Conditional Statements are used to execute a block of code based on certain conditions. These conditions typically evaluate to either True or False, and depending on the result, specific code blocks can be executed.

```
# The if statement is the most basic form of conditional statement in Python. It allows you to execute a block of code only if
age = 18
if age >= 18:
  print("You are eligible to vote")
```

```
You are eligible to vote
```

```
# The else statement is used to define an alternative block of code that runs when the if condition is not met.
age = 16
if age >= 18:
    print("You are eligible to vote")
else:
    print("You are not eligible to vote")
```

```
You are not eligible to vote
```

```
# If-elif-else Conditional Statements in Python
# The if statements are executed from the top down.
# As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest
# If none of the conditions is true, then the final “else” statement will be executed.

marks = 99

if marks < 35:
  print("Grade: F")
elif marks >= 90:
    print("Grade: A")
elif marks >= 80:
    print("Grade: B")
elif marks >= 70:
    print("Grade: C")
else:
    print("Grade: D")
```

```
Grade: A
```

```
# Nested if Statements
# You can nest if, elif, and else statements inside each other. This is useful for checking conditions within other conditions.

age = 20
is_registered = False

if age >= 18:
    if is_registered:
        print("You are eligible to vote")
    else:
        print("You need to register to vote")
else:
    print("You are not eligible to vote")
```

```
You need to register to vote
```

```
# The Python ternary Expression determines if a condition is true or false and then returns the appropriate value in accordance
# The ternary Expression is useful in cases where we need to assign a value to a variable based on a simple condition, and we w
# Syntax: [on_true] if [expression] else [on_false]
# expression: conditional_expression | lambda_expr


age = 2
print("Eligible to vote" if age >= 18 else "Not eligible to vote")
```

```
Not eligible to vote
```

```
# [on_true] if [condition] else [on_false]
# lambda arguments: on_true if condition else on_false

check_even = lambda x: "Even" if x % 2 == 0 else "Odd"
```

```
print(check_even(10))  # Output: Even
print(check_even(7))   # Output: Odd
```

```
Even
Odd
```

```
score = 99

grade = 'F' if score <= 35 else ('A' if score >= 90 else ('B' if score >= 80 else ('C' if score >= 70 else 'D')))
print(grade)
```

```
A
```

```
# match subject:
#     case pattern1:
#         # Code block if pattern1 matches
#     case pattern2:
#         # Code block if pattern2 matches
#     case _:
#         # Default case (wildcard) if no other pattern matches

def check_number(x):
    match x:
        case 10:
            print("It's 10")
        case 20:
            print("It's 20")
        case _:
            print("It's neither 10 nor 20")

check_number(10)
check_number(30)
```

```
It's 10
It's neither 10 nor 20
```

```
def num_check(x):
    match x:
        case 10 | 20 | 30:  # Matches 10, 20, or 30
            print(f"Matched: {x}")
        case _:
            print("No match found")

num_check(10)
num_check(20)
num_check(25)
```

```
Matched: 10
Matched: 20
No match found
```

```
def num_check(x):
    match x:
        case 10 if x % 2 == 0:  # Match 10 only if it's even
            print("Matched 10 and it's even!")
        case 10:
            print("Matched 10, but it's not even.")
        case _:
            print("No match found")

num_check(10)
num_check(15)
```

```
Matched 10 and it's even!
No match found
```

```
def process(data):
    match data:
        case [x, y]:
            # A list with two elements
            print(f"Two-element list: {x}, {y}")
        case [x, y, z]:
            # A list with three elements
            print(f"Three-element list: {x}, {y}, {z}")
        case _:
            print("Unknown data format")

process([1, 2])
process([1, 2, 3])
process([1, 2, 3, 4])
```

```
Two-element list: 1, 2
Three-element list: 1, 2, 3
Unknown data format
```

```
def person(person):
    match person:

        # Dictionary with name and age keys
        case {"name": name, "age": age}:
            print(f"Name: {name}, Age: {age}")

        # Dictionary with only name key
        case {"name": name}:
            print(f"Name: {name}")
        case _:
            print("Unknown format")

person({"name": "Alice", "age": 25})
person({"name": "Bob"})
person({"city": "New York"})
```

```
Name: Alice, Age: 25
Name: Bob
Unknown format
```

```
class Shape:
    pass

class Circle(Shape):
    __match_args__ = ("radius",)   # 👈 Required for pattern matching
    def __init__(self, radius):
        self.radius = radius

class Rectangle(Shape):
    __match_args__ = ("width", "height")   # 👈 Required for pattern matching
    def __init__(self, width, height):
        self.width = width
        self.height = height

def check_shape(obj):
    match obj:
        case Circle(radius):
            print(f"Circle radius {radius}.")
        case Rectangle(width, height):
            print(f"Rectangle width {width} and height {height}.")
        case _:
            print("This is an unknown shape.")

# Create objects
circle = Circle(10)
rectangle = Rectangle(4, 6)

# Test
check_shape(circle)
check_shape(rectangle)
```

```
Circle radius 10.
Rectangle width 4 and height 6.
```

```
# Python treats int(), str(), and list() inside a case as type checks (equivalent to isinstance(value, int), etc.).

# So, Python is checking whether value is an instance of the class int, str, or list.

def test(value):
    match value:
        case int():
            print("It is an integer.")
        case str():
            print("It is a string.")
        case list():
                print("It is a list.")
            case _:
                print("Unknown type")

    test(42)        # It is an integer.
    test("hello")   # It is a string.
    test([1,2,3])   # It is a list.
```

```
It is an integer.
It is a string.
It is a list.
```