

كلية الذكاء الاصطناعي العالمية

سلسلة «بعد الثانوية» · المقالة 7 من 10

البرمجة والتفكير الحاسوبي: لغة العصر التي تتيح لك أن تصنع أدواتك

لماذا تتعلم أن تكتب برنامجًا حتى لو لن تصير مبرمجًا؟ التفكير الحاسوبي وما بعد عصر الإكمال الآلي للكود

البرمجة والتفكير الحاسوبي: لغة العصر التي تتيح لك أن تصنع أدواتك

الجواب باختصار

تعلّم البرمجة ليس لمن يريد أن يصير مبرمجًا فقط، بل لكل من يريد أن يحوّل مشكلة غامضة إلى خطوات واضحة تنفذها الآلة. وحتى مع وجود مساعدات تكتب الكود، تبقى قيمة من يفهمه: يحدد المطلوب، ويراجع، ويختبر، ويكتشف الخطأ. وبايثون بداية مناسبة لأنها مقروءة وتفتح الباب إلى الذكاء الاصطناعي.

سؤال يتكرر: هل بقي معنى لتعلم البرمجة؟

يسألك الناس اليوم: «الذكاء الاصطناعي يكتب الكود، فلماذا أتعلمه؟». وهو سؤال وجيه، وجوابه ليس «لأن الآلة لا تحسن الكتابة» فهي تحسنها في كثير من الحالات. جوابه أن **كتابة الأسطر لم تكن يومًا أصعب ما في البرمجة**.

الأصعب هو أن تعرف ما تريد بدقة، وتقسّمه إلى خطوات، وتتوقع الحالات الغريبة، وتختبر أن الناتج صحيح، وتكتشف السبب حين يفشل. وهذه كلها مهارات تفكير، وهي ما يبقى مطلوبًا مهما تغيرت الأدوات.

وتؤيد ذلك قوائم المهارات الصاعدة. فتقرير «مستقبل الوظائف 2025» للمنتدى الاقتصادي العالمي يضع في مقدمة المهارات التقنية الأسرع نموًا: الذكاء الاصطناعي، وتحليل البيانات الضخمة، والشبكات، والأمن السيبراني. وكلها تقوم على أساس واحد هو فهم كيف تعالج الآلة المعلومات. ومن يملك هذا الأساس يجد الدخول إلى أي منها أسهل.

وفي تقرير جامعة ستانفورد لعام 2025 أن 78% من المؤسسات المشاركة في المسح تستعمل الذكاء الاصطناعي في 2024 مقابل 55% في 2023. وحيث ينتشر الذكاء الاصطناعي في المؤسسات يحتاج الأمر إلى من يربطه بالأنظمة القائمة ويضبطه ويختبره. وهذه أعمال برمجية بامتياز.

ما هو التفكير الحاسوبي؟

الخلاصة: هو طريقة لحل المشكلات بتحليلها إلى أجزاء، وكشف الأنماط، وتجريد المهم، وصياغة خطوات واضحة.

كتبت الباحثة جانيت وينغ عام 2006 في مجلة «اتصالات جمعية الحاسوب» مقالة بعنوان «التفكير الحاسوبي»، دعت فيها إلى أن يكون هذا النمط من التفكير مهارة أساسية لكل إنسان إلى جانب القراءة والكتابة والحساب، لا حكرًا على المتخصصين. وهو يقوم على أربع أدوات ذهنية:

1. **التحليل**: تقسيم المشكلة الكبيرة إلى مشكلات صغيرة.
2. **كشف الأنماط**: ملاحظة ما يتكرر بين الحالات لتحلها مرة واحدة.
3. **التجريد**: إهمال التفاصيل التي لا تهم والتركيز على الجوهر.
4. **الخوارزمية**: صياغة حل بخطوات مرتبة لا لبس فيها.

