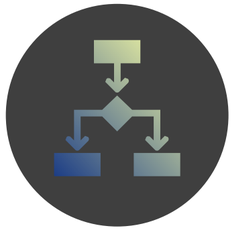
CONDITIONAL LOGIC

CONDITIONAL LOGIC



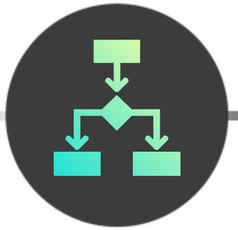
In this section we'll cover **conditional logic**, and write programs that make decisions based on given criteria using true or false expressions

TOPICS WE'LL COVER:

- The Boolean Data Type
- Boolean Operators
- Conditional Control Flow

GOALS FOR THIS SECTION:

- Understand the properties of the Boolean data type
- Learn to write true or false expressions using Boolean operators
- Learn to control the flow of the program using conditional logic statements



THE BOOLEAN DATA TYPE

The **Boolean data type** has two possible values: **True**& **False**

- **True** is equivalent to **1** in arithmetic operations
- **False** is equivalent to **0** in arithmetic operations

```
True * 1
```

```
1
```

```
False * 1
```

```
0
```

The **not** keyword inverts Boolean values:

- **notTrue** is equivalent to **False**
- **not False** is equivalent to **True**

```
(not True) * 1
```

```
0
```

```
(not False) * 1
```

```
1
```

COMPARISON OPERATORS

The Boolean Data Type

Boolean Operators

Conditional Control Flow

		True	False
==	Equal	`5==5`	`5==3`
!=	Not Equal	`'Hello' != 'world!'`	`10 != 10`
<	Less Than	`5 < 6 < 7`	`21 < 12`
>	Greater Than	`10>5`	`10>5>7`
<=	Less Than or Equal	`5<=5`	`(5 +5)<=8`
>=	Greater Than or Equal	`11>=9>=9`	`len('I') >=len('am')`

MEMBERSHIP TESTS

The Boolean Data Type

Boolean Operators

Conditional Control Flow

Membership tests check if a value exists insidean iterable data type

- The keywords **in** and **not in** are used to conduct membership tests

```
message = 'I hope it snows tonight!'
'snow' in message
```

```
True
```

*'snow' is in the string assigned to **message** so the membership test returns **True***

```
'rain' in message
```

```
False
```

*'rain' is NOT in the string assigned to **message** so the membership test returns **False***

```
message = 'I hope it snows tonight!'
'snow' not in message
```

```
False
```

*'snow' is in the string assigned to **message**, so the membership test checking if it is NOT in message returns **False***

BOOLEAN OPERATORS

The Boolean Data Type

Boolean Operators

Conditional Control Flow

Boolean operators allow you to combine multiple comparison operators

- The **and** operator requires *all* statements to be true
- The **or** operator requires *one* statement to be true

One is True and the other False

```
5 > 4 and 7 > 8
```

```
False
```

```
5 > 4 or 7 > 8
```

```
True
```

You can chain together any number of Boolean expressions

```
15 > 10 and 11 > 10 and 12 > 10
```

```
True
```

TIP: If the first expression chained by **and** is **False**, or by **or** is **True**, the rest of the clauses will not be evaluated

Place simple clauses first to eliminate the need to run complex clauses, making your program more efficient

BOOLEAN OPERATORS

The Boolean Data Type

Boolean Operators

Conditional Control Flow

Boolean operators allow you to combine multiple comparison operators

- The **and** operator requires *all* statements to be true
- The **or** operator requires *one* statement to be true

Use parentheses to control the order of evaluation:

```
price = 105
item = 'skis'

(price <= 100 and item == 'ski poles') or (item == 'skis')
```

```
True
```

Because ***item ==‘skis’*** *the* ***or*** *operator makes the expression True*

```
price <= 100 and (item == 'ski poles' or item == 'skis')
```

```
False
```

Because ***price <=100 and*** *returns false the operator makes the expression False*

THE IF STATEMENT

The **if statement** runs lines of indented code when a given logical condition is met


```
if condition:
```

Indicates an ***if*** *statement*

A logical expression that evaluates to True or False

Examples:

- *price >100*
- *product == 'skis'*
- *inventory >0* ***and*** *inventory <=10*

THE IF STATEMENT

The **if statement** runs lines of indented code when a given logical condition is met


```
if condition:
    do this
```

Code to run if the logical expression returns True

(must be indented!)

