Getting Started with Python

Python Qualities

- *Strongly* typed
  - It enforces data types so you can't concatenate a string and a integer, for example.
- *Dynamically*, *implicitly* typed
  - So, you don't have to explicitly declare variable data types. Data types are enforced at runtime.
- *Case sensitive*
  - For example, token and TOKEN are two different variables.
- *Object-oriented*
  - Everything is an object.

Language Primitives

Variables

- Variables are containers for data. The syntax to declare them is:
  - `variable_name = variable_value`
- Example:
  - `node_ip = "192.168.2.150"`
  - `username = "admin"`
  - `password = "Rubrik123!!"`

Numbers and Operators

- **Numbers** can be integers, floating points, Booleans, or complex numbers:
  - **Integers** are whole numbers, such as 1, 2, 22, 476, -99999
  - **Floats** have decimal points, such as 1.0, 2.22, 22.098, 476.1, -99999.9
  - **Booleans** represent either True or False (or 1 or 0)
- **Operators** are things like addition and subtraction as well as `and` and `or`

Python

```
>>> 2 + 3   # Addition
5
>>> num1 = 10
>>> num2 = 9.99
>>> num3 = num1 + num2
>>> num3
19.990000000000002
>>> 8 - 5   # Subtraction
3
>>> 2 * 6   # Multiplication
12
>>> 12 / 3  # Division
4.0
>>> 7 % 3   # Modulus (returns the remainder from division)
1
>>> 3 ** 2  # Raise to the power
9
```

Strings

- Strings are lines of text that are declared with single or double quotes:

```Python >>>
>>> simple_string = "hey!"
>>> simple_string
'hey!'
```

Functions

- A function in Python is a logical unit of code containing a sequence of statements indented under a name given using the `def` keyword
- Functions allow you to create a logical division of a big project into smaller modules making code more manageable and extensible
- While programming, it prevents you from adding duplicate code and promotes reusability

Writing Functions

- A function definition consists of following components:
  - Keyword `def` marks the start of function header
    - A function name to uniquely identify it
    - Optional parameters for passing values to a function
    - A colon (:) to mark the end of function header
  - Optional docstring to describe what the function does
  - One or more valid python statements that make up the function body
  - Optional return statement to return a value from the function

```
def fn(arg1, arg2,...):
    """docstring"""
    statement1
    statement2
```

Calling Functions

- Once a function is defined, simply type the function name with appropriate parameters

```
>>> greet('Paul')
Hello, Paul. Good morning!
```

Conditional Code

- Python supports the usual logical conditions when decision making is required :
  - Equals: `a == b`
  - Not Equals: `a != b`
  - Less than: `a < b`
  - Less than or equal to: `a <= b`
  - Greater than: `a > b`
  - Greater than or equal to: `a >= b`

Example

Python

```
# Python 3

count = 1  <-- variable
# Code block 1 <-- inline comment
while count < 11:
    print(count)
    count = count + 1
# Code block 2
if count == 11:  <-- conditional
    print('Counting complete.')
```

`if` Condition Example

- An "if statement" is written by using the `if` keyword.
- Example:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

`elif` Example

- The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

`else` Example

- The `else` keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

- You can also have an `else` without the `elif`:

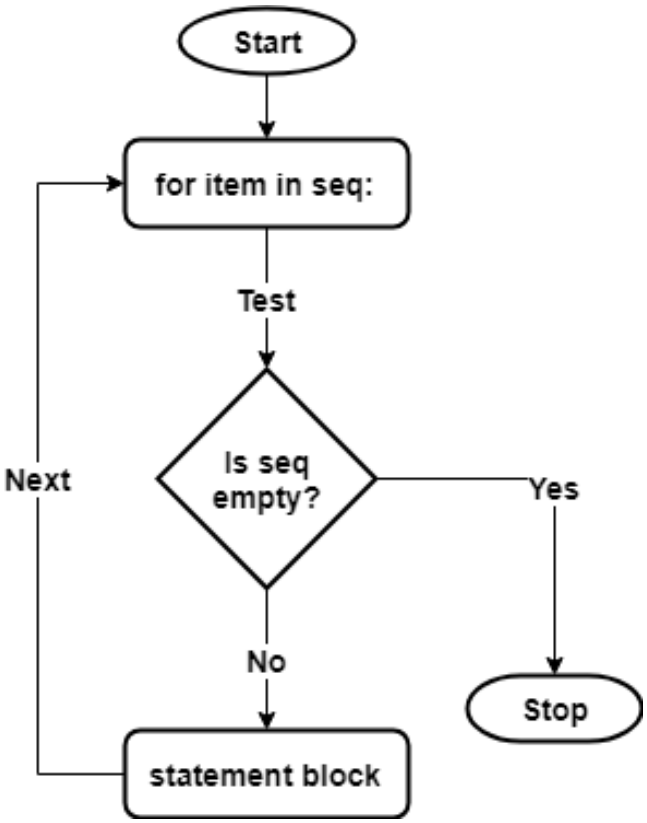
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Loops

