Python Revision Book

Comprehensive Practice with 20 Questions per Topic

Table of Contents

20. [Working with Dates and Time](#working-with-dates-and-time)

---

1. Variables and Data Types

Concepts: Variables, integers, floats, strings, booleans, type() function.

1. Declare a variable `name` and assign it your name as a string.

2. Assign the value `25` to a variable `age`. What is its data type?

3. Create a variable `price` with value `19.99`. Print its type.

4. Convert the integer `10` to a float and store it in a variable `num`.

5. Convert the string `"True"` to a boolean.

6. Is `None` a data type? Show an example.

7. Assign `3 + 4j` to a variable. What type is it?

8. What will `type(0)` return?

9. Can you use a number as the first character in a variable name? Why?

10. Correct this: `1var = "hello"`.

11. What is dynamic typing in Python?

12. What does `isinstance(5, int)` return?

13. Assign multiple variables in one line: `a=1`, `b=2`, `c=3`.

14. Swap two variables without a temporary variable.

15. What is the difference between `=` and `==`?

16. How do you check the memory address of a variable?

17. What happens when you reassign a variable?

18. Can a variable change types during execution?

19. What is the output of `print(type("123"))`?

20. Write code to print the data type of `[]`.

---

2. Operators

Concepts: Arithmetic, comparison, logical, assignment, identity, membership.

1. What is the result of `10 // 3`?

2. Evaluate: `5 ** 3`.

3. What does `%` operator do? Give an example.

4. Compare 10 and 20 using `>` and print the result.

5. Use `and` to check if `x > 5` and `x < 15`.

6. What is the output of `not True`?

7. What does `!=` mean?

8. Use `+=` to increase a variable by 5.

9. Check if `"apple"` is in the list `["banana", "apple", "cherry"]`.

10. Are `[1,2,3]` and `[1,2,3]` the same object? Use `is` to check.

11. What is the order of operations for `3 + 4 * 2`?

12. Evaluate: `(True or False) and not False`.

13. What is the result of `"hello" + "world"`?

14. Multiply a string: `"hi" * 3`.

15. What does `is not` do?

16. Use `in` to check if `'a'` is in `"cat"`.

17. What is the output of `10 == "10"`?

18. Calculate the remainder when 27 is divided by 4.

19. Combine conditions: `x = 10`; check if `x` is even and greater than 5.

20. What is the result of `bool("")`?

---

3. Input and Output

Concepts: `input()`, `print()`, formatting.

1. Ask the user for their name and print it.

2. Get a number from the user and double it.

3. Print “Hello World” on two separate lines.

4. Format output: “My name is Alice and I am 25 years old.”

5. Use f-strings to print a variable `score = 95`.

6. Print numbers 1 to 5 separated by commas.

7. Read two numbers and print their sum.

8. Ask for age and print whether they are adult (>=18).

9. Print with no newline: `print("Hello", end=" ")`.

10. Use `.format()` to insert values into “I have {} apples.”.

11. Prompt user to enter a float and convert it.

12. Print a floating point number rounded to 2 decimals.

13. Print a table header using tabs (`\t`).

14. Ask for favorite color and print in uppercase.

15. Read a sentence and count how many words it has.

16. Print a backslash: `\`.

17. Print quotes inside a string: He said "Hi!".

18. Use triple quotes to print a multi-line message.

19. Get input and check if it's empty.

20. Print a number with leading zeros (e.g., 5 as "005").

---

4. Conditional Statements

Concepts: `if`, `elif`, `else`, nested conditions.

1. Check if a number is positive, negative, or zero.

2. Determine if a year is a leap year.

3. Grade a score: A (>=90), B (80-89), etc.

4. Check if a number is even or odd.

5. Find the largest of three numbers.

6. Check if a person can vote (age >= 18).

7. Validate a password length (must be >=8).

8. Categorize temperature: cold (<0), mild (0-20), hot (>20).

9. Check if a character is a vowel.

10. Determine if a triangle is equilateral, isosceles, or scalene.

11. Simulate a login: correct username and password.

12. Check if a number is divisible by both 3 and 5.

13. Classify a person as child, teen, adult, senior.

14. Check if a string is empty.

15. Test if a number is between 1 and 100 inclusive.

16. Nested condition: If it's weekend and sunny → go out.

17. Check if a letter is uppercase.

18. Determine shipping cost based on weight.

19. Check if a number is within 10 of 100.

20. Validate a menu choice (1-4).

---

5. Loops

Concepts: `for`, `while`, `break`, `continue`, `range()`.

