Table of contents

1

-
- try-except
- finally else
-
-

1.1

Python

```
#
number = int("hello")  # ValueError: invalid literal
```

1.2

1.2.1

```
#
if x > 5
    print("5   ")

#
print("Hello"
```

1.2.2

```
#
result = 10 / 0

#
numbers = [1, 2, 3]
print(numbers[5])

#
int("hello")
```

1.3

```
# ValueError:
int("abc")
float("hello")

# TypeError:
"hello" + 5
len(42)

# IndexError:
my_list = [1, 2, 3]
my_list[10]

# KeyError:
my_dict = {"name": " "}
my_dict["age"]

# FileNotFoundError:
open("nonexistent.txt")
```

1.4 try-except

1.4.1

```
try:
    #
    risky_code()
except ExceptionType:
    #
    handle_error()
```

1.4.2

```
try:
    number = int(input("      : "))
    result = 10 / number
    print(f" : {result}")
except ValueError:
    print("        ")
except ZeroDivisionError:
    print("        ")
```

1.5

1.5.1 except

```
try:
    age = int(input("      : "))
    category = determine_category(age)
    print(f"  {category}    ")
except ValueError:
    print("        ")
except TypeError:
    print("        ")
except Exception as e:
    print(f"        : {e}")
```

1.5.2

```
try:
    #
    process_data()
except (ValueError, TypeError, IndexError):
    print("         ")
```

1.6 else finally

1.6.1 else

```
try:
    number = int(input("      : "))
except ValueError:
    print("     ")
else:
    print(f"    : {number}")
    #
```

1.6.2 finally

```
try:
    file = open("data.txt", "r")
    content = file.read()
except FileNotFoundError:
    print("       ")
finally:
    #       -
    if 'file' in locals():
        file.close()
```

1.7

1.7.1

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"    : {type(e).__name__}")
    print(f"     : {e}")
    print("        ")
```

1.7.2

```
try:
    #
    risky_operation()
except Exception as e:
    print(f"     : {e}")
    #
    import traceback
    traceback.print_exc()
```

1.8

1.8.1 raise

```
def validate_age(age):
    if age < 0:
        raise ValueError("       ")
    if age > 150:
        raise ValueError("     ")
    return True

try:
    age = -5
    validate_age(age)
```

```
except ValueError as e:
    print(f"   : {e}")
```

1.8.2

```
try:
    process_data()
except ValueError:
    print("     ...")
    raise  #
```

1.9

1.9.1

```
class InvalidEmailError(Exception):
    """           """
    pass

class PasswordTooWeakError(Exception):
    """           """
    def __init__(self, message, min_length=8):
        self.message = message
        self.min_length = min_length
        super().__init__(self.message)

def validate_password(password):
    if len(password) < 8:
        raise PasswordTooWeakError(
            "      ",
            min_length=8
        )
```

1.10

1.10.1

```
def read_file_safely(filename):
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            return file.read()
    except FileNotFoundError:
        print(f"  '{filename}'     ")
        return None
    except PermissionError:
        print(f"  '{filename}'      ")
        return None
    except Exception as e:
        print(f"      : {e}")
        return None

content = read_file_safely("data.txt")
if content:
    print("         ")
```

1.11

1.11.1

```
def get_integer_input(prompt, min_val=None, max_val=None):
    while True:
        try:
            value = int(input(prompt))

            if min_val is not None and value < min_val:
                print(f"  {min_val}     ")
                continue

            if max_val is not None and value > max_val:
                print(f"  {max_val}     ")
                continue
```

```
            return value

        except ValueError:
            print("        ")
        except KeyboardInterrupt:
            print("\n            ")
            return None

age = get_integer_input("       (0-120): ", 0, 120)
```

1.12

1.12.1 print

```
def calculate_average(numbers):
    print(f"  :    = {numbers}")  #

    if not numbers:
        print("  :        ")     #
        return 0

    total = sum(numbers)
    count = len(numbers)
    average = total / count

    print(f"  : total={total}, count={count}")  #
    return average
```

1.12.2 assert

```
def divide_numbers(a, b):
    assert b != 0, "        "
    assert isinstance(a, (int, float)), "         "
    assert isinstance(b, (int, float)), "         "

    return a / b
```

1.13

```
class Calculator:
    def safe_divide(self, a, b):
        try:
            #
            if not isinstance(a, (int, float)):
                raise TypeError("         ")
            if not isinstance(b, (int, float)):
                raise TypeError("         ")
            if b == 0:
                raise ZeroDivisionError("       ")

            result = a / b
            return {"success": True, "result": result}

        except (TypeError, ZeroDivisionError) as e:
            return {"success": False, "error": str(e)}
        except Exception as e:
            return {"success": False, "error": f"    : {e}"}

#
calc = Calculator()
result = calc.safe_divide(10, 2)
if result["success"]:
    print(f" : {result['result']}")
else:
    print(f"  : {result['error']}")
```

1.13.1 1

```
def safe_int_convert(value):
    """

      : (success: bool, result: int    error_message: str)
    """
```

```
    #
    pass

#
print(safe_int_convert("123"))      #
print(safe_int_convert("12.34"))    #
print(safe_int_convert("hello"))    #
```

1.13.2 2

```
def safe_list_access(my_list, index):
    """


    """
    #
    pass
```

1.13.1 1

```
def safe_int_convert(value):
    try:
        #
        return True, int(value)
    except ValueError:
        try:
            #
            return True, int(float(value))
        except ValueError:
            return False, f"'{value}'      "
    except Exception as e:
        return False, f"    : {e}"
```

1.13.2 2

```
def safe_list_access(my_list, index):
    try:
        return my_list[index]
    except IndexError:
        return f"    {index}    "
    except TypeError:
        return "        "
```

1.14

1. : Exception
2. :
3. :
4. :
5. :
6. : finally with
7. :

1.15

1.15.1 with

```
#
try:
    file = open("data.txt", "r")
    content = file.read()
    #
except FileNotFoundError:
    print("       ")
finally:
    file.close()  #

#
try:
    with open("data.txt", "r") as file:
        content = file.read()
```

```
        # 'with'
except FileNotFoundError:
    print("       ")
```

1.16

1.16.1

```
#   :
try:
    risky_operation()
except:
    pass  #

#   :
try:
    specific_operation()
except Exception:
    print("     ")  #

#   :
try:
    value = my_dict["key"]
except KeyError:
    value = "default"  #    .get()
```

1.17

- -
- try-except
-
- finally
-
-
- print

1.18

1.
2. `finally`
3.
4. `with`
5. 3