TKTips.org | Python + AI Guide

Home · Python Basics · AI & ML · Tools · FAQ

Python + AI Guide 2026

Master Python programming, automation, AI tools, ChatGPT API, and Machine Learning.
Everything you need to become a Python AI expert in 2026.

Python Basics · Automation · AI Tools · ChatGPT API · Machine Learning · Data Science · FastAPI

Python Basics for Beginners

Python is the most popular language for AI and automation. Here's what you need to start:

Variables & Data Types Beginner
Strings, integers, floats, booleans, lists, tuples, dictionaries

Control Flow Beginner
If-else statements, loops (for, while), list comprehensions

Functions & Modules Beginn
Defining functions, imports, bui

Essential Python Code Examples

Basic Python Syntax

```
"keyword">class="comment"># Variables and data types
"keyword">class="keyword">name = "keyword">class="string">"Python Developer"
"keyword">class="keyword">age = "keyword">class="string">25
"keyword">class="keyword">skills = ["keyword">class="string">"Python", "keyword">class="string">"AI", "keyword">class="string">"ML", "keyword">class="string">"FastAPI"]

"keyword">class="comment"># Control flow example
"keyword">class="keyword">"keyword">for skill "keyword">class="keyword">"keyword">in skills:
    "keyword">class="keyword">"keyword">if skill == "keyword">class="string">"Python":
        "keyword">class="function">print("keyword">class="string">f"Mastering {skill} ">in 2026!")
    "keyword">class="keyword">"keyword">else:
        "keyword">class="function">print("keyword">class="string">f"Learning {skill}")

"keyword">class="comment"># List comprehension
squared_numbers = [x**2 "keyword">class="keyword">"keyword">for x "keyword">class="keyword">"keyword">in "keyword">class="function">range("keyword">class="string">10)]
```

Function Example

```
"keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">process_data(data, operation="keyword">class="string">"analyze"):
    "keyword">class="string">"""Process data ">with different operations"""
    "keyword">class="keyword">"keyword">if operation == "keyword">class="string">"analyze":
        "keyword">class="comment"># Calculate basic statistics
        avg = "keyword">class="function">sum(data) / "keyword">class="function">len(data)
        "keyword">class="keyword">"keyword">return {"keyword">class="string">"average": avg, "keyword">class="string">"max": "keyword">class="function">max(data)}
    "keyword">class="keyword">"keyword">elif operation == "keyword">class="string">"filter":
        "keyword">class="comment"># Filter even numbers
        "keyword">class="keyword">"keyword">return [x "keyword">class="keyword">"keyword">for x "keyword">class="keyword">"keyword">in data "keyword">class="keyword">"keyword">if x % "keyword">
    "keyword">class="keyword">"keyword">else:
        "keyword">class="keyword">"keyword">raise "function">ValueError("keyword">class="string">"Unknown operation")

"keyword">class="comment"># Usage
numbers = ["keyword">class="string">1, "keyword">class="string">2, "keyword">class="string">3, "keyword">class="string">4, "keyword">class="string">5, "keyword">class="string">6]
result = "keyword">class="function">process_data(numbers, "keyword">class="string">"filter")
"keyword">class="function">print(result)  "keyword">class="comment"># Output: [2, 4, 6]
```

Python Automation Scripts

Automate repetitive tasks with Python. Here are practical automation examples:

- **File Management:** Organize files, rename in bulk, convert formats
- **Web Scraping:** Extract data from websites using BeautifulSoup/Scrapy
- **Email Automation:** Send personalized emails, process attachments
- **API Integration:** Connect services, automate data syncing
- **Data Processing:** Clean, transform, and analyze datasets automatically

Automation Script Example

File Organizer Automation

```
"keyword">class="keyword">"keyword">import os
"keyword">class="keyword">"keyword">import shutil
"keyword">class="keyword">"keyword">from pathlib "keyword">class="keyword">"keyword">import Path

"keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">organize_files(directory):
    "keyword">class="string">"""Organize files by extension into folders"""
    file_types = {
        "keyword">class="string">'Images': ["keyword">class="string">'.jpg', "keyword">class="string">'.jpeg', "keyword">class="string">'.png', "keyword">class="string">'.gif'],
        "keyword">class="string">'Documents': ["keyword">class="string">'.pdf', "keyword">class="string">'.docx', "keyword">class="string">'.txt', "keyword">class="string">'.xlsx'],
        "keyword">class="string">'Code': ["keyword">class="string">'.py', "keyword">class="string">'.js', "keyword">class="string">'.html', "keyword">class="string">'.css'],
        "keyword">class="string">'Videos': ["keyword">class="string">'.mp4', "keyword">class="string">'.mov', "keyword">class="string">'.avi']
    }

    "keyword">class="keyword">"keyword">for file "keyword">class="keyword">"keyword">in "function">Path(directory)."function">iterdir():
        "keyword">class="keyword">"keyword">if file."function">is_file():
            file_ext = file.suffix."function">lower()
            "keyword">class="keyword">"keyword">for folder, extensions "keyword">class="keyword">"keyword">in file_types."function">items():
                "keyword">class="keyword">"keyword">if file_ext "keyword">class="keyword">"keyword">in extensions:
                    folder_path = "function">Path(directory) / folder
                    folder_path."function">mkdir(exist_ok="keyword">class="keyword">"keyword">True)
                    shutil."function">move("function">str(file), "function">str(folder_path / file.name))
                    "keyword">class="function">print("keyword">class="string">f"Moved {file.name} to {folder}/")
                    "keyword">class="keyword">break

"keyword">class="comment"># Run the organizer
"keyword">class="keyword">"keyword">if __name__ == "keyword">class="string">"__main__":
    "keyword">class="function">organize_files("keyword">class="string">"./Downloads")
```

