VidhyaVision

AI और कोडिंग की शुरुआती गाइड

हिन्दी संस्करण

Python की बुनियाद, AI की शुरुआती समझ, सुरक्षित उपयोग और ऐसे छोटे प्रोजेक्ट जिन्हें सामान्य लैपटॉप पर बनाया जा सके।

> यह गाइड कैसे इस्तेमाल करें
> एक भाग पढ़ें → छोटा उदाहरण चलाएँ → एक चीज़ बदलें → अगली कक्षा या सत्र में अपना सवाल लेकर आएँ।

Python, input() और if/else जैसे कोड के जरूरी शब्द उसी रूप में रखे गए हैं जैसा वे प्रोग्राम लिखते समय दिखाई देंगे। बाकी समझाने वाली सामग्री हिन्दी में है।

- लैपटॉप या स्कूल कंप्यूटर पर काम करें।
- कोड को पंक्ति-दर-पंक्ति समझने की कोशिश करें।
- AI से मिले उत्तर को जाँचे बिना जमा न करें।
- फोन नंबर, पता, पासवर्ड या दूसरी निजी जानकारी सार्वजनिक AI साधनों में साझा न करें।

1. समस्या को छोटे चरणों में बाँटना

कोडिंग शुरू करने से पहले सबसे जरूरी कौशल है किसी समस्या को छोटे, साफ और क्रमबद्ध चरणों में बाँटना। इसे समस्या को व्यवस्थित ढंग से सोचने और हल करने की आदत कह सकते हैं।

> गतिविधि
> रोज़ का कोई काम चुनें—जैसे स्कूल के लिए तैयार होना, चाय बनाना या लाइब्रेरी से किताब लेना—और उसके 5 से 8 स्पष्ट चरण लिखें।

- हर चरण इतना साफ हो कि कोई दूसरा व्यक्ति उसे पढ़कर कर सके।
- देखें कौन-से चरण बार-बार दोहरते हैं।
- सोचें कि कंप्यूटर को किस चरण पर कौन-सी जानकारी चाहिए होगी।

उदाहरण: उपस्थिति दर्ज करने के चरण

- छात्र का नाम लें।
- पूछें कि छात्र उपस्थित है या नहीं।
- जवाब सहेजें।
- अगले छात्र के लिए यही प्रक्रिया दोहराएँ।
- अंत में उपस्थित छात्रों की कुल संख्या दिखाएँ।

2. Python की बुनियाद: वेरिएबल, इनपुट और आउटपुट

वेरिएबल जानकारी को नाम देकर रखने का तरीका है। input() उपयोगकर्ता से जानकारी लेता है और print() स्क्रीन पर परिणाम दिखाता है।

```
name = input("आपका नाम क्या है? ")
age = 13
print("नमस्ते", name)
print("उम्र:", age)
```

अभ्यास

- नाम के साथ उम्र या पसंदीदा विषय पूछें।
- एक नया वेरिएबल school बनाएँ और स्कूल का नाम सहेजें।
- print() की मदद से पूरा वाक्य स्क्रीन पर दिखाएँ।

3. शर्तें, लूप और फ़ंक्शन

if/else प्रोग्राम को अलग स्थिति में अलग निर्णय लेने देता है। लूप किसी काम को बार-बार कराते हैं। फ़ंक्शन बार-बार उपयोग होने वाले कोड को व्यवस्थित रखने में मदद करते हैं।

if / else

```
answer = input("5 x 6 कितना है? ")

if answer == "30":
    print("सही!")
else:
    print("फिर कोशिश करें।")
```

लूप

```
for number in range(1, 6):
    print(number)
```

फ़ंक्शन

```
def greet(name):
    print("नमस्ते", name)

greet("Riya")
```

4. AI क्या है?

कृत्रिम बुद्धिमत्ता (AI) ऐसे कंप्यूटर प्रोग्राम और प्रणालियों का क्षेत्र है जो डेटा या उदाहरणों में पैटर्न पहचानकर अनुमान, वर्गीकरण या नई सामग्री बनाने जैसे काम कर सकते हैं।

ट्रेनिंग डेटा
वे उदाहरण जिनसे मॉडल सीखता है।

मॉडल
डेटा में सीखे गए पैटर्न का गणनात्मक रूप।

अनुमान **(inference)**
नए इनपुट पर प्रशिक्षित मॉडल का परिणाम।

जनरेटिव **AI**
टेक्स्ट, चित्र, ऑडियो या कोड जैसी नई सामग्री बनाने वाला AI।

नियम-आधारित प्रोग्राम और मशीन लर्निंग में फर्क

नियम-आधारित प्रोग्राम में हम नियम खुद लिखते हैं—जैसे “अगर अंक 90 से अधिक हैं तो A दिखाओ।” मशीन लर्निंग में मॉडल कई उदाहरण देखकर पैटर्न सीखता है। दोनों तरीकों का उपयोग अलग-अलग समस्याओं में होता है।

5. जिम्मेदार AI: जाँच, गोपनीयता और पक्षपात

AI उपयोगी है, लेकिन उसका उत्तर हमेशा सही नहीं होता। छात्र को AI के उत्तर को अंतिम सत्य मानने के बजाय उसे जाँचना चाहिए।

