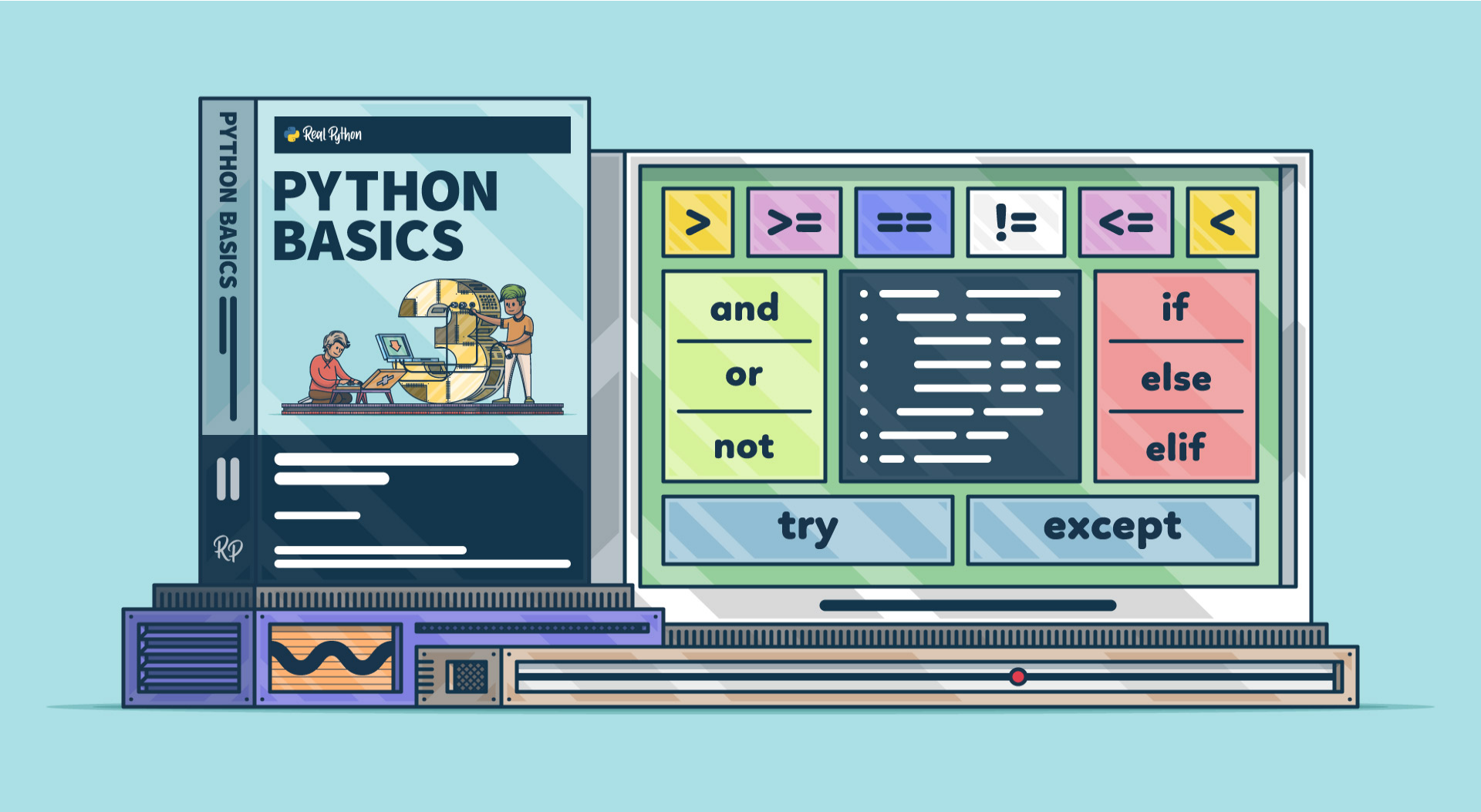
Python Basics: Conditional Logic & Control Flow



Conditional Logic and Control Flow

Do something **if** something else is true.

- **Conditional logic** allows you to **check** if something is true or false.
  - *Does Bobby have more apples than Bill?* 🍏
  - *Is Jane faster than June?* 🏃‍♀️💨
- **Control flow** allows you to create **branches** of execution.
  - *If Bobby has more apples, then give apples to Bill* ✋🍏
  - *If Jane is faster, then give June good sneakers* 👟👟

Conditional Logic

Find out whether something is true or false.

```
("a" != "a") or not (2 >= 3)
```

Control Flow

Create branches for your program to flow into.

```
while True:
    if python_is_awesome:
        if not overwhelmed:
            learn()
        elif overwhelmed:
            take_break()
            continue
        elif need_to_sleep:
            break
        else:
            continue

        reflect_on_learning()
    else:
        take_break()
```

Recover From Errors

Errors happen, handle them gracefully.

```
for chapter in python_basics:
    try:
        learn()
    else:
        take_break()
```

Conditional Logic and Control Flow

- Conditional logic 🤔
- Control flow 🚦
- Error handling ❗

Conditional Logic: *Add Some Logic*

Conditional Logic: Add Some Logic

Find out whether something is true or false.

- **Conditional logic** allows you to **check** if something is true or false.
  - *Does Bobby have more apples than Bill?* 🍏
  - *Is Jane faster than June?* 🏃‍♀️💨
  - *Is 5 larger than 8?*
  - *Is n before p in the alphabet?*

Add Some Logic

In this lesson you'll cover:

- Boolean comparators
- Conditional statements
- Logical operators
- Operator precedence

Boolean Comparators

- `>` Greater than
- `<` Less than
- `>=` Greater than or equal to
- `<=` Less than or equal to
- `!=` Not equal to
- `==` Equal to

Conditional Statements

These all result in `True`

- `10 > 5` Ten is greater than five
- `1 < 2` One is less than two
- `10 >= 9` Ten is greater than or equal to nine
- `5 <= 5` Five is less than or equal to five
- `1 != 0` One is not equal to zero
- `100 == 100` One-hundred is equal to one-hundred

Conditional Statements

These all result in `False`

- `10 < 5` Ten is less than five
- `1 > 2` One is greater than two
- `10 <= 9` Ten is less than or equal to nine
- `4 >= 5` Four is greater than or equal to five
- `1 == 0` One is equal to zero
- `100 != 100` One-hundred is not equal to one-hundred

Logical Operators

and

and

- `True and False` ❌
- `True and True` ✅
- `False and False` ❌
- `False and True` ❌

or

or

- `True or False` ✅
- `True or True` ✅
- `False or False` ❌
- `False or True` ✅

```
not
```

not

- `not False` ✅
- `not True` ❌

Operator Precedence

- `<`, `<=`, `==`, `!=`, `>=`, `>`
- `not`
- `and`
- `or`

Add Some Logic

In this lesson you've covered:

- Boolean comparators
- Conditional statements
- Logical operators
- Operator precedence

Conditional Logic: *Building Complex Expressions*

Building Complex Expressions

- Examples of complex conditional expressions

Example 1

```
True and not (1 != 1)
```

Example 2

```
("a" != "a") or not (2 >= 3)
```

Building Complex Expressions

- Examples of complex conditional expressions

Control Flow: *Control the Flow of Your Program*

Control the Flow of Your Program

Create branches 🌿🤖

Control the Flow of Your Program

In this lesson you'll be learning about:

- `if`
- `else`
- `elif`
- `if`…`elif`…`else`

```
if
```

if

```
if [condition]:
    ...

print("done")
```

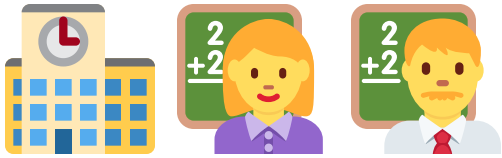
if

```
if True:
    print("Hello!")

if False:
    print("Hello!")

print("done")
```