Automation Tip: Use Python's schedule or cron to run automation scripts at specific times automatically.

Essential AI Tools & Libraries 2026

Tool/Library	Purpose	Installation
OpenAI API	ChatGPT, GPT-4, DALL-E integration	pip install openai
LangChain	Build LLM-powered applications	pip install langchain
Hugging Face	Pre-trained models & datasets	pip install transformers
TensorFlow/PyTorch	Deep learning frameworks	pip install tensorflow
scikit-learn	Traditional machine learning	pip install scikit-learn
FastAPI	Build AI API endpoints	pip install fastapi uvicorn

AI Tool Integration Example

AI-Powered Content Generator

```
"keyword">class="keyword">"keyword">import openai
"keyword">class="keyword">"keyword">from typing "keyword">class="keyword">"keyword">import List

"keyword">class="keyword">"keyword">class "keyword">class="function">ContentGenerator:
    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">__init__("keyword">class="keyword">self, api_key: str):
        openai.api_key = api_key
        "keyword">class="keyword">self.model = "keyword">class="string">"gpt-4"  "keyword">class="comment"># Use gpt-4-turbo "keyword">in 2026

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">generate_blog_post("keyword">class="keyword">self, topic: str, tone: str = "keyword">class="string">"professiona
        "keyword">class="string">"""Generate a blog post using AI"""
        prompt = "keyword">class="string">f"""Write a comprehensive blog post about {topic}.
        Tone: {tone}
        Include:
        1. Introduction
        2. Key concepts
        3. Practical examples
        4. Conclusion
        5. 3 actionable tips"""

        response = openai.ChatCompletion."function">create(
            model="keyword">class="keyword">self.model,
            messages=[{"keyword">class="string">"role": "keyword">class="string">"user", "keyword">class="string">"content": prompt}],
            max_tokens="keyword">class="string">1500,
            temperature="keyword">class="string">0.7
        )

        "keyword">class="keyword">"keyword">return response.choices["keyword">class="string">0].message.content

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">generate_code("keyword">class="keyword">self, task: str, language: str = "keyword">class="string">"python") ->
        "keyword">class="string">"""Generate code ">for a specific task"""
        prompt = "keyword">class="string">f"Write {language} code to {task}. Include comments."

        response = openai.ChatCompletion."function">create(
            model="keyword">class="keyword">self.model,
            messages=[{"keyword">class="string">"role": "keyword">class="string">"user", "keyword">class="string">"content": prompt}],
            temperature="keyword">class="string">0.3  "keyword">class="comment"># Lower temperature "keyword">for code generation
        )

        "keyword">class="keyword">"keyword">return response.choices["keyword">class="string">0].message.content

"keyword">class="comment"># Usage example
"keyword">class="keyword">"keyword">if __name__ == "keyword">class="string">"__main__":
    generator = "keyword">class="function">ContentGenerator(api_key="keyword">class="string">"your-api-key-here")
    blog_post = generator."function">generate_blog_post("keyword">class="string">"Python automation ">in 2026")
    "keyword">class="function">print(blog_post)
```

ChatGPT API Integration

Build applications with OpenAI's ChatGPT API. Here's how to integrate it effectively:

Basic ChatGPT API Setup

```
import openai
import json

# Initialize with your API key
openai.api_key = "sk-your-api-key-here"

def chat_with_gpt(messages, model="gpt-4", temperature=0.7):
    """Send messages to ChatGPT and get response"""
    try
        response = openai.ChatCompletion.create(
            model=model,
            messages=messages,
            temperature=temperature,
            max_tokens=1000
        )
        return response.choices[0].message.content
    except Exception as e:
        return f"Error: {str(e)}"

# Example conversation
messages = [
    {"role": "system", "content": "You are a helpful Python programming assistant."},
    {"role": "user", "content": "How do I optimize Python code for speed?"}
]

response = chat_with_gpt(messages)
print(response)
```

Advanced API Features

- **Streaming Responses:** Get responses in real-time for better UX
- **Function Calling:** Let ChatGPT call your functions
- **Fine-tuning:** Customize models for specific tasks
- **Context Management:**Handle long conversations with token limits

$ Cost Optimization: Use gpt-3.5-turbo for simple tasks, gpt-4 for complex reasoning. Monitor token usage with OpenAI's usage dashboard.

