses a result, so if something is **True**, not turns it **False**	x = 10 not(x > 5 and x < 15)

```
#and
x = 10
print(x > 5 and x < 15)
#This returns True because 10 is greater than 5 AND 10 is less than 15.
```

```
#or
x = 10
print(x > 5 or x < 2)

#This returns True because one of the conditions are true.
#10 is greater than 5, but 10 is not less than 2.

#not
x = 10
print(not(x > 5 and x < 15))
#This returns False because not reverses the result of and.
```

Exercises

Use the operators to evaluate whether these statements are True or False .

Qn	Equation	True or False?
1	x = 200 print(x > 4 or x < 300)	**True** or **False**
2	3 != 4 and 3 < 4	**True** or **False**
3	x = 200 print(not(x > 4 or x < 300))	**True** or **False**

```
#Question 1
x = 200
print(x > 4 or x < 300)
#This returns True as one of it is true.

#Question 2
print(3 != 4 and 3 < 4)
#This returns True as both are true.

#Question 3
x = 200
print(not(x > 4 or x < 300))
#This returns False, as it reverses the result of question 1.
```

Lab 4: For Statements

The for statement in Python differs a bit from what you may be used to in C or Pascal. Rather than always iterating over an arithmetic progression of numbers (like in Pascal), or giving the user the ability to define both the iteration step and halting condition (as C), Python's for statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence. For example:

```
# Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

The range() Function

If you do need to iterate over a sequence of numbers, the built-in function range() comes in handy. It generates arithmetic progressions:

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

The given end point is never part of the generated sequence; `range(10)` generates 10 values, the legal indices for items of a sequence of length 10. It is possible to let the range start at another number, or to specify a different increment (even negative; sometimes this is called the 'step'):

```
range(5, 10)
   5, 6, 7, 8, 9

range(0, 10, 3)
   0, 3, 6, 9
```

```
range(-10, -100, -30)
  -10, -40, -70
```

To iterate over the indices of a sequence, you can combine range() and len() as follows:

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])
...
0 Mary
1 had
2 a
3 little
4 lamb
```

In most such cases, however, it is convenient to use the enumerate() function, see Looping Techniques.

A strange thing happens if you just print a range:

```
>>> print(range(10))
range(0, 10)
```

In many ways the object returned by range() behaves as if it is a list, but in fact it isn't. It is an object which returns the successive items of the desired sequence when you iterate over it, but it doesn't really make the list, thus saving space.

We say such an object is iterable, that is, suitable as a target for functions and constructs that expect something from which they can obtain successive items until the supply is exhausted. We have seen that the for statement is such a construct, while an example of a function that takes an iterable is sum():

```
>>> sum(range(4))  # 0 + 1 + 2 + 3
6
```

Lastly, maybe you are curious about how to get a list from a range. Here is the solution:

```
>>> list(range(4))
[0, 1, 2, 3]
```

For loop with else block - In Python we can have an optional 'else' block associated with the loop. The 'else' block executes only when the loop has completed all the iterations.

Example:

```
for val in range(5):
print(val)
else:
print("The loop has completed execution")
```

Output:

```
0
1
2
3
4
The loop has completed execution.
```

For loop with else and break: The following example illustrates the combination of an else, break statement with a **for** statement that searches for even number in given list.

```
numbers = [11,33,55,39,55,75,37,21,23,41,13]

for num in numbers:
  if num%2 == 0:
    print ('the list contains an even number')
    break
else:
  print ('the list doesnot contain even number')
```

Output

When the above code is executed, it produces the following result −

```
the list does not contain even nu
```

Nested For loop in Python - When a for loop is present inside another for loop then it is called a nested for loop. Lets take an example of nested for loop.

Example:

```
for num1 in range(3):
for num2 in range(10, 14):
print(num1, ",", num2)
```

Output:
0 , 10
0 , 11
0 , 12
0 , 13
1 , 10
1 , 11
1 , 12
1 , 13
2 , 10
2 , 11
2 , 12
2 , 13

Lab(5) Programs Exercise

طباعة العدد الأكبر بين عددين

```
a = 90
b = 40
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

اكتب برنامج يطلب من طالب إدخال علامته و بعده يطبع له نتيجته إذ كان ناجحاً أم راسباً.

```
mark=int(input("enter your mark:"))
if mark>=50:
    print("successful")
else:
    print("Unsuccessful")
```

أكتب برنامج يطلب من المستخدم رقم ما و ثم يتحقق من الرقم إذا كان موجب أو سالب.

