大分類	中分類	小分類	番号	問題文	選択肢1	選択肢2	選択肢3	選択肢4	正解	解説
PythonとWeb処理の基礎	Pythonの基本構文	変数、データ型（リスト、辞書、タプルなど）	1	Pythonで変数 x に数値 10 を代入する正しい方法はどれですか？	int x = 10	let x = 10	x = 10	var x = 10	3	Pythonでは変数を宣言する際に型の指定やキーワードを使用する必要はありません。単純に x = 10 と書くだけで変数 x に値 10 を代入できます。Pythonは動的型付け言語なので、変数の型は自動で判断されます。
			2	Pythonでリストの型を調べるにはどうすればよいですか？	check_type(my_list)	gettype(my_list)	typeof(my_list)	type(my_list)	4	Pythonでは、type() 関数を使うことで任意の変数のデータ型を調べることができます。例えば、type([1, 2, 3])は < class list > を返します。他の選択肢はPythonに存在しない関数名です。
			3	次のコードの出力は何ですか？ person = {"name": "Alice", "age": 25} print(person["age"])	name	25	Alic	None	2	辞書はキーと値のペアでデータを格納します。person["age"] はキー "age" に対応する値 25 を返します。
			4	次のコードについて正しい説明はどれですか？ my_tuple = (1, 2, 3) my_tuple[0] = 5	タプルの最初の要素が 5 になる。	タプルに新しい要素を追加できる。	エラーが発生する。	タプルの要素を削除できる。	3	タプルはイミュータブル（変更不可能）なデータ型です。要素の追加、削除、変更はできません。そのため、my_tuple[0] = 5 を試みるとエラーになります。

		制御構文（if文、forループ、whileループ）	5	次のコードの出力は何ですか？
				x = 10
				if x > 5:
				  print("Greater")
				else:
				  print("Smaller")
					Greater	Smaller	エラーが発生する	何も表示されない	1	if 文は条件式 x > 5 を評価します。この場合、x は 10 なので、条件は真（True）になります。したがって、"Greater" が出力されます。

			6	for文を使って0〜2を出力するコードはどれですか？	for i in range(1, 3): print(i)	for i in range(3): print(i)	for i in [1, 2, 3]: print(i)	for i in range(0, 3, 2): print(i)	2	range(3) は 0 から 2 までの整数を生成します（終了値は含まれない）。したがって、print(i) で 0、1、2 が順に出力されます。「for i in range(1, 3): print(i)」は 1〜2、「for i in [1, 2, 3]: print(i)」は 1〜3 を出力、「for i in range(0, 3, 2): print(i)」は 0 と 2 のみです。
			7	次のコードについて、ループが実行される回数は何回ですか？ x = 0 while x < 5: x += 1	3 回	4 回	5 回	無限ループになる	3	while ループは条件 x < 5 が真の間繰り返されます。ループ内で x += 1 により x の値が増え、5 に達するとループが終了します。合計で 5 回ループが実行されます。

			8	次のコードの出力は何ですか？
				x = 7
				if x > 5:
				  if x < 10:
				  print("In range")
				  else:
				  print("Too high")
				else:
				  print("Too low")
					Too high	Too low	In range	エラーが発生する	3	最初の if 文で x > 5 が真、次の if 文で x < 10 も真なので、"In range" が出力されます。条件がネスト（入れ子）されていても、順に評価されます。

