**Functional programming**

البرمجة الوظيفية هي نموذج برمجة يتعامل مع الحساب باعتباره تقييمًا للوظائف الرياضية ويتجنب تغيير الحالة والبيانات القابلة للتغيير. في البرمجة الوظيفية، تعتبر الوظائف مواطنة من الدرجة الأولى، مما يعني أنه يمكن تعيينها للمتغيرات، وتمريرها كوسيطات إلى وظائف أخرى، وإعادتها كقيم من وظائف أخرى

**Procedural programming**

البرمجة الإجرائية هي نموذج برمجة يركز على استخدام الإجراءات أو الوظائف لأداء المهام وحل المشكلات. في البرمجة الإجرائية، يتم تقسيم البرنامج إلى إجراءات أو وظائف أصغر قابلة لإعادة الاستخدام، والتي يمكنها أخذ المدخلات ومعالجة البيانات وإنتاج المخرجات. عادةً ما يتم تنظيم هذه الوظائف بشكل تسلسلي ويمكن أن تتصل ببعضها البعض لإنجاز مهمة محددة. غالبًا ما يتناقض هذا النموذج مع البرمجة الشيئية والبرمجة الوظيفية.

فيما يلي بعض الخصائص والمفاهيم الأساسية المرتبطة بالبرمجة الإجرائية:

الإجراءات أو الوظائف: في البرمجة الإجرائية، يتم تنظيم البرنامج إلى إجراءات أو وظائف، وهي عبارة عن مجموعات مستقلة من التعليمات البرمجية تؤدي مهام محددة. يمكن استدعاء هذه الوظائف وإعادة استخدامها عدة مرات.

البيانات العالمية: غالبًا ما تستخدم البرامج الإجرائية المتغيرات العامة لتخزين وإدارة البيانات التي يمكن الوصول إليها من خلال وظائف متعددة. يمكن أن يؤدي هذا إلى مشكلات محتملة تتعلق بسلامة البيانات وقابلية صيانتها.

التنفيذ المتسلسل: يتم تنفيذ الإجراءات عادةً بترتيب تسلسلي، من الأعلى إلى الأسفل. يتم التحكم في تدفق البرنامج من خلال هياكل التحكم مثل الحلقات والعبارات الشرطية.

النمطية: تعمل البرمجة الإجرائية على تعزيز النمطية، مما يعني تقسيم المشكلة المعقدة إلى أجزاء أصغر يمكن التحكم فيها. يتم تنفيذ كل جزء كوظيفة منفصلة.

اللغات الإجرائية: بعض لغات البرمجة أكثر ملاءمة للبرمجة الإجرائية، مثل C، Pascal، وFortran. ومع ذلك، فإن العديد من لغات البرمجة الحديثة، بما في ذلك PHP وPython، تدعم البرمجة الإجرائية والموجهة للكائنات

**Object-Oriented Programming (OOP)**

البرمجة كائنية التوجه (OOP) هي نموذج برمجة يعتمد على مفهوم "الكائنات". إنه أسلوب قوي ومستخدم على نطاق واسع لتطوير البرمجيات، ويوفر طريقة لتنظيم التعليمات البرمجية وإدارة التعقيد وتعزيز إمكانية إعادة الاستخدام. في OOP، يتم تجميع البيانات والأساليب التي تعمل على تلك البيانات معًا في كائنات، والتي يمكن اعتبارها وحدات قائمة بذاتها وقابلة لإعادة الاستخدام. فيما يلي بعض المفاهيم الأساسية لـ OOP:

الفئات والكائنات: في OOP، الفصل عبارة عن مخطط أو قالب لإنشاء الكائنات. الكائنات هي مثيلات الطبقات. تحدد الفئة السمات (البيانات) والأساليب (الوظائف) التي ستحصل عليها الكائنات التي تم إنشاؤها منها.

التغليف: التغليف هو مفهوم تجميع البيانات (السمات) والأساليب (الوظائف) التي تعمل على تلك البيانات في وحدة واحدة تسمى الفئة. وهذا يوفر إمكانية إخفاء البيانات والتحكم في الوصول إليها، مما يسمح لك بحماية سلامة البيانات من خلال التحكم في كيفية الوصول إليها وتعديلها.

الوراثة: يسمح لك الوراثة بإنشاء فئة جديدة (الفرعية أو الفئة الفرعية) بناءً على فئة موجودة (الأصل أو الفئة الفائقة). ترث الفئة الفرعية سمات وأساليب الفئة الأصلية، ويمكنك أيضًا إضافة أو تجاوز الوظائف في الفئة الفرعية.

تعدد الأشكال: تعدد الأشكال يعني أنه يمكن التعامل مع الكائنات من فئات مختلفة ككائنات من فئة فائقة مشتركة. فهو يسمح لك بكتابة تعليمات برمجية يمكنها العمل مع كائنات من أنواع مختلفة، طالما أنها تلتزم بواجهة مشتركة.

التجريد: التجريد هو عملية تبسيط الأنظمة المعقدة عن طريق تقسيمها إلى أجزاء أصغر وأكثر قابلية للإدارة. في OOP، يتم استخدام الفئات لتجريد كيانات العالم الحقيقي، ونمذجتها بطريقة منطقية للمشكلة التي يتم حلها.

الكائنات: الكائنات هي مثيلات للفئات. إنها تمثل أمثلة محددة لمفهوم صاغه الفصل الدراسي، ويمكن أن يكون لها بياناتها وحالتها الفريدة