Machine Learning Roadmap 2026

Month 1-2: Foundations
Python basics → NumPy/Pandas → Data visualization (Matplotlib/Seaborn) → Statistics basics

Month 3-4: Traditional ML
scikit-learn → Regression/Classification → Model evaluation → Feature engineering

Month 5-6: Deep Learning
Neural networks basics → TensorFlow/PyTorch → CNN for images → RNN/LSTM for text

Month 7-8: Advanced Topics
Transfer learning → Transformers → LLMs → Deployment (FastAPI/Docker)

Complete ML Project Example

End-to-End ML Pipeline

```
"keyword">class="keyword">"keyword">import pandas "keyword">class="keyword">"keyword">as pd
"keyword">class="keyword">"keyword">import numpy "keyword">class="keyword">"keyword">as np
"keyword">class="keyword">"keyword">from sklearn.model_selection "keyword">class="keyword">"keyword">import train_test_split
"keyword">class="keyword">"keyword">from sklearn.ensemble "keyword">class="keyword">"keyword">import RandomForestClassifier
"keyword">class="keyword">"keyword">from sklearn.metrics "keyword">class="keyword">"keyword">import accuracy_score, classification_report
"keyword">class="keyword">"keyword">import joblib

"keyword">class="keyword">"keyword">class "keyword">class="function">MLPipeline:
    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">__init__("keyword">class="keyword">self, data_path: str):
        "keyword">class="keyword">self.data = pd."function">read_csv(data_path)
        "keyword">class="keyword">self.model = "keyword">class="keyword">"keyword">None

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">preprocess_data("keyword">class="keyword">self):
        "keyword">class="string">"""Clean and prepare data ">for ML"""
        "keyword">class="comment"># Handle missing values
        "keyword">class="keyword">self.data = "keyword">class="keyword">self.data."function">fillna("keyword">class="keyword">self.data."function">median())

        "keyword">class="comment"># Encode categorical variables
        categorical_cols = "keyword">class="keyword">self.data."function">select_dtypes(include=["keyword">class="string">'object']).columns
        "keyword">class="keyword">"keyword">for col "keyword">class="keyword">"keyword">in categorical_cols:
            "keyword">class="keyword">self.data[col] = pd."function">Categorical("keyword">class="keyword">self.data[col]).codes

        "keyword">class="keyword">"keyword">return "keyword">class="keyword">self.data

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">train_model("keyword">class="keyword">self, target_column: str, test_size: float = "keyword">class="string">0.2):
        "keyword">class="string">"""Train a Random Forest classifier"""
        "keyword">class="comment"># Prepare features and target
        X = "keyword">class="keyword">self.data."function">drop(columns=[target_column])
        y = "keyword">class="keyword">self.data[target_column]

        "keyword">class="comment"># Split data
        X_train, X_test, y_train, y_test = "function">train_test_split(
            X, y, test_size=test_size, random_state="keyword">class="string">42
        )

        "keyword">class="comment"># Train model
        "keyword">class="keyword">self.model = "function">RandomForestClassifier(n_estimators="keyword">class="string">100, random_state="keyword">class="string">42)
        "keyword">class="keyword">self.model."function">fit(X_train, y_train)

        "keyword">class="comment"># Evaluate
        y_pred = "keyword">class="keyword">self.model."function">predict(X_test)
        accuracy = "function">accuracy_score(y_test, y_pred)

        "keyword">class="function">print("keyword">class="string">f"Model Accuracy: {accuracy:.2f}")
        "keyword">class="function">print("keyword">class="string">"\nClassification Report:")
        "keyword">class="function">print("function">classification_report(y_test, y_pred))

        "keyword">class="keyword">"keyword">return "keyword">class="keyword">self.model

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">save_model("keyword">class="keyword">self, filename: str):
        "keyword">class="string">"""Save trained model to file"""
        "keyword">class="keyword">"keyword">if "keyword">class="keyword">self.model:
            joblib."function">dump("keyword">class="keyword">self.model, filename)
            "keyword">class="function">print("keyword">class="string">f"Model saved to {filename}")

    "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">load_model("keyword">class="keyword">self, filename: str):
        "keyword">class="string">"""Load model ">from file"""
        "keyword">class="keyword">self.model = joblib."function">load(filename)
        "keyword">class="function">print("keyword">class="string">f"Model loaded ">from {filename}")

"keyword">class="comment"># Usage
"keyword">class="keyword">"keyword">if __name__ == "keyword">class="string">"__main__":
    pipeline = "keyword">class="function">MLPipeline("keyword">class="string">"data.csv")
    pipeline."function">preprocess_data()
    pipeline."function">train_model(target_column="keyword">class="string">"target")
    pipeline."function">save_model("keyword">class="string">"model.pkl")
```