		関数とモジュールの使い方	9	次のコードの出力は何ですか？ def greet(name): return "Hello, " + name print(greet("Alice"))	Hello, Alice	Hello Alice	Hello, Alice（引用符なし）	greet	1	greet 関数は引数 name を受け取り、"Hello, " に name を連結した文字列を返します。greet("Alice") を呼び出すと "Hello, Alice" が返され、それが print で表示されます。
			10	次のコードの出力は何ですか？ def add(a, b): return a + b print(add(3, 4))	7	34	3 + 4	エラーが発生する	1	add 関数は2つの引数 a と b を受け取り、その和を返します。add(3, 4) を呼び出すと 3 + 4 の結果である 7 が返され、表示されます。
			11	次のコードの出力として正しいものはどれですか？ def greet(name="Guest"): print("Hello, " + name) greet()	Hello, name	Hello, Guest	Hello,	エラーが発生する	2	関数 greet() にはデフォルト引数 name="Guest" が設定されているため、引数を渡さなくても "Guest" が使用され、print("Hello, Guest") と出力されます。「Hello, name」は変数展開されない例、「Hello,」は引数が空、「エラーが発生する」は構文的に誤りです。
			12	次のコードについて、正しい文法はどれですか？	import math.square(4)	import math.sqrt	from math import sqrt	from math import square	3	math モジュールの sqrt 関数を使用するには、from math import sqrt と指定します。選択肢 import math.square(4) と import math.sqrt は誤った構文、選択肢 from math import square は square が math に存在しないため不正解です。
	Web処理とHTTPの基礎	HTTPの基本概念	13	HTTPはどのような目的で使用されるプロトコルですか？	メールの送受信を行うためのプロトコル	ファイルを暗号化して保存するためのプロトコル	ネットワーク上の機器を識別するためのプロトコル	Web上で情報の送受信を行うためのプロトコル	4	HTTP（HyperText Transfer Protocol）は、WebブラウザとWebサーバーの間でテキストや画像などの情報を送受信するために使用されるプロトコルです。メール送受信にはSMTPやIMAP、機器識別にはIPなど、用途に応じて他のプロトコルが使われます。
			14	HTTP通信における「クライアント」と「サーバー」の関係を説明してください。	クライアントは情報を提供し、サーバーが要求する側である	サーバーが情報を提供し、クライアントがそれを要求する	クライアントとサーバーは対等な役割を持つ	クライアントがWebページを保存し、サーバーがそれを参照する	2	HTTP通信では、クライアント（たとえばWebブラウザ）がサーバーにリクエストを送り、サーバーがレスポンスとして情報を返すという役割関係にあります。情報の提供はサーバー側であり、クライアントはその利用者です。
			15	HTTP通信が「ステートレス」であるとは、どのような意味ですか？	通信が暗号化されないことを意味する	サーバーがクライアントの状態を記憶しないことを意味する	クライアントが一度だけ通信できることを意味する	接続が常に切断されていることを意味する	2	HTTPは「ステートレス」なプロトコルであり、各リクエストが独立しており、サーバーは前のリクエストの状態を保持しません。そのため、ログイン状態などを保持するにはCookieやセッションなどの補助技術が必要です。
			16	WebブラウザがHTTP通信において果たす役割は何ですか？	ユーザーの代わりにWebサーバーを構築する	HTTPリクエストを送信し、レスポンスを表示する	サーバーの設定を遠隔で変更する	インターネット接続そのものを提供する	2	Webブラウザは、ユーザーが入力したURLに基づきHTTPリクエストを送信し、サーバーから返ってきたHTMLや画像などのレスポンスを表示するクライアントの役割を果たします。サーバーを構築したり、ネット接続を提供する機能はありません。
		標準ライブラリによるWebサーバー	17	Pythonの標準ライブラリを使用して簡易なHTTPサーバーを起動するコマンドはどれですか？	python -m http.server	python -m flask run	python -http start	python http_server.py	1	Pythonには簡易HTTPサーバーを立ち上げるための標準ライブラリhttp.serverが用意されています。python -m http.server を実行することで、現在のディレクトリを公開するWebサーバーが起動します。
			18	http.server モジュールを使ってサーバーを起動したとき、どのディレクトリが公開されますか？	Pythonのインストールディレクトリ	スクリプトが置かれているディレクトリ	ユーザーのホームディレクトリ	サーバーを起動したカレントディレクトリ	4	http.server は コマンドを実行したときの作業ディレクトリ（カレントディレクトリ）をWeb上に公開します。起動時にどのディレクトリにいるかがそのままWebルートになります。
			19	Pythonの http.server を使用したサーバーを終了するために一般的に使われる操作はどれですか？	ブラウザを閉じる	exit() コマンドを入力する	サーバーのコードに stop() を追加する	Ctrl + C を押す	4	http.server は通常、ターミナル上で実行されており、Ctrl + C による割り込みでサーバーを停止するのが一般的です。これはUNIX系OSでもWindowsでもよく使われる終了操作です。
			20	標準ライブラリを用いたWebサーバーで、ポート番号8080を指定する理由として正しいものはどれですか？	8080はセキュリティ的に安全なポートであるため	80番ポートが使用中または管理者権限が必要なため	Pythonが8080専用に設計されているため	インターネットに公開するには8080しか使えないため	2	標準的なHTTP通信はポート80を使いますが、ポート80は管理者権限がないと使用できないことが多いため、開発時には8080などの高い番号のポートを使うのが一般的です。8080はよく使われる代替ポートの1つです。