THE IF STATEMENT

The **if statement** runs lines of indented code when a given logical condition is met

- The Boolean Data Type
- Boolean Operators
- Conditional Control Flow

EXAMPLE | *Determining experience level for a snowboard*

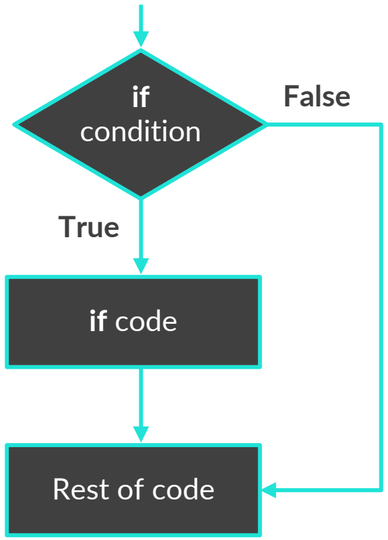
```
price = 999.99

if price > 500:
    print('This snowboard is designed for experienced users.')

print('This code will run whether or not the if statement is True.')
```

```
This product is for experienced users.
This code will run whether or not the if statement is True.
```

How does this code work?



CONTROL FLOW

The Boolean Data Type

Boolean Operators

Conditional Control Flow

Control flow is a programming concept that allows you to choose which lines to execute, rather than simply running all the lines from top to bottom

There are three conditional statements that help achieve this: **if**, **else**, and **elif**

EXAMPLE *Determining experience level for a snowboard*

```
price = 499.99

if price > 500:
    print('This product is for experienced users.')

print('This code will run whether or not the if statement is True.')
```

```
This code will run whether or not the if statement is True.
```

*Python uses **indentation** to depart from a linear flow*

In this case, the first print statement only runs if price is greater than 500

Press tab, or use four spaces, to indent your code when writing if statements or you will receive an **IndentationError**

THE ELSE STATEMENT

The **else statement** runs lines of indented code when the none of the logical conditions in an if or elifstatements are met

- The Boolean Data Type
- Boolean Operators
- Conditional Control Flow

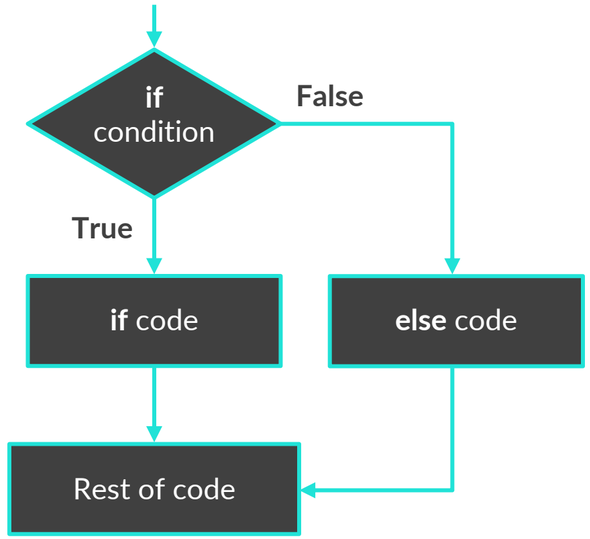
EXAMPLE | *Determining experience level for a snowboard*

```
price = 499.99
price_threshold = 500

if price > price_threshold:
    print('This snowboard is for experienced users.')
else:
    print('This board is suitable for a beginner.')
```

```
This board is suitable for a beginner.
```

How does this code work?