1. Print numbers 1 to 10 using `for`.

2. Print even numbers from 1 to 20.

3. Sum all numbers from 1 to 100.

4. Print multiplication table of 5 (1 to 10).

5. Reverse a list using a loop.

6. Count down from 10 to 1.

7. Keep asking for input until user enters 'quit'.

8. Skip printing number 5 in a loop from 1 to 10.

9. Print each character in a string.

10. Find the factorial of 6 using a loop.

11. Print only vowels in a given string.

12. Search for a number in a list; break when found.

13. Use `continue` to skip negative numbers.

14. Generate Fibonacci sequence up to 50.

15. Print squares of numbers 1 to 5.

16. Count how many times a letter appears in a string.

17. Print a right triangle of stars (height 5).

18. Use `while True` with `break` to simulate menu.

19. Print numbers divisible by 3 from 1 to 30.

20. Loop through dictionary keys and values.

---

6. Functions

Concepts: Defining, calling, parameters, return, scope.

1. Define a function `greet()` that prints "Hello!".

2. Create `add(a, b)` and return the sum.

3. Write `is_even(n)` returning True if even.

4. Define `max_of_two(x, y)`.

5. Function to calculate area of a circle.

6. Write a function with default parameter `greeting="Hi"`.

7. Create a function that returns multiple values.

8. Use keyword arguments in a function call.

9. Explain local vs global scope with example.

10. Write a function that counts vowels in a string.

11. Recursive countdown function (print n to 1).

12. Function to reverse a string.

13. Accept variable number of arguments (`*args`).

14. Use `**kwargs` to handle named arguments.

15. Function to check palindrome.

16. Closure example: make a counter function.

17. Lambda inside a function.

18. Function that modifies a global variable.

19. Function to generate random password.

20. Decorator concept: write a simple timing decorator.

---

7. Strings

Concepts: Indexing, slicing, methods, immutability.

1. Access the first character of `"Python"`.

2. Slice `"Hello"` to get `"ell"`.

3. Convert `"hello"` to uppercase.

4. Check if `"world"` is in `"Hello world"`.

5. Replace `"cat"` with `"dog"` in a string.

6. Split `"apple,banana,cherry"` by comma.

7. Join a list `['a','b','c']` into a string.

8. Strip whitespace from `"  hello  "`.

9. Find index of `'o'` in `"Python"`.

10. Count occurrences of `'l'` in `"Hello"`.

11. Check if string starts with `"Py"`.

12. Make a string title case.

13. Format string using `.format()` with two values.

14. Use f-string to include a variable.

15. Reverse a string using slicing.

16. Check if string is numeric.

17. Remove vowels from a string.

18. Capitalize first letter of each word.

19. Pad a string with zeros to length 5.

20. Check if two strings are anagrams.

---

8. Lists

Concepts: Creation, indexing, methods, mutability.

1. Create a list of 5 fruits.

2. Access the third element.

3. Change the second item.

4. Add an item to the end.

5. Insert "mango" at index 1.

6. Remove "banana" from the list.

7. Pop the last element and print it.

8. Find index of "apple".

9. Sort the list alphabetically.

10. Reverse the list.

11. Count how many times "apple" appears.

12. Extend list with another list.

13. Copy a list safely.

14. Clear all elements.

15. Check if "grape" is in the list.

16. Slice the list to get first 3 items.

17. Create a list of squares from 1 to 5.

18. Use list comprehension to get even numbers.

19. Find max and min in a number list.

20. Flatten a 2D list: `[[1,2],[3,4]]`.

---

9. Tuples

Concepts: Immutable sequences, packing, unpacking.

1. Create a tuple with three colors.

2. Access the first element.

3. Try to modify a tuple element — what happens?

4. Unpack a tuple into three variables.

5. Concatenate two tuples.

6. Convert a list to a tuple.

7. Find length of a tuple.

8. Check if "red" is in the tuple.

9. Count occurrences of a value.

10. Use tuple as a dictionary key.

11. Create a single-element tuple.

12. Slice a tuple to get last two elements.

13. Swap two variables using tuple unpacking.

14. Return multiple values from a function using tuple.

15. Iterate over a tuple.

16. Find max in a tuple of numbers.

17. Use `*` in unpacking: `a, *b = (1,2,3,4)`.

18. When would you prefer tuple over list?

19. Check if a tuple is empty.

20. Nest tuples: `((1,2), (3,4))`.

---

10. Dictionaries

Concepts: Key-value pairs, methods, iteration.