School Grading Example



```
else
```

else

```
if [condition]:
    ...
else:
    ...
```

```
elif
```

elif

```
if [condition]:
    ...
elif [condition]:
    ...
else:
    ...
```

elif

```
if [condition]:
    ...
elif [condition]:
    ...
elif [condition]:
    ...
else:
    ...
```

Control the Flow of Your Program

In this lesson you've learnt about:

- `if`
- `else`
- `elif`
- `if`...`elif`...`else`

Control Flow: *Nested* `if` *Statements*

Example: Evaluate the Winner

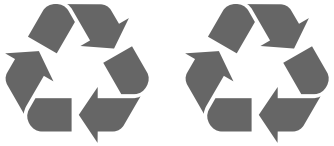
- Two people play either basketball or golf
- They tell you what sport they have been playing and their score
- You have to evaluate who won

Example: Evaluate the Winner

- In golf, the lower score wins ⛳
- In basketball, the higher score wins 🏀

Refactor!

Refactor Again!



Challenge: *Find the Factors of a Number*

Control Flow: *Break Out of the Pattern*

Break Out of the Pattern

In this lesson you'll cover:

- Bringing loops into the mix
- Using the `break` keyword
- Using the `continue` keyword

if Statements and for Loops

```
sum_of_evens = 0

for n in range(101):
    if n % 2 == 0:
        sum_of_evens = sum_of_evens + n

print(sum_of_evens)
```

```
break
```

break

```
for n in range(4):
    if n == 2:
        break
    print(n)
```

break

```
n = 0

while True:
    print(n)
    if n > 5:
        break
    n = n + 1
```

```
continue
```

continue

```
for n in range(4):
    if n == 2:
        print("there goes two")
        continue
    print(n)
```

continue

```
n = 0

while True:
    print(n)
    if n < 5:
        n = n + 1
        continue
    else:
        break
    print("end of loop")
```

Break Out of the Pattern

In this lesson you've covered:

- Bringing loops into the mix
- Using the `break` keyword
- Using the `continue` keyword

Control Flow: *Recover From Errors*

Recover From Errors

In this lesson you'll cover:

- Errors and exceptions
- Types of errors and exceptions

SyntaxError

```
>>> if
  ...
    if
      ^
SyntaxError: invalid syntax
```

ValueError

```
>>> int("hello")
Traceback (most recent call last):
    ...
ValueError: invalid literal for int() with base 10: 'not a number'
```

TypeError

```
>>> "1" + 2
Traceback (most recent call last):
    ...
TypeError: can only concatenate str (not "int") to str
```

NameError

```
>>> print(does_not_exist)
Traceback (most recent call last):
    ...
NameError: name 'does_not_exist' is not defined
```

ZeroDivisionError

```
>>> 1 / 0
Traceback (most recent call last):
    ...
ZeroDivisionError: division by zero
```

OverflowError

```
>>> pow(2.0, 1_000_000)
Traceback (most recent call last):
    ...
OverflowError: (34, 'Result too large')
```

Recover From Errors

In this lesson you've covered:

- Errors and exceptions
- Types of errors and exceptions

Control Flow: *The* `try` *and* `except` *Keywords*

The `try` and `except` Keywords

In this lesson you'll cover:

- The `try` … `except` structure
- How to handle exceptions gracefully

The try and except Keywords

```
try:
    number = int(input("Enter an integer: "))
except ValueError:
    print("That was not an integer")
```

The try and except Keywords

```
try:
    ...
except [exception]:
    ...
```

Catching Different Exceptions

```
def divide(num1, num2):
    try:
        print(num1 / num2)
    except TypeError:
        print("Both arguments must be numbers")
    except ZeroDivisionError:
        print("num2 must not be 0")
```