ولا تحتاج إلى حاسوب لتمارس هذا. فلنأخذ مثالًا: تريد أن تنظم رحلة عائلية. تحلل: السكن والنقل والطعام والأنشطة. وترصد النمط: كل رحلة سابقة احتجنا فيها إلى قائمة تجهيز. وتجرد: لا تهمنا تفاصيل المطاعم الآن بل الميزانية. وتكتب خوارزمية: أولًا حدد المدة، ثم حدد الميزانية، ثم احجز كذا. وهذا نفسه ما يفعله المبرمج حين يحل مسألة.

ولهذا نقول إن البرمجة تدرب العقل، حتى لمن لن يكتب كودًا في عمله. فالطبيب الذي يفهم الخوارزميات يتحقق من أنظمة التشخيص الذكية، والمحاسب الذي يكتب برنامجًا بسيطًا يوفر أيامًا من العمل اليدوي، والمعلم الذي يبرمج يصمم أدواته التعليمية.

كيف يغيّر الذكاء الاصطناعي مهنة البرمجة؟

الصورة الصحيحة ليست «اختفاء المبرمج» ولا «لا تغير شيء». الذي يتغير هو توزيع الجهد.

ما تخففه الأدوات: كتابة الأكواد الروتينية، وتذكّر صيغ اللغة، وإنشاء هياكل المشاريع، وكتابة اختبارات بسيطة، وشرح كود غير مألوف.

ما يبقى للإنسان:

- **تحديد المطلوب**: ما المشكلة؟ ولمن؟ وما معيار النجاح؟
- **التصميم**: كيف تتوزع أجزاء النظام؟ وما الذي ينبغي أن يتغير بسهولة؟

- **المراجعة**: هل الكود يفعل ما أريد فعلًا؟ هل فيه ثغرة أمنية؟ هل يتعامل مع الحالات الشاذة؟
- **الاختبار**: كيف أثبت أنه صحيح؟
- **تحمل المسؤولية**: حين يفشل في الإنتاج، من يصلحه؟

والمفارقة أن من لا يفهم البرمجة يجد صعوبة بالغة في استعمال مساعد البرمجة. فهو لا يعرف كيف يصف المطلوب، ولا يستطيع أن يكتشف أن الكود المولَّد فيه خطأ دقيق، ولا يعرف ماذا يفعل حين يعطيه المساعد حلًّا لا يعمل. أما من فهم الأساس فيستعمل المساعد كما يستعمل الحرفي الماهر أداة كهربائية: تسرّع عمله ولا تغني عن خبرته.

> ما نسميه «البرمجة بمساعدة الذكاء الاصطناعي» أقرب إلى الإشراف على موهوب سريع قليل الخبرة: يعطيك عملًا كثيرًا بسرعة، وعليك أن تراجع كل ما يعطيك.

لماذا بايثون؟

يسأل المبتدئون: «أي لغة أتعلم؟». والجواب عن هدفنا هنا: **بايثون**. والأسباب:

- **مقروءة**: قريبة من اللغة الإنجليزية البسيطة، فتفهم ما يفعله الكود بسهولة.
- **واسعة الاستعمال في الذكاء الاصطناعي**: أغلب أدوات تعلم الآلة والبيانات تُستعمل عبرها.
- **مجتمع كبير**: إن واجهتك مشكلة فغالبًا سبقك إليها غيرك ووجدت لها حلًا.
- **سريعة التجربة**: تكتب سطرًا فترى الناتج فورًا، وهذا يشجع المبتدئ.
- **تنفع خارج الذكاء الاصطناعي**: في أتمتة المهام وتحليل الجداول وبناء مواقع بسيطة.

ولا نقول إنها أفضل لغة في كل شيء، فلكل لغة ميدانها. لكنها نقطة انطلاق مناسبة إلى الذكاء الاصطناعي، وما تتعلمه فيها من المفاهيم ينتقل إلى غيرها.