THE ELIF STATEMENT

The **elif statement** lets you specify additional criteria to evaluate when the logical condition in an if statement is not met

- The Boolean Data Type
- Boolean Operators
- Conditional Control Flow

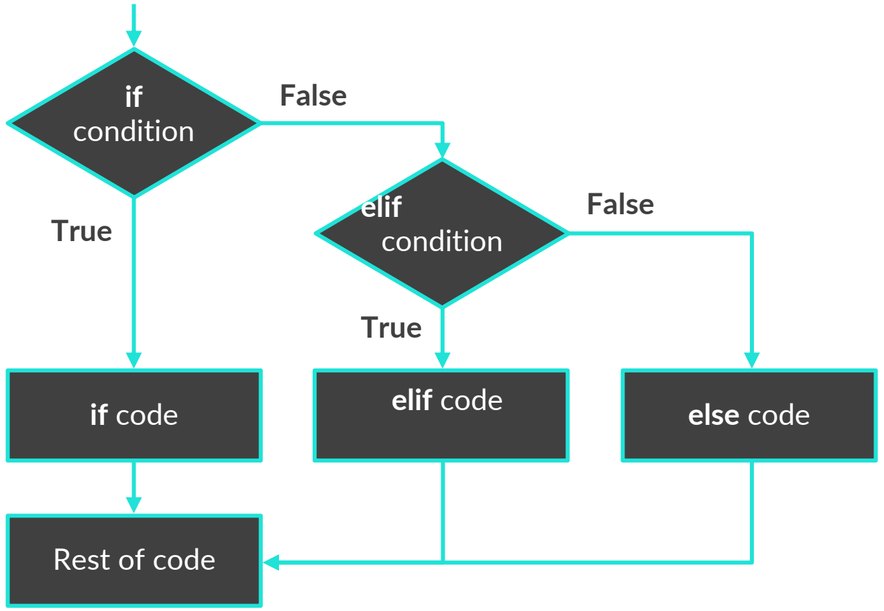
EXAMPLE | *Determining experience level for a snowboard*

```
price = 499.99
expert_threshold = 500
intermediate_threshold = 300

if price > expert_threshold:
    print('This board is for experienced users.')
elif price > intermediate_threshold:
    print('This board is good for intermediate_users.')
else:
    print('This should be suitable for a beginner.')
```

```
This board is good for intermediate_users.
```

How does this code work?



THE ELIF STATEMENT

Any number of **elif statements** can be used between the if and else statements

As more logical statements are added, it's important to be careful with their order

Will the elifs below ever run?

```
price = 1499.99
luxury_threshold = 1000
expert_threshold = 500
intermediate_threshold = 300

if price > intermediate_threshold:
    print('This board is good for intermediate users')
elif price > expert_threshold:
    print('This board is for experienced users')
elif price > luxury_threshold:
    print("The gold doesn't improve the ride but it looks great!")
else:
    print('This should be suitable for a beginner.')
```

```
This board is good for intermediate users
```

***No**, because any value that is true for them will also be true for the if statement above them*

Either put the most restrictive conditions first, or be more explicit with your logic

NESTED IF STATEMENTS

Nested if statements let you specify additional criteria to evaluate after a logical condition in an if statement is met

- The Boolean Data Type
- Boolean Operators
- Conditional Control Flow

EXAMPLE | *Trying to purchase a product*

```
price = 499.99
budget = 500
inventory = 0


if budget > price:
    if inventory > 0:
        print('You can afford this and we have it in stock!')
    else:  # equivalent to if inventory <= 0
        print("You can afford this but it's out of stock.")
else:
    print(f'Unfortunately, this board costs more than ${budget}.')
```

```
You can afford this but it's out of stock.
```

How does this code work?