		HTTPリクエストとレスポンスの構造	21	HTTPリクエストの最初の行にはどのような情報が含まれますか？	リクエストメソッド・パス・HTTPバージョン	クッキー情報	IPアドレス・ポート番号	サーバーからのレスポンスコード	1	HTTPリクエストの最初の行は「リクエストライン」と呼ばれ、GETやPOSTなどのメソッド、パス（URL）、そしてHTTPのバージョン（例：HTTP/1.1）が記述されます。
			22	GET リクエストと POST リクエストの違いを簡単に説明してください。	GETはサーバーにファイルを送信し、POSTは起動させるファイルを取得する	GETはサーバーを停止させ、POSTは起動させる	GETはURLにデータを含め、POSTは本文にデータを含める	GETはセッション情報を破棄し、POSTは保存する	3	GETはパラメータをURLに含めて送信するのに対し、POSTはデータをリクエストの本文（ボディ）に含めて送信します。一般的に、GETはデータの取得、POSTはデータの登録・更新に使用されます。
			23	HTTPレスポンスのステータスコード「200 OK」はどのような意味ですか？	クライアントのリクエストが見つからなかった	サーバーエラーが発生した	認証が必要である	リクエストが正常に処理された	4	200 OKは、HTTPレスポンスにおいてリクエストが成功し、正常に処理されたことを意味します。最も一般的な成功ステータスコードです。
			24	HTTPヘッダーに含まれる Content-Type の役割は何ですか？	リクエスト送信元のIPアドレスを指定する	サーバーの種類を指定する	メッセージ本文のデータ形式を示す	ステータスコードの詳細を記述する	3	Content-Type ヘッダーは、送信されるデータの種類（たとえば HTML、JSON、画像など）をクライアントやサーバーに伝えるための情報です。これにより、受け取った側が正しい方法でデータを処理できます。
	Webフレームワークの導入（Flask）	Flaskのインストールと基本的な使い方	25	Pythonの仮想環境内でFlaskをインストールするためのコマンドとして正しいものはどれですか？	pip install flask	python install flask	flask install	pip flask install	1	Flaskをインストールするには pip install flask を使用します。pip はPythonパッケージ管理ツールで、必要なライブラリのインストールに使用されます。

			26	以下のコードのうち、Flaskアプリケーションを作成するための正しい基本構造はどれですか？	import flask
					app = flask.Flask(__name__)
					@app.route('/')
					def home():
					  return 'Hello, World!'
						from flask import Flask
						app = Flask('app_name')
						@app.route('/')
						def index():
						  return 'Welcome to Flask!'
							from flask import Flask
							app = Flask(__name__)
							@app.route('/')
							def hello():
							  return 'Hello, Flask!'
								import Flask
								app = Flask()
								def main():
								  print('Running Flask')
									3	Flaskアプリケーションの基本構造は Flask(__name__) のようにインスタンスを作成し、@app.route デコレータでルートを定義します。選択肢Cはその正しい例です。

			27	Flaskアプリケーションを起動するコマンドとして適切なものはどれですか？	python app.py	flask run app.py	run flask app.py	python run.py	1	Flaskアプリケーションのスクリプトファイルが app.py である場合、python app.py で実行できます。また、flask run コマンドもありますが、設定ファイルに基づく起動方法です。
			28	Flaskで flask run を実行すると、アプリケーションはどのポートで起動しますか？	ポート 80	ポート 8080	ポート 5000	ポート 3000	3	Flaskはデフォルトでポート5000を使用してアプリケーションを起動します。ポート番号を変更する場合は、flask run --port=XXXX で指定できます。
			29	Flaskアプリケーションを終了するには、一般的にどのような操作を行いますか？	flask shutdown コマンドを使う	ブラウザのタブを閉じる	Flask のコードに app.stop() を追加する	ターミナルで Ctrl + C を押す	4	Flask アプリケーションはターミナル上で起動しているため、一般的には Ctrl + C による割り込み操作でアプリケーションを停止します。これは他のサーバープロセスと同様の操作です。