```
number=int(input("enter number: "))
if number>=0:
    print("positive")
```

```
else:
    print("negative")
```

أكتب برنامج يطلب من المستخدم رقم ما و ثم يتحقق من الرقم إذا كان زوجي أو فردي.

```
number=int(input("enter number: "))
if number%2==0:
    print("even")
else:
    print("odd")
```

أكتب برنامج يقوم بطباعة جميع الأعداد الصحيحة بين ١ و ٢٠

```
for i in range(1, 21):
    print(i)
```

أكتب برنامج يقوم بطباعة جميع الأعداد الزوجية بين ١ و ٣٠

```
for i in range(1, 31):
    if i%2==0:
        print(i)
```

اكتب برنامج يقوم بطباعة جميع الأعداد الفردية بين ١ و ٤٠

```
for i in range(1, 41):
    if i%3==0:
        print(i)
```

أكتب برنامج يطبع احرف اللغة الأنكليزية كاملة بشكل صغير

```
for i in range(97,123):
    print(chr(i))
```

أكتب برنامج يطبع احرف اللغة الأنجليزية كاملة بشكل كبير

```
for i in range(65,91):
    print(chr(i))
```

انشئ مصفوفة ارقام ومن ثم اطبع عناصر المصفوفة

```
n=[18,6,9,40,60,125]
for index in range(0,6):
    print(n[index])
```

أكتب برنامج يمر على جميع احرف كلمة ما ويطبع كل حرف بسطر

```
for x in "class":
  print(x)
```

اكتب برنامج يخرج من الحلقة عندما يصل للكلمة class

```
admins = ["data", "class", "three", "kaso", "s49"]
for x in admins:
  print(x)
  if x == "class":
    break
```

أكتب برنامج يتجاهل كلمة class و يكمل الحلقة

```
admins = ["data", "class", "three", "kaso", "s49"]
for x in admins:
  if x == "class":
    continue
  print(x)
```

- What is the output of the following code in (a) and (b) if number is 30, then if number is 35.

```
number = int(input("Enter any Number: "))
if number % 2 == 0:
print(number, "is even.")
else:
print(number, "is odd.")
```

- Write a Python program that uses an if statement to find the smallest of three given integers. (The user should enter the three numbers).

- Write a Python program that reads username and password from the user and prints "successful login" if the username and password are correct, otherwise, it prints "username or password is wrong".

- Write a Python Program to display the Multiplication Table of 12 or based on user Input using for loop.

```
''' Python program to find the multiplication table (from 1 to 10)'''
num = 12
# To take input from the user
# num = int(input("Display multiplication table of? "))
# use for loop to iterate 10 times
for i in range(1, 11):
print(num,'x',i,'=',num*i)
```

Output

```
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
```

12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120

- **Write a Python Program to find the Factorial of a Number.**

Explanation:
The factorial of a number is the product of all the integers from 1 to that number.
For example, the factorial of 6 (denoted as 6!) is 1*2*3*4*5*6 = 720.
Factorial is not defined for negative numbers and the factorial of zero is one, 0! = 1.
Source Code:

```
# Python program to find the factorial of a number provided by
the user.
# change the value for a different result
num = 7
# uncomment to take input from the user
#num = int(input("Enter a number: "))
factorial = 1
# check if the number is negative, positive or zero
if num < 0:
print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
print("The factorial of 0 is 1")
else:
for i in range(1,num + 1):
factorial = factorial*i
print("The factorial of",num,"is",factorial)
```

Output:
The factorial of 7 is 5040

Write a Python Program to Print all Prime Numbers in an Interval
Explanation:
In this program, you'll learn to print all prime numbers within an interval using for loops and display it. To understand this example.

```
# Python program to display all the prime numbers within an interval
# change the values of lower and upper for a different result
```

```
lower = 900
upper = 1000
# uncomment the following lines to take input from the user
#lower = int(input("Enter lower range: "))
#upper = int(input("Enter upper range: "))
print("Prime numbers between",lower,"and",upper,"are:")
for num in range(lower,upper + 1):
            # prime numbers are greater than 1
        if num > 1:
            for i in range(2,num):
                if (num % i) == 0:
                    break
                else:
                            print(num)
```

Output:

Prime numbers between 900 and 1000 are:
907
911
919
929
937
941
947
953
967
971
977
983
991
997