- Loops are used in programming to repeat a specific block of code
- There are two common types of loops with Python:
  - for
  - while

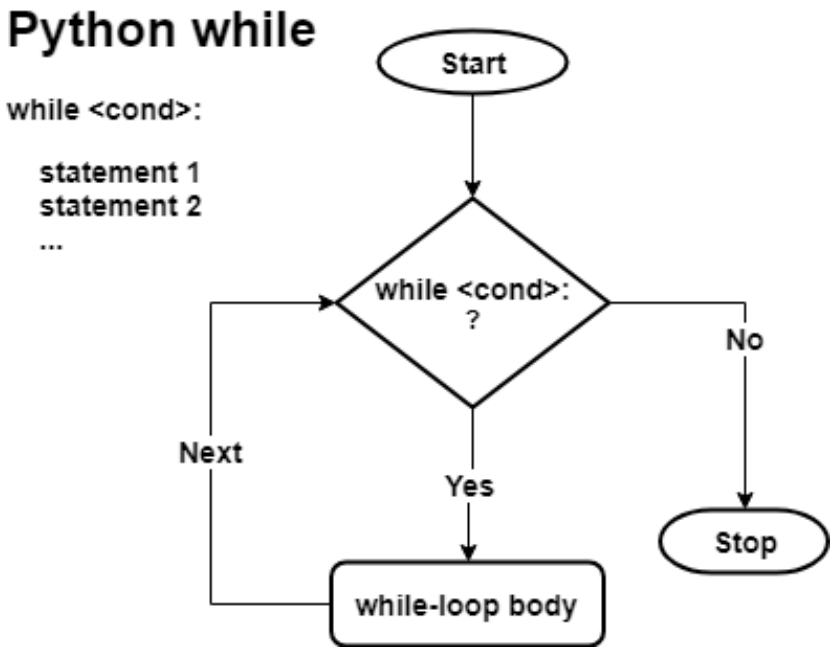
for Loop

- A “for loop” in Python requires at least two variables to work
  - The first variable is the iterable object such as a list or a string
  - The second variable is to store the successive values from the sequence in the loop



while Loop

- A while loop is a control flow structure which repeatedly executes a block of code indefinite number of times until the given condition becomes false



Errors and Exceptions

Syntax Errors

- Syntax errors will prevent execution of the program. In this example, the if statement is missing a colon to end the statement.

Python >>>

```
>>> if x < 9
   File "<stdin>", line 1
     if x < 9
              ^
SyntaxError: invalid syntax
```

Exceptions

- Exception errors occur during program execution. Python has a number of built-in exceptions. For example:

Python >>>

```
>>> 12/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Rubrik SDK for Python

Python Module

- Easy to install, easy to consume
- Quick start guide and documentation available
- Build scripts to automate workflows in Python

```
pip install rubrik_cdm
```

Base API Calls

- DELETE, GET, PATCH, POST, JOB_STATUS

```
import rubrik

rubrik = rubrik.Connect()

cluster_version = rubrik.get('v1', '/cluster/me/version')

print(cluster_version)
```

Simplified Functions

```
import rubrik

rubrik = rubrik.Connect()

cluster_version = rubrik.cluster_version()

print(cluster_version)
```


build
Building the Future of Data Management