كيف تتعلم البرمجة تعلمًا صحيحًا؟

أكثر المبتدئين يتعثرون لا لأن البرمجة صعبة، بل لأنهم يتعلمونها بطريقة خاطئة. وهذه ست قواعد تجنبك ذلك:

أولًا: اكتب الكود بنفسك

مشاهدة فيديو يشرح برنامجًا لا تعلمك البرمجة، كما أن مشاهدة طبّاخ لا تعلمك الطبخ. اكتب كل سطر بيدك، وغيّر فيه، واكسره عمدًا لترى ماذا يحدث.

ثانيًا: ابدأ بمشكلات صغيرة من حياتك

لا تبدأ بتطبيق ضخم. ابدأ بحاسبة مصروف، أو برنامج يرتب أسماء، أو أداة تحسب درجاتك. المشروع الذي يهمك أنت تذاكره بشوق.

ثالثًا: تعلّم قراءة الأخطاء

رسالة الخطأ ليست عدوًا، بل هي أصدق دليل. تقول لك في أي سطر وقعت المشكلة وما نوعها. ومعظم المبتدئين يقفزون عنها ويسألون «لماذا لا يعمل؟». اقرأ الرسالة، فإن لم تفهمها فانسخها وابحث عنها أو اشرحها بمساعد ذكي ثم تحقق.

رابعًا: قسّم المسألة

قبل أن تكتب، اكتب بالعربية الخطوات التي تريدها. ثم حوّل كل خطوة إلى كود صغير، وجربه وحده قبل أن تجمع الأجزاء.

خامسًا: اختبر ما كتبت

لا تفترض أن برنامجك صحيح لأنه عمل مرة. جرّب حالات غريبة: قائمة فارغة، ورقم سالب، ونص طويل جدًا. وتعلّم لاحقًا كيف تكتب اختبارات آلية.

سادسًا: استعن بالمساعد الذكي كمعلم لا كناسخ

إن علقت فاطلب منه أن يشرح الفكرة ويعطيك تلميحًا لا الحل الكامل. وبعد أن تحل اطلب منه أن ينتقد حلك. فإن أعطاك كودًا جاهزًا فاقرأه سطرًا سطرًا حتى تفهم كل جزء، ثم أعد كتابته من ذاكرتك. وإن نسخته دون فهم فستقف عاجزًا

عند أول عطل.

خريطة بسيطة لأول تسعين يومًا في البرمجة

هذه خطة تقريبية لمبتدئ يخصص ساعة يوميًا. تتبعها كما هي أو عدّلها.

الأسابيع 1–3: الأساس. المتغيرات والأنواع (أرقام ونصوص)، والشروط (إذا... فإن...)، والحلقات (كرر...). وهدفك مشروعان صغيران: لعبة تخمين رقم، ومحوّل وحدات.

الأسابيع 4–6: تنظيم الكود. الدوال، والقوائم والقواميس، وقراءة الملفات وكتابتها، ومعالجة الأخطاء. وهدفك: برنامج يقرأ ملفًا فيه مصروفاتك ويحسب مجموعها حسب النوع.

الأسابيع 7–9: التعامل مع البيانات. مكتبات الجداول، وتنظيف البيانات، ورسم مخطط بسيط. وهدفك: تحليل بيانات صغيرة تهمك، كنتائج مباريات أو أسعار أو درجات.

الأسابيع 10–13: مشروع الختام. اختر مشكلة حقيقية من حولك وابنِ حلًّا مبسطًا. قد يكون برنامجًا يلخص تقارير، أو أداة تصنف رسائل. وفي هذه المرحلة تجرب استدعاء نموذج لغوي عبر واجهة برمجية لتضيف إلى مشروعك ذكاءً.

وفي نهاية التسعين يومًا ستملك ثلاثة أمور: مشاريع صغيرة تعرضها، وعادة كتابة وتجريب، وفهمًا يسمح لك أن تتعلم الأدوات الأخرى.

ثلاثة مشاريع أولى بالتفصيل

الخلاصة: المشروع الصغير الكامل أنفع من عشرة دروس متفرقة.

نفصّل لك ثلاثة مشاريع تصلح للبدء. لا نعطيك الكود جاهزًا عن قصد، لأن الغاية أن تكتبه أنت. نعطيك التفكير الذي يسبقه.

المشروع الأول: حاسبة المصروف الشهري

الفكرة: برنامج تدخل فيه مصروفاتك واحدًا واحدًا (الاسم والمبلغ والنوع)، ثم يعطيك المجموع الكلي والمجموع لكل نوع.

التفكير قبل الكتابة:

- ما المعلومات التي سأخزنها لكل مصروف؟ (اسم، ومبلغ، ونوع)
- كيف أحتفظ بالمصروفات؟ في قائمة أو ملف؟
- كيف أجمع حسب النوع؟ أحتاج إلى «قاموس» يربط كل نوع بمجموعه.
- ماذا لو أدخل المستخدم نصًا بدل الرقم؟ لا بد أن أتعامل مع الخطأ.

ما ستتعلمه: المتغيرات، والقوائم، والقواميس، والحلقات، والتعامل مع إدخال خاطئ.

تحسينات تالية: احفظ البيانات في ملف حتى لا تضيع، وأضف رسمًا بسيطًا يوضح النسب.

المشروع الثاني: منظم الملفات

الفكرة: برنامج يقرأ مجلدًا مليئًا بالملفات المبعثرة وينقل كل ملف إلى مجلد فرعي بحسب نوعه: صور، ومستندات، وجداول.

التفكير قبل الكتابة:

- كيف أعرف نوع الملف؟ من امتداد اسمه.
- كيف أتعامل مع ملف بامتداد غير معروف؟ أضعه في مجلد «أخرى».
- ماذا لو وُجد ملف بالاسم نفسه في المجلد الهدف؟ لا أريد أن أستبدله فأفقد البيانات.
- كيف أجرب البرنامج دون أن أخسر ملفاتي الحقيقية؟ أجربه أولًا على نسخة.

ما ستتعلمه: التعامل مع الملفات والمجلدات، والدوال، والشروط، وأهمية الحذر قبل تغيير ما لا يمكن استرجاعه.

المشروع الثالث: ملخّص تقارير بمساعدة نموذج لغوي

الفكرة: برنامج يقرأ ملف نص، ويطلب من نموذج لغوي عبر واجهة برمجية أن يلخصه في خمس نقاط، ثم يحفظ الملخص.

التفكير قبل الكتابة:

- كيف أقسّم النص الطويل إلى أجزاء يتحملها النموذج؟

- ما الأمر الذي سأرسله؟ وكيف أطلب منه ألا يختلق؟
- كيف أتحقق من الملخص؟ أطلب أن يذكر موضع كل نقطة في النص.
- ماذا أفعل إن فشل الطلب أو تأخر الرد؟
- ما كلفة كل طلب؟ وكيف أمنع استعمالًا مفرطًا؟
- هل في النص بيانات شخصية لا ينبغي إرسالها؟

ما ستتعلمه: الواجهات البرمجية، وهندسة الأوامر، والتعامل مع الفشل، وحساب الكلفة، والخصوصية. وهي مهارات هذا العصر مجتمعة في مشروع واحد.

ولاحظ أن كل مشروع يتضمن أسئلة عن **الخطأ والاختبار والأمان** قبل أن يبدأ الكود. هذا هو الفرق بين من يكتب كودًا ومن يبني برنامجًا.

منهج تصحيح الأخطاء: كيف تفكر حين لا يعمل البرنامج؟

الخلاصة: تصحيح الخطأ تحقيق منظم، وليس تخمينًا عشوائيًا.