1. Create a dict with name, age, city.

2. Access the value of "name".

3. Add a new key "job".

4. Update the value of "age".

5. Remove "city" using `pop()`.

6. Check if "name" exists in dict.

7. Get all keys.

8. Get all values.

9. Get key-value pairs.

10. Use `.get()` to access a key safely.

11. Merge two dictionaries.

12. Create dict from two lists using `zip()`.

13. Count frequency of letters in a string.

14. Iterate over keys and print them.

15. Use dict comprehension: `{x: x**2 for x in range(5)}`.

16. Clear all entries.

17. Copy a dictionary.

18. Use `setdefault()` to add default value.

19. Check if two dicts are equal.

20. Invert a dictionary (swap keys and values).

---

11. Sets

Concepts: Unordered, unique elements, set operations.

1. Create a set with numbers 1,2,3.

2. Add 4 to the set.

3. Remove 2.

4. Discard 5 (even if not present).

5. Check if 3 is in the set.

6. Union of two sets.

7. Intersection of two sets.

8. Difference: set1 - set2.

9. Symmetric difference.

10. Check if set1 is subset of set2.

11. Convert a list to a set to remove duplicates.

12. Find common elements in two lists using sets.

13. Clear all elements.

14. Pop an arbitrary element.

15. Create empty set (correct way).

16. Use set comprehension: even numbers from 0 to 10.

17. Check if two sets are disjoint.

18. Update set with another set.

19. Length of a set.

20. Use sets to find unique words in a sentence.

---

12. File Handling

Concepts: Open, read, write, close, modes.

1. Open and read a text file.

2. Read all lines into a list.

3. Read file line by line.

4. Write "Hello World" to a file.

5. Append text to existing file.

6. Use `with` statement to open a file.

7. Handle `FileNotFoundError`.

8. Copy content from one file to another.

9. Count number of lines in a file.

10. Count words in a file.

11. Write a list of strings to a file (each on new line).

12. Check if a file exists.

13. Delete a file.

14. Rename a file.

15. Read CSV-like data (comma-separated).

16. Write data in JSON format (using `json` module).

17. Read binary file (e.g., image).

18. Use different encodings (e.g., utf-8).

19. Lock a file for writing (conceptual).

20. Log messages to a file.

---

13. Exception Handling

Concepts: `try`, `except`, `finally`, `raise`.

1. Handle division by zero.

2. Catch `IndexError` when accessing invalid list index.

3. Use `else` block in try-except.

4. Use `finally` to clean up resources.

5. Raise a `ValueError` manually.

6. Handle multiple exceptions.

7. Catch all exceptions (generic).

8. Create custom exception class.

9. Re-raise an exception.

10. Validate input in a loop with exceptions.

11. Handle `KeyError` when accessing dict.

12. Use assertions: `assert x > 0`.

13. Open file with error handling.

14. Convert string to int with try-except.

15. Prevent crash when user enters non-number.

16. Handle `ImportError`.

17. Use context manager in exception safety.

18. Log errors to a file.

19. Retry logic on failure.

20. Validate age input with custom exception.

---

14. Object-Oriented Programming (OOP)

Concepts: Classes, objects, inheritance, encapsulation, polymorphism.

1. Define a `Person` class with `name` and `age`.

2. Create an instance of `Person`.

3. Add a method `greet()` to `Person`.

4. Use `__init__` constructor.

5. Add private attribute (name mangling).

6. Use property decorator for getter/setter.

7. Create a `Student` class inheriting `Person`.

8. Override a method in child class.

9. Use `super()` in constructor.

10. Demonstrate polymorphism with `make_sound()` in `Animal`, `Dog`, `Cat`.

11. Static method in a class.

12. Class method using `@classmethod`.

13. Class variable vs instance variable.

14. Multiple inheritance example.

15. Abstract base class (use `abc` module).

16. Method overloading (simulate with default args).

17. `__str__` method for string representation.

18. `__len__` method for custom length.

19. Encapsulation: restrict direct access.

20. Composition: `Car` has an `Engine`.

---

15. Modules and Packages

Concepts: Import, create modules, `__init__.py`.

1. Import `math` and use `sqrt()`.

2. Import only `pi` from `math`.

3. Import with alias: `import numpy as np`.

4. Create your own module `utils.py` with a function.

5. Import and use function from your module.

6. Use `from module import *` (explain caution).

7. Reload a module (using `importlib.reload`).

8. List all functions in a module using `dir()`.

9. Check module file location.

10. Create a package with `__init__.py`.

11. Import from subpackage.

12. Use `if __name__ == "__main__":`.

13. Install external package using pip.

14. Use `datetime` module to get current time.

15. Import and use `random.randint()`.

16. Use `os` module to list directory contents.

17. Use `sys` to access command-line arguments.

18. Create virtual environment.

19. Use `json` module to parse string.

20. Import your own package from another directory.

---

16. Regular Expressions

Concepts: `re` module, patterns, matching.