तथ्य जाँचें
महत्वपूर्ण जानकारी को पाठ्यपुस्तक, शिक्षक या भरोसेमंद स्रोत से मिलाकर देखें।

निजी जानकारी सुरक्षित रखें
फोन नंबर, पता, पासवर्ड, पहचान संख्या या निजी स्कूल रिकॉर्ड साझा न करें।

पक्षपात समझें
ट्रेनिंग डेटा असंतुलित होने पर मॉडल का परिणाम अधूरा या पक्षपाती हो सकता है।

गलत उत्तर की संभावना समझें
AI आत्मविश्वास के साथ गलत जानकारी भी बना सकता है।

अपना काम समझें

AI से विचार या समझाने में मदद लें, लेकिन अंतिम काम अपनी समझ और अपने शब्दों में करें।

त्वरित जाँच

- इस जानकारी का स्रोत क्या है?
- क्या मैं इसे किसी भरोसेमंद स्रोत से जाँच सकता/सकती हूँ?
- क्या इसमें कोई निजी जानकारी तो नहीं है?

6. बिना कोड के अपना पहला AI क्लासिफ़ायर

Google Teachable Machine जैसे साधन से मशीन लर्निंग का कोड लिखे बिना एक छोटा क्लासिफ़ायर प्रशिक्षित किया जा सकता है। इस गतिविधि का लक्ष्य यह देखना है कि उदाहरणों की गुणवत्ता मॉडल के परिणाम को कैसे बदलती है।

- 2 या 3 सुरक्षित वस्तुओं की श्रेणियाँ चुनें—जैसे नोटबुक, बोतल और पेंसिल।
- हर श्रेणी के कई अलग-अलग उदाहरण इकट्ठे करें।
- मॉडल को प्रशिक्षित करें।
- नए उदाहरणों से जाँचें कि मॉडल सही पहचान कर रहा है या नहीं।
- गलती होने पर सोचें कि ट्रेनिंग डेटा में कौन-सी कमी हो सकती है।

सुरक्षा

पहचान योग्य छात्र चेहरों या निजी तस्वीरों को अनुमति के बिना ट्रेनिंग डेटा में उपयोग न करें।

7. शुरुआती प्रोजेक्ट: क्विज़ गेम

यह छोटा प्रोजेक्ट वेरिएबल, input(), शर्तें और स्कोर गिनने का अभ्यास कराता है। पहले कोड चलाएँ, फिर अपने सवाल जोड़ें।

```
score = 0

answer = input("लाल ग्रह किसे कहा जाता है? ")
if answer.lower() == "mars":
    print("सही!")
    score += 1

print("स्कोर:", score)
```

- विज्ञान या गणित के 5 सवाल जोड़ें।
- राजस्थान के इतिहास या क्रिकेट पर क्विज़ बनाएँ।
- अंत में प्रतिशत स्कोर दिखाएँ।

8. प्रोजेक्ट की योजना: विचार से प्रदर्शन तक

प्रोजेक्ट कठिन होना जरूरी नहीं है। अच्छा शुरुआती प्रोजेक्ट वह है जिसे छात्र खुद चला सके, समझा सके और उसमें कम से कम एक बदलाव कर सके।

चरण	पूछने वाला सवाल
समस्या	मैं किस समस्या या गतिविधि पर काम कर रहा/रही हूँ?
उपयोगकर्ता	यह किसके लिए उपयोगी है?
सबसे छोटा रूप	सबसे सरल चलने वाला संस्करण क्या है?
जाँच	मैं कैसे जाँचूँगा/जाँचूँगी कि यह सही काम करता है?
समझाना	मैं किसी दूसरे छात्र को इसका तर्क कैसे समझाऊँगा/समझाऊँगी?

9. आगे सीखने के लिए भरोसेमंद संसाधन

Code.org / CodeAI

AI की बुनियाद और छोटे वीडियो पाठ।

MIT Scratch

गेम और एनीमेशन के जरिए शुरुआती कोडिंग।

Khan Academy

ब्राउज़र में शुरुआती Python अभ्यास।

Harvard CS50P

Python का अधिक विस्तृत निःशुल्क पाठ्यक्रम।

DIKSHA और ePathshala

स्कूल की पढ़ाई के लिए भारत सरकार और NCERT के डिजिटल संसाधन।

Arduino Learn

हार्डवेयर उपलब्ध होने पर रोबोटिक्स और इलेक्ट्रॉनिक्स अभ्यास।

10. अंतिम चुनौती

इनमें से एक प्रोजेक्ट चुनें: क्विज़ गेम, छात्र उपस्थिति ट्रैकर, लाइब्रेरी बुक मैनेजर, मैथ चैलेंज, होमवर्क ट्रैकर, टिक-टैक-टो या व्यक्तिगत वेबसाइट।

- प्रोजेक्ट चलाएँ।
- कम से कम एक सुविधा बदलें या जोड़ें।
- दो मिनट में किसी दूसरे व्यक्ति को समझाएँ कि आपका कोड कैसे काम करता है।

अगले सत्र के लिए अपना सवाल लिखें

“मैं अब क्या बनाना चाहता/चाहती हूँ, और मुझे किस हिस्से में मदद चाहिए?”