سيحدث أن يتوقف برنامجك أو يعطي نتيجة خاطئة. وهنا يظهر الفرق بين المبتدئ والمحترف: المبتدئ يغيّر أشياء عشوائية على أمل أن يعمل، والمحترف يتحقق منهجيًا. وهذه خطواته:

1. **أعد إنتاج الخطأ.** حدد بدقة ما الذي تفعله فيظهر الخطأ. فإن لم يظهر دائمًا فسجّل متى يظهر.
2. **اقرأ الرسالة.** ما نوع الخطأ؟ في أي سطر؟ وما القيمة التي سببت المشكلة؟
3. **كوّن فرضية.** قل لنفسك: «أظن أن المتغير فلان فارغ هنا لأن الدالة لم تُستدع».
4. **اختبر الفرضية.** اطبع قيمة المتغير قبل السطر المشكوك فيه، أو استعمل أداة تتبع.
5. **غيّر شيئًا واحدًا في كل مرة.** وإلا لن تعرف أي تغيير أصلح.
6. **اشرحها لغيرك.** كثيرًا ما تجد الحل وأنت تشرح المشكلة. ويسمي المبرمجون هذا «شرح المشكلة لبطة مطاطية».
7. **ثبّت الدرس.** بعد أن تحل المشكلة اكتب ما كانت وكيف عرفتها، وأضف اختبارًا يمنع عودتها.

وهذا التفكير نفسه ينفعك خارج البرمجة: في الطب والإدارة والصيانة والتخطيط. أنت دائمًا تعيد إنتاج المشكلة، وتفحص الفرضيات، وتغيّر عاملًا واحدًا في كل مرة.

الكود المولَّد: كيف تراجعه وتأمنه؟

حين يعطيك مساعد ذكي كودًا، لا تثق به لأنه «يبدو سليمًا». راجعه بهذه الأسئلة:

- **هل أفهم كل سطر؟** ما لا أفهمه لا أستعمله.
- **هل يفعل المطلوب فعلًا؟** جرّبه على أمثلة أعرف نتيجتها.
- **كيف يتعامل مع الحالات الشاذة؟** فارغ وسالب وكبير جدًا ومكرر.
- **هل فيه مفاتيح أو كلمات سر مكتوبة داخل الكود؟** لا تضع أسرارك في الكود ولا ترفعها إلى مواقع عامة.
- **هل يستورد مكتبات لا أعرفها؟** تحقق من أنها موجودة ومعروفة، فقد تُنسب أحيانًا مكتبات غير موجودة وقد يستغل مخترقون ذلك.
- **هل يرسل بيانات إلى جهات خارجية؟** افحص ما يرسله وإلى أين.
- **هل هو مرخَّص؟** إن نقل المساعد كودًا من مصدر مفتوح فقد يكون له ترخيص يشترط شروطًا.

وتذكّر قاعدة نحرص عليها في الكلية: **ما لا تفهمه لا تسلمه.** وقد رأينا في مقالة الأمانة المهنية أن المسؤولية تبقى على الإنسان.

أسئلة يسألها الطلاب عن البرمجة

هل أحتاج إلى حاسوب قوي؟

لا في البداية. أي حاسوب حديث متوسط يكفي لتعلم بايثون، وهناك بيئات على الإنترنت لا تحتاج إلى تثبيت. وفي المراحل المتقدمة قد تحتاج إلى موارد حوسبة أكبر، وتتوفر خدمات سحابية لذلك، ونعلّم استعمالها بمسؤولية.

أنا لا أحب الرياضيات، فهل أستطيع؟

نعم. تبدأ البرمجة بالمنطق أكثر مما تبدأ بالمعادلات. وحين تصل إلى تعلم الآلة ستحتاج إلى بعض الرياضيات، وسنقدمها لك بالأمثلة والبرمجة لا بالحفظ.

هل أبدأ بدورات مجانية أم ببرنامج منظم؟

المجاني مفيد للتجربة وتحديد ميلك. لكن البرنامج المنظم يعطيك ترتيبًا ومشاريع ومن يراجع عملك ويكشف أخطاءك، وهذا ما يختصر عليك الوقت. والأفضل غالبًا الجمع: جرّب مجانًا ثم التزم حين تتأكد.

