Introduction to Python

Conditionals

Topics

1) Comparison Operators

2) Boolean Operations(and, or not)

3) Operator Precedence

4) Conditionals
    a) if, if-if, if-elif, if-elif-else
    b) Ternary operator

Comparison Operators

Operation	Description
a == b	a equal to b
a != b	a not equal to b
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

Note that = is for assignment and == is for equals.

These operators return either True or False.

Comparison Operators

```python
a = (10 == 5)    # value of a is False
print(a)         # False is printed
print(3 <= 7)    # True
print(3 != 7)    # True
x = 1
b = (x > 10)     # value of b is False
print(b)         # False
```

Boolean Operations

Python provides operators to combine the values using the standard concepts of "**and**", "**or**", and "**not**".

These operators are expressed using the words and, or, and not:

X	Y	X or Y
True	True	True
True	False	True
False	True	True
False	False	False

X	Y	X and Y
True	True	True
True	False	False
False	True	False
False	False	False

X	not X
True	False
False	True

Note: The above Or is also know as the "inclusive or". The conversational "or" may sometime be used as the "exclusive or"(one or the other but not both).

Boolean Operations

```python
x = 4
print((x < 6) and (x > 2))    # True
a = (x > 10) or (x % 2 == 0)
print(a)        # True


b = not (x < 6)
print(b)        # False
print(True or False)    # True
print(True and False)    # False
```

Operator Precedence

Precedence	Operator	Operation
highest	**	exponentiation
	-	negation
	*, /, //, %	multiplication, division, floor division, modulus
	+, -	adding, subtraction
	==,!=,<,>,<=,>=	comparisons
	not	logical not
	and	logical and
	or	logical or
lowest	=	assignment

Boolean Operations

Math operators have the highest precedence. Then comparison operators are followed by logical operators. The assignment operator is evaluated last.

```
result = 3 + 2 * 4 < 14 or 3 == 5
print(result)         # True
```

Although the above expression is correct, often for the sake of readability, and clarity it is often good practice to include parenthesis.

```
result = (3 + 2 * 4 < 14) or (3 == 5)
print(result)         # True
```

Conditionals

The reserved word **if** begins an conditional block.

```
if condition:
    block
```

The condition determines if the block is to be executed.

A block contains one or more statements.

The statements inside of a block must be indented the same number of spaces from the left. The standard is 4 spaces.

If block

What's the output?

```
x = -5
if x > 0:
    print(x)
    print("x is positive")
print("outside of block")
```

outside of block

Note: Since the conditional is false, the entire block is skipped. The print statement outside the block, however, is executed.

If block

What's the output?

```python
x = 5
if x > 0:
    print(x)
    print("x is positive")
print("outside of block")
```

```
5
x is positive
outside of block
```

Note: The above conditional block is executed.

Sequence of Ifs

A sequence of consecutive if statements are independent. None, some or all of them can be executed.

```python
x = 4
if x % 2 == 0:
    print("x is even")
if x > 0:
    print("x is positive")
```

```
x is even
x is positive
```

Note: Both of the above blocks are executed.

Sequence of Ifs

```python
x = -8
if x % 2 == 0:
    print("x is even")
if x > 0:
    print("x is positive")
```

x is even

if-elif

An if block followed by a sequence of elif blocks will execute the **first** block whose condition evaluates to True. No block is executed if all conditions evaluate to False.

```python
x = 25
if x < 5:
    print("x is less than 5")
elif x < 10:
    print("x is less than 10")
elif x < 15:
    print("x is less than 15")
```

Note that all of the above conditions are false and thus no block is executed.

if-elif

```
x = 7
if x < 5:
    print("x is less than 5")
elif x < 10:
    print("x is less than 10")
elif x < 15:
    print("x is less than 15")
```

x is less than 10

Note that only the middle elif block is executed!

if-elif

```
x = 1
if x < 5:
    print("x is less than 5")
elif x < 10:
    print("x is less than 10")
elif x < 15:
    print("x is less than 15")
```

x is less than 5

Note that only the first if block is executed, even though all three conditions are true.

if-elif-else

An `if` statement followed by a sequence of `elif` statements and ending in an `else` statement will execute the first block whose condition evaluates to `True`. If all conditions evaluate to `False`, it will execute the default `else` block.

```python
x = 0
if x < 0:
    print("x is negative")
elif x > 0:
    print("x is positive")
else:
    print("x is zero")
```

x is zero

if-elif-else

```python
x = 10
if x < 0:
    print("x is negative")
elif x > 0:
    print("x is positive")
else:
    print("x is zero")
```

x is positive

and, or, not

Use *and, or,* and *not* Boolean operators to simplify conditionals.