		ルーティング（URLと関数の紐付け）	30	FlaskアプリケーションでルートURL（/）にアクセスしたときに「Welcome!」と表示するには、どのコードが正しいですか？	@app.route('/welcome')
					def welcome():
					  return 'Welcome!'
						@app.route('/')
						def home():
						  return 'Welcome!'
							@app.route('/')
							def index():
							  print('Welcome!')
								def home():
								  return 'Welcome!'
									2	@app.route('/') でルートURLを指定し、そのURLにアクセスしたときに実行される関数を定義します。関数内で return を使って文字列を返すと、それがページに表示されます。

			31	URLに埋め込まれたパラメータ（例： /user/John ）を取得する方法を説明してください。	request.form['user'] で取得する	@app.route("/user/<username>") を使って関数引数で受け取る	request.get('username') を使って取得する	URL末尾のパスを自動的に変数に代入する機能を使う	2	Flask では、URLパラメータを <パラメータ名> の形式でルートに定義し、それを関数の引数で受け取ることで処理できます。例: @app.route ("/user/<username>") に対して def user(username): で値を取得します。

			32	Flaskのルーティングで、HTTPメソッドを POST のみに制限したい場合の指定方法はどれですか？	@app.route('/submit', methods=['POST'])
					def submit():
					  return 'Form submitted!'
						@app.route('/submit')
						def submit():
						  return 'Form submitted!'
							@app.route('/submit', method='POST')
							def submit():
							  return 'Form submitted!'
								@app.route('/submit', methods=['GET'])
								def submit():
								  return 'Form submitted!'
									1	Flaskの @app.route デコレータで methods パラメータを使うことで、ルートで使用できるHTTPメソッドを制限できます。リスト形式で指定し、複数のメソッドも指定可能です。

			33	Flaskアプリで、定義されていないルートにアクセスするとどのエラーコードが表示されますか？	200	400	404	500	3	ルートが見つからない場合、FlaskはHTTPステータスコード 404 Not Found を返します。これはWeb標準のエラーコードで、URLが存在しないことを示します。
			34	1つのURLに対して複数のHTTPメソッド（GETとPOSTなど）を許可したい場合の記述例は？	@app.route("/submit", methods=["GET", "POST"])	@app.route("/submit", method=["GET", "POST"])	@app.route("/submit", http=["GET", "POST"])	@app.route("/submit").method("GET", "POST")	1	Flaskでは、methods 引数を使ってリスト形式で許可するHTTPメソッドを指定します。これにより、同じURLに対してGETやPOSTの両方のリクエストを処理できます。
			35	Flaskアプリでルーティング設定を変更した際、変更が即時反映されない原因として考えられる設定は何ですか？	Flaskのバージョンが古い	debug=True が設定されていない	URLの指定が相対パスでない	app.route() の書き方が間違っている	2	Flaskでコード変更を即時反映するには、デバッグモード（debug=True）を有効にする必要があります。これにより、アプリのコードが変更された際に自動で再起動されます。

		動的なレスポンスの作成	36	Flaskでユーザーの名前を受け取り、"Hello, [ユーザー名]!" をHTMLとして表示する関数を作成するときのコードとして正しいものはどれですか？	@app.route('/hello/<name>')
					def hello(name):
					  return ' Hello, name! '
						@app.route('/hello/<name>')
						def hello():
						  return f' Hello, {name}! '
							@app.route('/hello/<name>')
							def hello(name):
							  return f' Hello, {name}! '
								@app.route('/hello/')
								def hello(name):
								  return ' Hello, name! '
									3	動的なコンテンツを生成するには、ルートのURLパラメータを関数の引数として受け取り、f-string を使って文字列をフォーマットします。

			37	FlaskでJSON形式のレスポンスを返すために正しい方法はどれですか？	@app.route('/json')
					def get_json():
					  return '{ "message": "Hello, World!" }'
						@app.route('/json')
						def get_json():
						  return json.dumps({ "message": "Hello, World!" })
							@app.route('/json')
							def get_json():
							  return jsonify({ "message": "Hello, World!" })
								@app.route('/json')
								def get_json():
								  return { "message": "Hello, World!" }
									3	Flaskの jsonify 関数を使うと、辞書型のデータを自動的にJSON形式に変換し、適切なContent-Typeヘッダーを付加してレスポンスを返します。

			38	render_template 関数を使用する主な目的は何ですか？	HTMLファイルを圧縮して送信するため	JavaScriptコードをFlaskで実行するため	HTMLテンプレートにデータを埋め込んでレスポンスを返すため	CSSファイルを動的に読み込むため	3	Flaskのrender_template関数は、HTMLテンプレートと変数を組み合わせて、ユーザーに返すHTMLを生成するために使用されます。テンプレートエンジン（Jinja2）と連携して動的なHTMLを構築できます。