كم يستغرق الوصول إلى مستوى يمكّنني من العمل؟

يتوقف على هدفك وجهدك. بعض الوظائف التطبيقية البسيطة، كأتمتة المهام وتحليل الجداول، تتطلب أشهرًا من الدراسة المنتظمة. أما هندسة الأنظمة والنماذج فتحتاج مدة أطول. ولا نعدك بمدة سحرية، لكن برامجنا مصممة بأربع مدد لتختار ما يناسب هدفك.

هل البنات والأولاد سواء في هذا المجال؟

المجال مفتوح لكل من يجتهد. ونلتزم في الكلية بالاحترام وبقيم المجتمع في التعامل والتعليم، ونرحب بأسئلة الطالبات والطلاب وأولياء الأمور عن ترتيبات الدراسة قبل التسجيل.

أخطاء المبتدئين

- **القفز بين اللغات والدورات.** اثبت على لغة واحدة حتى تنجز فيها ثلاثة مشاريع.
- **الإحباط من الأخطاء.** كل مبرمج، مهما كانت خبرته، يقضي نصف وقته في تصحيح الأخطاء. هذا طبيعي.
- **انتظار الفهم الكامل قبل البدء.** تفهم الكثير بالممارسة بعد أن تبدأ.
- **إهمال الإنجليزية التقنية.** أغلب الوثائق والرسائل بالإنجليزية. ولا تحتاج إلى إتقانها، لكن تحتاج إلى قراءتها. ابدأ بجمع المصطلحات في دفتر.
- **الخوف من الرياضيات.** تكفيك رياضيات المرحلة الثانوية لتبدأ. وما يلزم لاحقًا في الذكاء الاصطناعي، كالمتجهات والمشتقات، يُتعلم بشكل تطبيقي مع البرمجة.

- **نسيان الراحة**. الجلوس عشر ساعات أمام الشاشة يضر بصحتك وبانتباهك. اجعل جلساتك قصيرة ومركزة، وحافظ على صلاتك ونومك وحركتك.

قصة متخيلة: ريم وجدول المواعيد

هذه القصة من الخيال لنقرّب الفكرة.

تعمل ريم موظفة استقبال في عيادة أسنان. كانت تقضي ساعتين يوميًا في الاتصال بالمرضى لتذكيرهم بمواعيدهم، وتدوّن من لم يرد. وكانت تحسّ أن العمل يستنزفها.

اشتركت في برنامج تعلم بايثون، وبعد شهرين كتبت برنامجًا صغيرًا يقرأ جدول المواعيد، ويبعث رسالة نصية بصياغة مهذبة قبل الموعد بيوم، ويسجل الردود. استعانت بمساعد ذكي في كتابة الأجزاء الصعبة، لكنها كانت تفهم كل جزء وتختبره على أرقام وهمية قبل أن تجربه.

وفي الأسبوع الأول اكتشفت خطأ: البرنامج أرسل رسالتين لمريض واحد لأن اسمه مكتوب بطريقتين في الجدول. فأصلحت المشكلة وأضافت خطوة تنظيف الأسماء. ولو لم تكن تفهم الكود لما عرفت كيف تتعامل مع الخطأ، ولربما أغضبت المرضى.

قل عدد المواعيد الضائعة، وصار لدى ريم وقت لعمل أهم. ولم يمض وقت طويل حتى طلب مدير العيادة منها أن تتولى تحسين العمليات الإدارية. ولم تصر مهندسة برمجيات، لكنها صارت من تعرف كيف تجعل الآلة تخدمها.

ثلاث درجات من «كتابة الكود»: أين تقف؟

الخلاصة: لا يلزم أن يكون كل إنسان مهندس برمجيات. لكن يلزم أن تعرف أي درجة تناسب عملك.