Python + AI Project Ideas

FastAPI AI Endpoint Example

FastAPI AI Service

```
"keyword">class="keyword">"keyword">from fastapi "keyword">class="keyword">"keyword">import FastAPI, HTTPException
"keyword">class="keyword">"keyword">from pydantic "keyword">class="keyword">"keyword">import BaseModel
"keyword">class="keyword">"keyword">import openai
"keyword">class="keyword">"keyword">import uvicorn

app = "function">FastAPI(title="keyword">class="string">"Python AI API", version="keyword">class="string">"1.0")

"keyword">class="comment"># Request models
"keyword">class="keyword">"keyword">class "keyword">class="function">ChatRequest(BaseModel):
    message: str
    model: str = "keyword">class="string">"gpt-3.5-turbo"

"keyword">class="keyword">"keyword">class "keyword">class="function">CodeRequest(BaseModel):
    task: str
    language: str = "keyword">class="string">"python"

"keyword">class="comment"># Initialize OpenAI (set "keyword">in environment variables)
openai.api_key = "keyword">class="string">"your-api-key"

@app."function">get("keyword">class="string">"/")
"keyword">class="keyword">"keyword">async "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">root():
    "keyword">class="keyword">"keyword">return {"keyword">class="string">"message": "keyword">class="string">"Python AI API is running!"}

@app."function">post("keyword">class="string">"/chat")
"keyword">class="keyword">"keyword">async "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">chat(request: ChatRequest):
    "keyword">class="string">"""Chat ">with AI"""
    "keyword">class="keyword">"keyword">try:
        response = openai.ChatCompletion."function">create(
            model=request.model,
            messages=[{"keyword">class="string">"role": "keyword">class="string">"user", "keyword">class="string">"content": request.message}],
            max_tokens="keyword">class="string">500
        )
        "keyword">class="keyword">"keyword">return {"keyword">class="string">"response": response.choices["keyword">class="string">0].message.content}
    "keyword">class="keyword">"keyword">except Exception "keyword">class="keyword">"keyword">as e:
        "keyword">class="keyword">"keyword">raise "function">HTTPException(status_code="keyword">class="string">500, detail="function">str(e))

@app."function">post("keyword">class="string">"/generate-code")
"keyword">class="keyword">"keyword">async "keyword">class="keyword">"keyword">class="keyword">def "keyword">class="function">generate_code(request: CodeRequest):
    "keyword">class="string">"""Generate code ">for a task"""
    prompt = "keyword">class="string">f"Write {request.language} code to {request.task}"

    "keyword">class="keyword">"keyword">try:
        response = openai.ChatCompletion."function">create(
            model="keyword">class="string">"gpt-4",
            messages=[{"keyword">class="string">"role": "keyword">class="string">"user", "keyword">class="string">"content": prompt}],
            temperature="keyword">class="string">0.3
        )
        "keyword">class="keyword">"keyword">return {"keyword">class="string">"code": response.choices["keyword">class="string">0].message.content}
    "keyword">class="keyword">"keyword">except Exception "keyword">class="keyword">"keyword">as e:
        "keyword">class="keyword">"keyword">raise "function">HTTPException(status_code="keyword">class="string">500, detail="function">str(e))

"keyword">class="keyword">"keyword">if __name__ == "keyword">class="string">"__main__":
    uvicorn."function">run(app, host="keyword">class="string">"0.0.0.0", port="keyword">class="string">8000)
```

Learning Resources 2026

Resource	Type	Focus Area
Python.org Documentation	Official Docs	Python Language
FastAPI Documentation	Official Docs	API Development
Hugging Face Course	Free Course	Transformers & NLP
OpenAI Cookbook	GitHub Repo	API Examples
Kaggle Learn	Free Courses	Data Science & ML
TKTips.org Python PDF	Premium Guide	Complete Python + AI

Learning Path: Start with Python basics → Build small automation scripts → Learn API integration → Dive into ML → Build full-stack AI applications.

About TKTips.org Python

Your complete resource for Python programming, automation, AI integration, and machine learning. Learn to build practical applications.

Python Topics

- Python Basics
- Web Development
- Data Science
- Machine Learning
- Automation

AI Resources

- ChatGPT API
- LangChain
- TensorFlow
- FastAPI
- Deployment

Newsletter

Get weekly Python tips, AI updates, and code examples.

Your email address

Subscribe

© 2026 TKTips.org - Python + AI Guide. All rights reserved.
Python is a trademark of the Python Software Foundation.