			39	Flaskでメッセージとともにステータスコード 201 Created を返すにはどうすればよいですか？	@app.route('/create')
					def create():
					  return 'Created', 201
						@app.route('/create')
						def create():
						  return 'Created'
							@app.route('/create')
							def create():
							  return 'Created', 404
								@app.route('/create')
								def create():
								  return 'Created', 200
									1	Flaskではレスポンスを返す際にタプル形式でメッセージとステータスコードを指定できます。201 はリソースが正常に作成されたことを示すHTTPステータスコードです。

			40	テンプレートエンジンで変数をHTMLに埋め込む構文は何ですか？	{{ variable }}	[% variable %]	<< variable >>	((variable))	1	Flaskで使用されているテンプレートエンジン（Jinja2）では、変数を埋め込むには {{ 変数名 }} の構文を使います。これはHTMLテンプレート内で値を表示する際に頻繁に使われます。
			41	POSTメソッドで送信されたフォームデータをFlaskで受け取る方法はどれですか？	request.get("form")	form.request["data"]	request.POST["フィールド名"]	request.form["フィールド名"]	4	Flaskでは、POSTメソッドで送信されたフォームデータは、request.form["キー名"] でアクセスできます。これは辞書のように扱われ、HTMLフォームのname属性に対応する値を取得します。
データとAPIの活用	SQLとデータベース設計	基本的なCRUD操作	42	データベースから全ての列を取得するSQL文として正しいものはどれですか？	GET * FROM table;	SELECT ALL FROM table;	FETCH * FROM table;	SELECT * FROM table;	4	SQLにおいて、すべての列を取得するには SELECT * FROM テーブル名; を使用します。* はワイルドカードで全列を意味します。
			43	INSERT 文を使ってデータを追加する際に必要な情報は何ですか？	テーブル名とWHERE句	テーブル名とカラム名、追加する値	主キーだけ	テーブル名とORDER句	2	INSERT文では、追加対象のテーブル名と、どのカラムにどの値を挿入するかを明示する必要があります。例：INSERT INTO users (name, age) VALUES ('Tom', 25);
			44	UPDATE 文で特定のレコードの内容を変更するには、どのような構文になりますか？	UPDATE users SET name = 'Ken' SELECT *;	UPDATE users name = 'Ken' WHERE id = 1;	UPDATE users SET name = 'Ken' WHERE id = 1;	CHANGE users name = 'Ken' WHERE id = 1;	3	UPDATE文は、特定のレコードを変更するために SET で更新内容を指定し、WHEREで対象行を特定します。WHERE句がないとすべての行が更新されてしまいます。
			45	DELETE 文を使って全てのレコードを削除するとどうなりますか？	テーブルが削除される	削除できずエラーになる	テーブル構造だけが残り、全データが消える	主キーだけが残る	3	DELETE FROM テーブル名; と書くと、すべてのデータが削除されますが、テーブル自体（構造）は残ります。これは DROP文との違いです。
			46	WHERE 句の役割について説明してください。	表の列を指定する	レコードの削除を行う	結果を昇順に並び替える	条件に一致するレコードのみを抽出・更新・削除する	4	WHERE句は、SQL文が適用される行を絞り込むために使用されます。SELECT、UPDATE、DELETEなどと組み合わせて使います。
			47	SELECT 文で列名を指定する理由は何ですか？	実行速度が遅くなるから	特定の列だけを取得し、必要な情報を絞り込むため	SELECT * は文法エラーになるため	複数のテーブルを削除できるから	2	SELECT * では全ての列を取得しますが、列名を指定することで必要な情報だけを取得でき、パフォーマンスや可読性が向上します。
			48	LIMIT句を使うことで、どのような操作が可能になりますか？	検索結果の表示件数を制限する	検索結果から特定の列を除外する	テーブルに保存できる最大行数を設定する	クエリ実行時のパフォーマンスを保証する	1	LIMIT句は、SELECT文の結果として取得する行数を制限するために使われます。大量のデータの中から一部だけを取り出したいときに便利です。例えば SELECT * FROM users LIMIT 10; は最初の10件のみ取得します。
		テーブル設計と正規化	49	主キー（PRIMARY KEY）の役割を簡潔に説明してください。	各レコードを一意に識別するため	データの削除を防ぐため	データベースのバックアップのため	外部キーとの整合性を保つため	1	主キーは、テーブル内で各行（レコード）を一意に識別するためのカラムです。重複不可・NULL不可という制約があります。