1. Check if string contains digits.

2. Match email pattern.

3. Find all numbers in a string.

4. Replace all whitespace with underscore.

5. Validate phone number (e.g., 123-456-7890).

6. Split string by non-word characters.

7. Check if string starts with "Hello".

8. Extract domain from email.

9. Use `re.search()` vs `re.match()`.

10. Case-insensitive search.

11. Find words with exactly 5 letters.

12. Validate IPv4 address.

13. Escape special characters.

14. Use groups to extract parts.

15. Substitute with captured group.

16. Check password strength (length, digit, symbol).

17. Remove HTML tags.

18. Find repeated words.

19. Validate URL.

20. Extract dates in YYYY-MM-DD format.

---

17. List Comprehensions

Concepts: Concise list creation.

1. Create list of squares from 1 to 10.

2. Get even numbers from 1 to 20.

3. Convert list of strings to uppercase.

4. Extract digits from a string.

5. Flatten a 2D list.

6. Create list of tuples `(x, x**2)` for x in 1-5.

7. Filter names starting with 'A'.

8. Double each number in a list.

9. Exclude negative numbers.

10. Create list of lengths of words.

11. Nested list comprehension: multiplication table.

12. Use conditionals: `'even' if x%2==0 else 'odd'`.

13. Generate Pythagorean triples (up to 20).

14. Remove vowels from list of characters.

15. Square only positive numbers.

16. Create list of prime numbers (up to 30).

17. Extract keys where value > 10.

18. Pair elements from two lists.

19. Reverse each string in a list.

20. Generate combinations of two lists.

---

18. Lambda Functions

Concepts: Anonymous functions, `map`, `filter`, `sorted`.

1. Lambda to add two numbers.

2. Use `map()` to square all numbers in a list.

3. Use `filter()` to keep even numbers.

4. Sort list of tuples by second element.

5. Lambda with no arguments.

6. Lambda with conditional expression.

7. Use lambda in `key` parameter of `max()`.

8. Lambda to check if string is palindrome.

9. Use `reduce()` to multiply all numbers.

10. Lambda with `*args`.

11. Assign lambda to variable.

12. Use lambda in list comprehension (rare but possible).

13. Pass lambda to higher-order function.

14. Lambda for string formatting.

15. Sort strings by length using lambda.

16. Filter names longer than 5 chars.

17. Map to convert Celsius to Fahrenheit.

18. Use lambda with `sorted()` on dictionary items.

19. Lambda to compute distance from origin.

20. Use lambda in event handling (simulated).

---

19. Recursion

Concepts: Function calling itself, base case.

1. Factorial using recursion.

2. Fibonacci sequence recursively.

3. Sum of numbers from 1 to n.

4. Power function: `power(base, exp)`.

5. Reverse a string recursively.

6. Check if string is palindrome recursively.

7. Binary search using recursion.

8. Greatest common divisor (GCD) with Euclid’s algorithm.

9. Tower of Hanoi (print moves).

10. Count digits in a number.

11. Sum digits of a number.

12. Recursive linear search.

13. Flatten nested list recursively.

14. Tree traversal (simple structure).

15. Ackermann function (advanced).

16. Recursive countdown.

17. Generate permutations of a string.

18. Draw fractal pattern (conceptual).

19. Recursive exponentiation (fast power).

20. Memoization: cache recursive Fibonacci.

---

20. Working with Dates and Time

Concepts: `datetime`, `date`, `time`, `timedelta`.

1. Get current date and time.

2. Create a specific date: Jan 1, 2023.

3. Format date as "DD/MM/YYYY".

4. Add 5 days to current date.

5. Calculate difference between two dates.

6. Parse string "2023-12-25" to date.

7. Get day of week (Monday=0).

8. Sleep program for 3 seconds.

9. Measure execution time of a function.

10. Get current timestamp.

11. Convert date to Unix timestamp.

12. Schedule a task (conceptual).

13. Check if date is weekend.

14. Find next Monday.

15. Birthday calculator: days until next birthday.

16. Age in years from birth date.

17. Use `strftime()` to format time.

18. Use `strptime()` to parse time string.

19. Work with time zones (using `pytz` or `zoneinfo`).

20. Countdown timer to New Year.

✅ **Answer Keys Available Upon Request**

This revision book provides **400 practice questions** across core Python topics. Ideal for beginners to intermediate learners preparing for exams or interviews.

Let me know if you'd like:

- Full answer key

- PDF version

- Jupyter Notebook format

- Flashcards

- Interactive quiz mode

Happy coding! 🐍