The Bare `except` Clause

```
try:
    # Do lots of hazardous things that might break
except:
    print("Something bad happened!")
```

The Bare except Clause

Recover From Errors

In this lesson you've covered:

- The `try` … `except` structure
- How to handle exceptions gracefully

Putting it Together: *Simulate and Calculate Probabilities*

Simulate and Calculate Probabilities

In this lesson you'll implement a program that:

- Uses the `random` module
- Simulates many coin tosses
- Calculates the ratio of heads to tails
- Allows you to alter the behavior of the coin

The random Module

```
>>> import random
>>> random.randint(1, 10)
    9
```

Fair Coins

```python
import random

def coin_flip():
    """Randomly return 'heads' or 'tails'."""
    if random.randint(0, 1) == 0:
        return "heads"
    else:
        return "tails"
```

Simulate

```
# First initialize the tallies to 0
heads_tally = 0
tails_tally = 0

for trial in range(10_000):
    if coin_flip() == "heads":
        heads_tally = heads_tally + 1
    else:
        tails_tally = tails_tally + 1
```

Unfair Coins

```
import random

def unfair_coin_flip(probability_of_tails):
    if random.random() < probability_of_tails:
        return "tails"
    else:
        return "heads"
```

Simulate and Calculate Probabilities

In this lesson you've implemented a program that:

- Used the `random` module
- Simulated many coin tosses
- Calculated the ratio of heads to tails
- Allowed you to alter the behavior of the coin

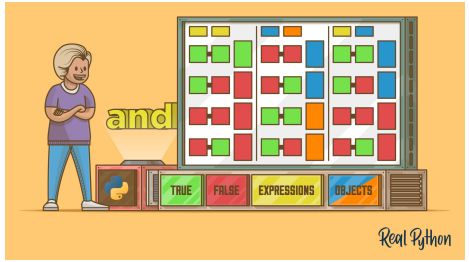
PYTHON BASICS: *CONDITIONAL LOGIC & CONTROL FLOW*

Conditional Logic & Control Flow

In this course you have

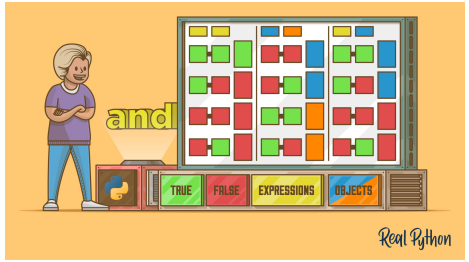
- Compared values
- Used logical operators
- Learned how to control the flow of your programs
  - Creating branches with `if` blocks
  - Using Loop keywords
  - Combining `if` with loops.
- Learned how to handle exceptions

Additional Resources

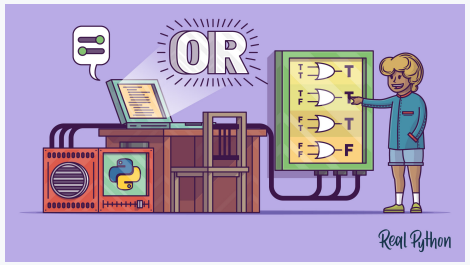


Conditional Statements in Python

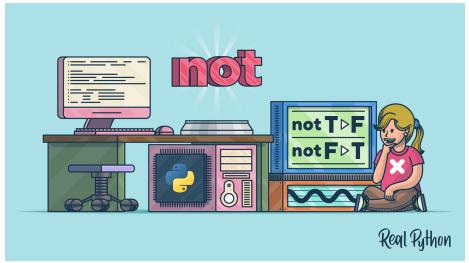
Additional Resources



Using the “and” Boolean Operator in Python



Using the “or” Boolean Operator in Python



Using the “not” Boolean Operator in Python

Additional Resources

Python “for” Loops (Definite Iteration)

Python “while” Loops (Indefinite Iteration)

Python Exceptions: An Introduction