			50	外部キー（FOREIGN KEY）は何のために使われますか？	データベースのサイズを制限する	別のテーブルとの関連性を定義する	主キーを削除するためのキー	クエリの高速化のために使われる	2	外部キーは、あるテーブルのカラムが他のテーブルの主キーを参照していることを表します。これにより、データの整合性を保てます。
			51	テーブル設計時に「重複データ」を避けるための手法は何ですか？	テーブルを1つにまとめる	正規化によりデータを分割し、重複を排除する	文字列をすべて大文字で保存する	WHERE句を多用する	2	正規化（Normalization）を行うことで、重複した情報を排除し、データの整合性や保守性を高めることができます。
			52	正規化の第一正規形（1NF）で求められる条件は何ですか？	テーブルに主キーがあること	すべての列が原子値（単一値）であること	列名がすべて英語であること	テーブルの行数が同じであること	2	第一正規形（1NF）では、各列に格納される値は単一の値（原子値）であることが求められます。例えば、カンマ区切りで複数値を入れるのはNGです。
			53	テーブルのリレーション設計で重要なことは何ですか？	すべてのテーブルを結合する	テーブルごとに主キーを別々にする	適切な外部キーでテーブル間の関係性を明確にする	すべてのカラムにインデックスを付ける	3	リレーショナルデータベースでは、外部キーを用いてテーブル間の関係性を正しく設計することが重要です。これにより、整合性が保たれます。
			54	同じ情報を複数のテーブルに繰り返し保存する場合のデメリットは何ですか？	データの不整合が起こるリスクがある	クエリが速くなる	ストレージ使用量が減る	主キーが不要になる	1	同じ情報が複数の場所に保存されると、片方だけ更新された場合などに矛盾（不整合）が生じやすくなります。これを防ぐために正規化が有効です。
		リレーションと結合	55	INNER JOIN と LEFT JOIN の違いを説明してください。	INNER JOIN は複数のテーブルを削除する結合方法	LEFT JOIN は条件に合うデータがない場合、NULLを返す	INNER JOIN は常にLEFT JOINよりも高速	LEFT JOIN は結合条件を無視する	2	INNER JOIN は両方のテーブルに共通するデータだけを取得し、LEFT JOIN は左側のテーブルのすべてのデータを取得し、右側に対応するデータがなければNULLを返す結合です。
			56	外部キーを使って2つのテーブルを結合する際の条件は何ですか？	片方のテーブルにしかデータが存在してはいけない	結合するカラムのデータ型が異なっている必要がある	一方のテーブルに主キーが存在しなくてもよい	外部キーと主キーが関連付けられている必要がある	4	外部キー（ foreign key ）は、別のテーブルの主キー（ primary key ）を参照することで、テーブル間の整合性を保ちます。結合するには、両者のデータ型と意味が一致している必要があります。
			57	同一テーブルを自己結合（SELF JOIN）するケースとはどのような場合ですか？	重複データを削除するとき	すべての行を1つの列に変換するとき	テーブルの削除と再作成を行うとき	テーブル内の親子関係を扱いたいとき	4	SELF JOIN（自己結合）は、同じテーブルを2回参照して親子関係や上下関係などを扱いたいときに使用します。たとえば、社員表の中で「上司と部下」を同じテーブル内で関連づける場合などに利用されます。
			58	テーブルを結合せずにサブクエリを使う利点は何ですか？	データベース設計を簡素化できる	サブクエリはSQL文の簡略化に不向きである	複数テーブルを同時に更新できる	条件に合致するデータを一時的に絞り込むことができる	4	サブクエリは、メインクエリの一部として実行され、絞り込みや集計などに使われます。結合を使わず、対象のデータを柔軟に選びたいときに便利です。
			59	GROUP BY 句を使用する際に必要な注意点は何ですか？	GROUP BY 句ではWHERE句を使ってはいけない	GROUP BY を使うとSQLのパフォーマンスが必ず落ちる	GROUP BY句を使うときは、集約関数かグループ化する列だけをSELECTする必要がある	GROUP BY を使うには、LIMIT句が必須である	3	GROUP BY 句を使うときは、グループ化されていない列を SELECT 句で指定する場合、それらを集約関数（ COUNT、SUM、AVG など）で包む必要があります。そうしないと文法エラーになります。
			60	テーブルの結合で生じる「カーディナリティ」の問題とは何ですか？	1対多の関係で想定以上に多くの結果が返ることがある	結合条件に誤りがあると全件削除される	すべての列が NULL になる	テーブルが自動的にインデックスされなくなる	1	「カーディナリティ」とは、テーブル間の結合における件数の関係性を指します。例えば、1対多の結合では、片方のレコードに対して複数のレコードが結びつくため、結合結果が膨れ上がることがあります。