يمكن أن نقسم التعامل مع البرمجة إلى ثلاث درجات تقريبية:

1. **الاستعمال الواعي**: تستعمل أدوات جاهزة (جداول بيانات متقدمة، وأدوات أتمتة بالنقر، ونماذج لغوية) وتفهم ما وراءها. هذا يكفي كثيرًا من العاملين في الإدارة والتسويق والتعليم والمحاسبة.

2. **البرمجة التطبيقية**: تكتب برامج صغيرة وسكربتات تؤتمت بها عملك وتربط بها الأدوات وتحلل البيانات. وهي أنسب لمن يريد أن يصير «الشخص الذي يعرف كيف يوفر الوقت» في أي جهة.

3. **الهندسة**: تبني أنظمة كبيرة وتصممها وتختبرها وتشغلها وتصونها، وتدرّب النماذج وتنشرها. وهذه تحتاج دراسة أعمق ومدة أطول.

ولا تعد إحدى الدرجات أفضل من غيرها. الأهم أن تختار بحسب هدفك، وأن تعلم أن الانتقال بين الدرجات ممكن. فمن بدأ بالاستعمال الواعي قد يصير بعد سنة مبرمجًا تطبيقيًا، ومن بدأ بالتطبيقي قد يتعمق ليصير مهندسًا. وهذا بالضبط ما تعكسه مدد برامج الكلية الأربع: من فهم المجال وأول مشروع في ثلاثة أشهر، إلى دبلوم تطبيقي في ستة، إلى تأهيل هندسي كامل في سنة، إلى تخصص وبحث في سنتين.

كيف تقرأ كودًا لا تعرفه؟

في العمل الحقيقي ستقرأ من الكود أكثر مما تكتب. فكثير من الوقت يمضي في فهم ما كتبه غيرك أو كتبته الآلة. وهذه طريقة تعينك:

- **ابدأ من الصورة الكبرى**. ما الذي يفعله البرنامج بأكمله؟ ما مدخلاته وما مخرجاته؟
- **ابحث عن نقطة البداية**. أين يبدأ التنفيذ؟ ثم اتبع الخيط.
- **اقرأ أسماء الدوال والمتغيرات**. الأسماء الجيدة تحكي القصة.
- **شغّله بمثال صغير** وتتبع القيم أثناء التنفيذ.
- **غيّر شيئًا صغيرًا** وانظر ماذا يتغير. هذه أسرع وسيلة لتفهم دور الجزء.
- **اكتب بكلماتك ما يفعله كل جزء**. فإن عجزت عن الشرح فأنت لم تفهمه بعد.
- **اسأل مساعدًا ذكيًا لا ليحلّ محلك** بل ليشرح لك سطرًا غامضًا، ثم تحقق.

وهذه المهارة تفيد حتى لغير المبرمجين. فقراءة نص معقد أو عقد أو تقرير هي فعليًا نفس العملية: فهم الصورة الكبرى، ثم التفاصيل، ثم التجربة بأمثلة.

البرمجة والصبر: خلق تتعلمه

لا نبالغ إن قلنا إن أعظم ما تعلمك البرمجة ليس لغة بعينها، بل **الصبر المنظم**. أنت تحاول فتفشل، فتحلل، فتجرب، فتفشل بشكل مختلف، ثم تنجح. وتتعلم أن الفشل في البرمجة ليس حكمًا عليك بل معلومة عن البرنامج.

وهذا يتوافق مع ما ربّانا عليه ديننا من الصبر والأخذ بالأسباب، وقد قال تعالى: ﴿وَاسْتَعِينُوا بِالصَّبْرِ وَالصَّلَاةِ﴾ [البقرة: 45]. فإن أعجزك حل مشكلة فاستعن بالله، وخذ استراحة، وصلِّ، ثم عد بذهن أصفى. فكثير من الحلول تأتي بعد الراحة.