The following

```
if x > 0:
    if x < 10:
        print(x)
```

is equivalent to

```
if x > 0 and x < 10:
    print(x)
```

```
if 0 < x < 10:
    print(x)
```

and, or, not

The following code prints the quadrant of an ordered (x,y) on the Cartesian plane.

```python
x = 4
y = 7
if (x > 0) and (y > 0):
    print("first quadrant.")
elif (x < 0) and (y > 0):
    print("second quadrant.")
elif (x < 0) and (y < 0):
    print("third quadrant.")
elif (x > 0) and (y < 0):
    print("fourth quadrant.")
else:
    print("on x or y axis.")
```

Output:
first quadrant

and, or, not

The following code prints the quadrant of an ordered (x,y) on the Cartesian plane.

```python
x = -26
y = -31
if (x > 0) and (y > 0):
    print("first quadrant.")
elif (x < 0) and (y > 0):
    print("second quadrant.")
elif (x < 0) and (y < 0):
    print("third quadrant.")
elif (x > 0) and (y < 0):
    print("fourth quadrant.")
else:
    print("on x or y axis.")
```

Output:
third quadrant

Ternary Operators

A *ternary operator* evaluates an expression based on the value of a boolean condition. This is sometimes called *conditional expression* or an *inline if-else* statement.

```python
x = 50
grade = "pass" if x >= 60 else "fail"
print(grade)

fail
```

Ternary Operators

```python
x = 60
grade = "pass" if x >= 60 else "fail"
print(grade)
```

pass

Example 1

Write a segment of code which asks the user to enter a number and prints out whether the number is even.

```python
x = int(input('Enter an integer: '))
if x % 2 == 0:
    print("x is even")
else:
    print("x is odd")
```

Example 2: FizzBuzz

Write a segment of code which asks the user to enter a number. Print "fizz" if the number is a multiple of 3, "buzz" if it is a multiple of 5 and "fizzbuzz" if it is a multiple of both 3 and 5. In all other cases, print the number.

```python
x = int(input('Enter an integer: '))
if x % 3 == 0:
    print("fizz")
elif x % 5 == 0:
    print("buzz")
elif x % 3 == 0 and x % 5 == 0:
    print("fizzbuzz")
else:
    print(x)
```

Does the code work?

No, if x = 15, it incorrectly prints "fizz".

Example 2: FizzBuzz

Here's the correct way to implement FizzBuzz.

```python
x = int(input('Enter an integer: '))
if x % 3 == 0 and x % 5 == 0:
    print("fizzbuzz")
elif x % 3 == 0:
    print("fizz")
elif x % 5 == 0:
    print("buzz")
else:
    print(x)
```

Example 2: FizzBuzz

Does the following code work?

```python
x = int(input('Enter an integer: '))
answer = ""
if x % 3 == 0:
    answer += "fizz"
if x % 5 == 0:
    answer += "buzz"
if answer == "":
    answer += str(x)
print(answer)
```

Yes! Compare this to the previous slide.

Lab 1: Absolute Value

Create a new repl on repl.it. If you wish, you can use this same repl for all of the labs in this lecture.

Write a program which asks for an integer and prints out the absolute value of the number.

Example output 1:

Enter a number: 34

Absolute value of 34 is 34.

Example output 2:

Enter a number: -11

Absolute value of -11 is 11.

Lab 2: Quadratic

Create a new repl on repl.it. Write a program which asks for the coefficients of the quadratic f(x) = ax^2 + bx + c and prints out the number of real roots of f(x).

Hint: Compute the discriminant b^2 – 4ac. If the discriminant is > 0, it has two real roots. If it is < 0 , it has no real roots and if it is = 0, it has one repeated root.

Enter a: 1
Enter b: -2
Enter c: -15
Two real roots.

Enter a: 1
Enter b: 0
Enter c: 1
No real roots.

Enter a: 1
Enter b: -2
Enter c: 1
One repeated real root.

References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.

2) Richard Halterman, Fundamental of Python Programming.