	APIとHTTP通信の基礎	HTTPメソッドとステータスコード	61	リソースの作成に使用されるHTTPメソッドとして適切なものはどれですか？	GET	POST	DELETE	HEAD	2	POST メソッドは、新しいリソースをサーバー上に作成する目的で使用されます。GET はデータの取得、DELETE は削除、HEAD はヘッダー情報のみ取得するために使われます。
			62	GET メソッドと POST メソッドの主な違いを簡潔に説明してください。	GET はデータを更新し、POST はデータを取得する	GET はサーバーを変更し、POST は変更しない	POST は常により安全である	GET は URL にパラメータを含み、POST は本文にデータを送る	4	GETはクエリパラメータをURLに含めてリクエストを送るのに対し、POSTはリクエストボディ（本文）にデータを含めて送信します。これにより、POSTは機密性のあるデータにより適しています。
			63	クライアントのリクエストが正しく処理され、データが返されたときのステータスコードは何ですか？	200 OK	201 Created	301 Moved Permanently	400 Bad Request	1	ステータスコード 200 OK は、リクエストが成功し、レスポンスとしてデータが正常に返されたことを示します。201 Created はリソースの作成成功時に使われます。
			64	PUT と PATCH の違いを説明してください。	PUT は部分更新、PATCH は全体更新	PUT は新規作成専用、PATCH は削除専用	PUT は HTTP/1.0 で、PATCH は HTTP/2.0 専用	PUT は全体を上書き、PATCH は一部を更新	4	PUT はリソース全体を新しい内容で上書きするのに対し、PATCH はリソースの一部のみを変更するために使います。部分的な変更が必要なときには PATCHが適しています。
			65	ステータスコード「500 Internal Server Error」が示す状況は何ですか？	クライアントの入力ミス	認証に失敗した	サーバー側で予期せぬエラーが発生した	リソースが見つからなかった	3	500 Internal Server Error は、サーバー内部で処理中に予期しないエラーが発生したことを示すステータスコードです。原因はサーバー側にあります。
		JSONデータ形式とその活用	66	JSON形式で正しい構文の例として最も適切なものはどれですか？	{name: "Alice", age: 25}	["apple", "banana", "cherry"]	{"name": "Alice", "age": 25}	(name: "Alice", age: 25)	3	JSONでは文字列のキーと値はダブルクォーテーションで囲む必要があります。{"name": "Alice", "age": 25} の形式が正しい JSON 構文です。{name: "Alice", age: 25} や (name: "Alice", age: 25) は JavaScript や Python の形式であり、JSON としては不正です。
			67	JSONとXMLの違いを簡潔に説明してください。	JSON は軽量で読みやすく、XML は冗長だが柔軟性がある	JSON は構造を持たないが、XML は階層構造がある	XML はプログラミング言語であるが、JSON はデータ形式	JSON はバイナリ形式であり、XML はテキスト形式である	1	JSON はシンプルで読みやすく、データ通信に適しています。XML はタグベースで構造やメタ情報を多く含められる反面、冗長になりがちです。
			68	APIのレスポンスとしてJSONを使用する利点は何ですか？	データが暗号化される	人間に読みやすく、機械でも処理しやすい	HTML として直接表示できる	サーバーのセキュリティを強化できる	2	JSON は軽量なテキスト形式で、可読性が高く、さまざまな言語でパース（解析）しやすいため、API レスポンスの標準形式として広く利用されています。
			69	PythonでJSON文字列を辞書型に変換する関数は何ですか？	json.parse()	json.convert()	json.load()	json.loads()	4	Python では、文字列としてのJSONを辞書型に変換するには、json.loads() 関数を使います。ファイルから読み込む場合は json.load() を使用します。
			70	JSONで扱えるデータ型として正しいものを2つ挙げてください。	関数と正規表現	オブジェクトと配列	クラスとメソッド	モジュールと例外	2	JSON では、オブジェクト（連想配列・辞書）と配列（リスト）を扱えます。これに加えて、文字列・数値・真偽値・null もサポートされています。
		APIの設計とエラーハンドリング	71	RESTful APIにおいて、エンドポイントとは何を指しますか？	APIを呼び出すユーザーの端末	APIの仕様書に記載された使用ルール	サーバー上でAPIが応答するURLのこと	データベースのテーブル名そのもの	3	エンドポイントとは、APIで特定のリソースや機能にアクセスするためのURLのパス部分を指します。例えば/users や /products/123 などがそれにあたります。
			72	APIのバージョニング（例：/v1/ ）を行う主な理由は何ですか？	既存の利用者に影響を与えずに機能を更新するため	セキュリティ強化のため	URLを短くするため	APIを暗号化するため	1	APIにバージョン（例：/v1/, /v2/ ）を含めることで、既存のクライアントアプリケーションを壊さずにAPIの改良や仕様変更が可能になります。後方互換性を保つ目的です。
			73	APIからのレスポンスに含まれる「エラーメッセージ」はどのような目的で設計されるべきですか？	利用者が問題の原因を理解し、適切に修正できるようにする	サーバーのログ用に技術的な詳細をそのまま返す	ユーザーにできるだけ抽象的な表現を使う	エラーはレスポンスに含めないのが望ましい	1	エラーメッセージは、利用者（開発者など）がなぜリクエストが失敗したのかを理解し、修正の手がかりを得られるように設計されるべきです。ただし、過剰な内部情報はセキュリティリスクとなるため、バランスが重要です。
			74	不正なパラメータが送信された場合に返すべき適切なステータスコードは何ですか？	200 OK	401 Unauthorized	403 Forbidden	400 Bad Request	4	400 Bad Request は、リクエストの文法やパラメータが不正でサーバーが処理できない場合に返されます。クライアント側のミスに起因するエラーであり、正しいデータ形式で再送する必要があります。
	データ処理とセキュリティの基本	フォームと入力データの処理	75	Webフォームから送信されたデータをFlaskアプリケーションで受け取る方法として正しいものはどれですか？	request.form["username"] を使う	request.headers["username"] を使う	request.get("username") を使う	response.form["username"] を使う	1	フォームから POST リクエストで送信されたデータは、Flaskでは request.form["フィールド名"] を使って取得します。GET リクエストの場合は request.args を使います。
			76	フォームで送信されるデータが POST リクエストで送られる理由を説明してください。	URLにデータを含める必要があるため	サーバーにファイルを保存するため	データを非公開で送信するため	データがブラウザのキャッシュに残るため	3	POST リクエストは、データをHTTP メッセージのボディ部に含めて送信するため、URL に表示されません。そのため、パスワードや個人情報などを送信する際にも適しています。
			77	Flaskにおいて、フォームの入力値を辞書型のように扱えるオブジェクトはどれですか？	request.form	request.args	request.data	request.json	1	request.form は、POST リクエストで送られたフォームデータを格納するオブジェクトで、辞書のようにキーと値でアクセスできます。GET パラメータは request.args に格納されます。
		バリデーションと例外処理	78	入力バリデーションを行う主な目的は何ですか？	処理速度を上げるため	データベースを自動生成するため	不正なデータや予期しない値を排除するため	レイアウトを調整するため	3	バリデーションの目的は、入力値が正しい形式・範囲かを検証し、不正なデータによる処理ミスやセキュリティリスクを防ぐことにあります。
			79	Pythonにおいて try / except 文が用いられる主な理由は何ですか？	コードの動作を高速化するため	実行中に発生する例外を処理し、プログラムの異常終了を防ぐため	Webページのレイアウトを動的に変更するため	ユーザーインターフェースを作成するため	2	try / except は、実行中にエラー（例外）が発生する可能性のある処理を安全に実行するために使われます。例外を捕捉して、エラー内容を表示したり代替処理を行うことが可能です。
			80	バリデーションを行わずにデータを処理した場合に起こり得る問題を1つ挙げてください。	データが自動的に補正される	プログラムが正常に動作し続ける	処理速度が向上する	セキュリティリスクやデータ破損の可能性が高まる	4	バリデーションを怠ると、予期しないデータによってプログラムの誤動作やクラッシュ、さらにはSQLインジェクションなどのセキュリティリスクが発生する恐れがあります。