ما الذي تقدمه الكلية في هذا الباب؟

نبدأ في كلية الذكاء الاصطناعي العالمية من الصفر، ونحرص على أن تكون البرمجة وسيلة لبناء أشياء حقيقية:

- **برمجة بايثون للذكاء الاصطناعي**: من أساسيات اللغة (المتغيرات والتحكم والدوال) إلى هياكل البيانات والملفات والاستثناءات، والبرمجة الكائنية والوحدات، ومكتبتي NumPy وPandas لمعالجة الجداول، والاختبار الآلي وكتابة كود مقروء.
- **أدوات المطوّر والبيئات السحابية**: سطر الأوامر ونظام الملفات، وGit والعمل الجماعي، وJupyter والبيئات الافتراضية، ومقدمة في Docker.
- **رياضيات الذكاء الاصطناعي**: الرياضيات التي يحتاجها صانع النماذج، مشروحة بالبرمجة والأمثلة.
- **مشروع التأسيس المصغّر**: تنفذ مشروعًا صغيرًا كاملًا بنفسك وتعرضه أمام لجنة.
- **الذكاء الاصطناعي واللغة العربية**: في البرامج الأطول، تبني أدوات عربية الجودة، وتتعامل مع الصرف العربي واللهجات والتشكيل.

وتشترط برامجنا الأقصر أن يكون الطالب قادرًا على القراءة الأساسية بالإنجليزية وعنده حاسوب واتصال بالإنترنت، ولا تشترط خلفية في البرمجة في برنامج الثلاثة أشهر. ويقدّم الطالب اختبار تحديد مستوى غير إقصائي ليعرف من أين يبدأ.

ما الذي تفعله هذا الأسبوع؟

1. ثبّت بايثون على حاسوبك، أو افتح بيئة تجريبية على الإنترنت، واكتب أول برنامج يطبع اسمك.
2. اختر مشكلة صغيرة تتكرر في حياتك، واكتب خطوات حلها بالعربية.
3. حوّل خطوة واحدة إلى كود وجرّبها.
4. اكسر برنامجك عمدًا بإدخال قيمة غريبة، واقرأ رسالة الخطأ.
5. اطلب من مساعد ذكي أن يشرح لك الخطأ، ثم تحقق من شرحه بنفسك.

الخلاصة

- البرمجة في الأصل تفكير: تحليل وتجريد وخوارزمية.
- المساعدات الذكية تسرّع الكتابة، لكنها تزيد الحاجة إلى من يحدد المطلوب ويراجع ويختبر.
- بايثون بداية مناسبة للدخول إلى الذكاء الاصطناعي.
- تعلّم بالكتابة والتجريب والمشاريع الصغيرة، واقرأ الأخطاء، واستعن بالمساعد معلمًا.
- القيمة الكبرى أن تنتقل من مستهلك للأدوات إلى صانع لها.

وإن أردت أن تبدأ بخطة منظمة ومشاريع ومن يراجع عملك، فتفضل بزيارة صفحة البرامج في موقع الكلية، وراسلنا على admissions@iai.college.

ابدأ طريقك مع كلية الذكاء الاصطناعي العالمية

سجّل طلبك من صفحة التسجيل، أو جرّب اختبار تحديد المستوى، أو راسلنا على

admissions@iai.college · www.iai.college

المراجع والمصادر

1. المنتدى الاقتصادي العالمي، «تقرير مستقبل الوظائف 2025». https://www.weforum.org/press/2025/01/future-of-jobs-report-2025-78-million-new-job-opportunities-by-2030-but-urgent-upskilling-needed-to-prepare-workforces/

2. جامعة ستانفورد، «تقرير مؤشر الذكاء الاصطناعي 2025». https://hai.stanford.edu/news/ai-index-2025-state-of-ai-in-10-charts

3. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

هذه المقالة تعليمية عامة لا تُغني عن استشارة مختص، ولا تَعِد بوظيفة أو دخل. القصص المعنونة «قصة متخيلة» من الخيال لتقريب الفكرة. وكلية الذكاء الاصطناعي العالمية لا تدّعي اعتمادًا حكوميًا لشهاداتها.