Table of contents

1

1.1

-
-
- Python
-

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

1.2

  :

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

#
print("Hello"
```

     :

```
#
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 = {" ": " "}
my_dict[" "]

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

1.4 try-except

:

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

:

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

```
    print("        ")
except ZeroDivisionError:
    print("        ")
```

1.5

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}")
```

:

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

1.6 else finally

else:

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

finally:

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

1.7

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

#
import traceback

try:
    risky_operation()
except Exception as e:
    print("     :")
    print(traceback.format_exc())
```

1.8

```
class CustomError(Exception):
    """     """
    def __init__(self, message, error_code=None):
        super().__init__(message)
        self.error_code = error_code
```

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

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

try:
    validate_age(-5)
except AgeError as e:
    print(f"   : {e}")
```

1.9

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

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

1.10 assert

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

    return a / b

#
try:
    result = divide(10, "2")  # AssertionError
except AssertionError as e:
    print(f"     : {e}")

#     assert
# python -O script.py
```

1.11

```
import logging

#
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

def process_user_data(data):
    try:
        #
        result = complex_calculation(data)
        logging.info("        ")
        return result
    except ValueError as e:
        logging.error(f"    : {e}")
        raise
    except Exception as e:
```

```
        logging.critical(f"     : {e}")
        raise

#
try:
    result = process_user_data([1, 2, 3])
except Exception:
    logging.error("       ")
```

1.12

1.13 :

```
class Calculator:
    def __init__(self):
        self.history = []

    def calculate(self, expression):
        try:
            #    eval
            allowed_chars = set('0123456789+-*/(). ')
            if not all(c in allowed_chars for c in expression):
                raise ValueError("       ")

            result = eval(expression)
            self.history.append(f"{expression} = {result}")
            return result

        except ZeroDivisionError:
            return "  :   "
        except ValueError as e:
            return f"  : {e}"
        except SyntaxError:
            return "  :   "

#
calc = Calculator()
expressions = ["2+2", "10/0", "2*invalid", "3**2"]
```

```
for expr in expressions:
    result = calc.calculate(expr)
    print(f"{expr} → {result}")
```

1.14 :

```
import json

class DataManager:
    def __init__(self, filename):
        self.filename = filename
        self.data = self.load_data()

    def load_data(self):
        try:
            with open(self.filename, 'r', encoding='utf-8') as file:
                return json.load(file)
        except FileNotFoundError:
            print("                ")
            return {}
        except json.JSONDecodeError:
            print("                  ")
            return {}

    def save_data(self):
        try:
            with open(self.filename, 'w', encoding='utf-8') as file:
                json.dump(self.data, file, ensure_ascii=False, indent=2)
            return True
        except PermissionError:
            print("          ")
            return False
        except Exception as e:
            print(f"       : {e}")
            return False

#
manager = DataManager("user_data.json")
manager.data["users"] = [" ", " "]
```

```
if manager.save_data():
    print("        ")
```

1.15

1. - Exception
2. -
3. -
4. - finally with
5. -

1.16

- ( )
- ( )
- ( )
- ( )

1.17

1.17.1

: - Python.org - - Real Python - - Python Tutor

1.18

:

:

|

: