1. Python Variable Scope in Functions:

Python handles variable scope by differentiating between **local** and **global** variables. A local variable is only accessible within the function, while a global variable can be accessed anywhere in the code.

Code Example:

```
python
Copy code
x = 10  # Global variable
def my_function():
    x = 5  # Local variable
    print("Inside function:", x)

my_function()print("Outside function:", x)
```

Output:

```
bash
Copy code
Inside function: 5
Outside function: 10
```

2. Python Built-in Data Types & Type Conversion:

- **int**: `x = 5`
- **float**: `y = 3.14`
- **str**: `s = "Hello"`
- **list**: `lst = [1, 2, 3]`
- **tuple**: `tup = (1, 2, 3)`
- **dict**: `d = {'a': 1, 'b': 2}`
- **set**: `st = {1, 2, 3}`

Type Conversion Example:

```
python
Copy code
x = 10
y = float(x)  # int to float
z = str(y)    # float to string
```

3. Shallow Copy vs. Deep Copy:

Shallow copy copies references, while deep copy duplicates everything.

Code Example:

```
python
Copy code
import copy

lst = [[1, 2], [3, 4]]
shallow = copy.copy(lst)
deep = copy.deepcopy(lst)

lst[0][0] = 99print("Shallow:", shallow)print("Deep:", deep)
```

4. Difference Between Lists and Tuples:

Lists are mutable, and tuples are immutable.

Code Example:

```
python
Copy code
lst = [1, 2, 3]
tup = (1, 2, 3)

lst[0] = 10  # Modifying list# tup[0] = 10  # This will raise an
error
```

5. Handling Exceptions:

Python uses `try-except` blocks for exception handling.

Code Example:

```
python
Copy code
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Cannot divide by zero!"
print(divide(10, 0))
```

6. List Comprehensions:

Code Example:

```
python
Copy code
# Traditional for loop
squares = []for x in range(5):
    squares.append(x**2)
# List comprehension
squares_comp = [x**2 for x in range(5)]
```

7. Sets and Set Operations:

Sets are unordered and don't allow duplicates, while lists do.

Code Example:

```
python
Copy code
a = {1, 2, 3}
b = {2, 3, 4}print("Union:", a | b)print("Intersection:", a &
b)print("Difference:", a - b)
```

8. Lambda Function:

Lambda functions are anonymous functions.

Code Example:

```
python
Copy code
tuples = [(1, 2), (3, 1), (5, 0)]
sorted_tuples = sorted(tuples, key=lambda x: x[1])print(sorted_tuples)
```

9. filter(), map(), reduce():

Code Example:

```
python
Copy code
from functools import reduce

nums = [1, 2, 3, 4]
# filter: keeps even numbers
evens = list(filter(lambda x: x % 2 == 0, nums))
# map: squares numbers
squares = list(map(lambda x: x**2, nums))
# reduce: sum of numbers
total = reduce(lambda x, y: x + y, nums)
```

10. Working with Files:

Code Example:

```
python
Copy code
with open("example.txt", "w") as f:
    f.write("Hello, world!")
with open("example.txt", "r") as f:
    content = f.read()
    print(content)
```

11. Naming Identifiers:

- Valid: `my_var`, `_hidden`, `value123`
- Invalid: `123abc`, `my-var`, `@var`

Identifiers must not start with a digit and should not include special characters.

12. `assert` Keyword for Debugging:

Code Example:

```
python
Copy code
def divide(a, b):
    assert b != 0, "Division by zero!"
    return a / b
print(divide(10, 2))
```

13. Arithmetic and Logical Operators:

Code Example:

```
python
Copy code
# Arithmeticprint(10 + 5)print(10 - 5)
# Logicalprint(10 > 5 and 5 < 10)print(10 < 5 or 5 < 10)
```

14. List Operations:

Code Example:

```
python
Copy code
lst = [1, 2, 3]
lst.append(4)     # Add
lst.remove(2)     # Remove
slice_lst = lst[1:3]  # Slicing
```

15. Tuples:

Code Example:

```
python
Copy code
single = (1,)  # Single element tuple
multiple = (1, 2, 3)  # Multiple element tuple
```

16. Dictionaries (Key-Value Pairs):

Code Example:

```
python
Copy code
d = {'a': 1, 'b': 2}
d['c'] = 3        # Add
d['a'] = 10       # Updatedel d['b']        # Remove
```

17. Mutable vs. Immutable:

Lists are mutable; tuples are immutable.

Code Example:

```
python
Copy code
lst = [1, 2, 3]
tup = (1, 2, 3)
lst[0] = 10  # Mutable# tup[0] = 10  # Immutable: Raises error
```

18. Nested Data Structures:

Code Example:

```
python
Copy code
nested = {'a': [1, 2, {'b': 10}]}print(nested['a'][2]['b'])  #
Accessing nested values
```

19. Python Decorators:

Code Example:

```
python
Copy code
def decorator(func):
    def wrapper():
        print("Before function call")
        func()
        print("After function call")
    return wrapper
@decoratordef say_hello():
    print("Hello!")

say_hello()
```

20. Generator Functions:

Generators yield values one at a time.

Code Example:

```
python
Copy code
def fibonacci():
    a, b = 0, 1
    for _ in range(10):
        yield a
        a, b = b, a + b
for num in fibonacci():
    print(num)
```