### Cryptocurrency Algorithmic Trading

HKU-SCF FinTech Academy

# Python Basics

Python is a programming language named after the British comedy troupe Monty Python.
In This chapter, we will cover the key Python skills you'll need so you can start programming.

**Note that there are quite some optional external references included in this Module, feel free to go through or skip them at your own interest.**

## Estimated Time to Finish:

3-5 hours (excluding Optional Materials)

## Main Learning Objectives:

- learning the basic syntax of python, variables, functions, arithmetics
- learning about `string`, `list`, `tuple` and `dictionary`
- learning how to import external libraries

## Learning Programming

It can take months or years to become proficient in basic programming. Many times, you'll need to review lessons several times. Programming also necessitates hands-on expertise; instead of reading/watching the examples, actively follow along on your computer. Learning programming is similar to learning how to drive a car or doing physical fitness exercises. it is not possible to attain the skills solely by reading about it and/or viewing videos.

You are recommended to read this article: How to teach yourself hard things

Solving practice problems is very useful to review your understanding. After familiarizing with basics, try experimenting with your newly acquired skills. If you get stuck, you can search online/documentation/books for those specific problems (go to the Optional Resources below for more), and if that fails, you can ask us for help.

## The Tutorial

You will be learning the basics of python from the tutorials on Kaggle, a pupular platform for data science competitions and also a source of useful datasets. Catered towards data science

(which is relevant to algotrading), Kaggle will be main learning resource for this module as the tutorials are interactive and filled with examples. You will also get to try out some exercises inside Kaggle notebooks after learning each subtopic.

https://www.kaggle.com/learn/python

# Overview

1. Syntax
2. Functions
3. Booleans & Conditions
4. Lists
5. Loops and List comprehension
6. Strings and dictionaries
7. Working with External Libraries

---

# Summary

Below is the summary of the content covered in the Kaggle tutorials. Relevant Cheatsheets from Python Crash Course 2nd Edition is also provided (Credit goes to the author Eric Matthes)

## 1. Syntax (cheatsheet)

- `print("Strings are enclosed by double or single quotation marks")`
- arithmetics

| Operator | Name | Description |
|---|---|---|
| `a + b` | Addition | Sum of `a` and `b` |
| `a - b` | Subtraction | Difference of `a` and `b` |
| `a * b` | Multiplication | Product of `a` and `b` |
| `a / b` | True division | Quotient of `a` and `b` |
| `a // b` | Floor division | Quotient of `a` and `b`, removing fractional parts |
| `a % b` | Modulus | Integer remainder after division of `a` by `b` |
| `a ** b` | Exponentiation | `a` raised to the power of `b` |
| `-a` | Negation | The negative of `a` |

- declaring variables

> **Variables** are used to assign **labels** to values.

- commenting with `#`

---

## 2. Functions (cheatsheet)

> Functions are named blocks of code, designed to do one specific job. Information passed to a function is called an **argument**, and information received by a function is called a **parameter**.

```
def double(num):
    return num*2

print(double(10))
```

In the above example, `10` is the argument, `num` is the parameter

- Getting help with `help()`
- How to define functions and write docstrings
- the `return` keyword
- function arguments
- stacking functions

---

## 3. Booleans & Conditions (cheatsheet)

> If statements are used to test for particular conditions and respond appropriately.

- Booleans: `True` or `False`
- Comparisons

| Operation | Description | | Operation | Description |
|---|---|---|---|---|
| `a == b` | `a` equal to `b` | | `a != b` | `a` not equal to `b` |

---

| Operation | Description | | Operation | Description |
|---|---|---|---|---|
| `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` |

- `and`, `or`, `not` keywords
- `if`, `elif`, `else` conditionals

---

## 4. Lists (cheatsheet)

> A list stores a series of items in a particular order. You access items using an **index**, or within a **loop**.

- indexing (first element has index 0, last element has index -1)
- slicing (`list[start:end:strides]`)
- modifying lists in place
- list functions: `len()`, `sorted()`, `sum()`, `max()`
- list methods: `.append()`, `.index()`, `.pop()`
- `in` operator
- Lists `[]` vs Tuples `{}`

> Tuples are similar to lists, but the items in a tuple can't be modified.

---

## 5. Loops and List comprehension (cheatsheet)

- `for` loops and `range()`

> A For loop is used to repeat a specific block of code a known number of times.

- `while` loops

> A while loop repeats a block of code as long as a certain condition is true.
> While loops are especially useful when you
> can't know ahead of time how many times a loop should run.

- shortening your code with List comprehension

---

## 6. Strings and dictionaries

### Strings

> A string is a series of characters, surrounded by single or double quotes.

- String syntax, manipulation

| What you type | What you get | example | `print(example)` |
|---|---|---|---|
| `\'` | `'` | `'What\'s up?'` | `What's up?` |
| `\"` | `"` | `"That's \"cool\""` | `That's "cool"` |
| `\\` | `\` | `"Look, a mountain: /\\"` | `Look, a mountain: /\` |
| `\n` | | `"1\n2 3"` | `1`<br>`2 3` |

- String methods: `.upper()`, `.lower()`, `.index()`, `.startswith()`
  - besides using `.format()`, f-strings also allows using variables inside strings to build dynamic messages.
- conversion to/from lists: `.join()`, `.split()`
- `str()`
- Exercises

### Dictionaries (cheatsheet)

> Dictionaries store connections between pieces of information. Each item in a dictionary is a **key-value** pair.

- dictionary comprehension
- looping over a dictionary with methods: `.keys()`, `.values()`, `.items()`
- Exercises

---

## 7. Working with External Libraries

- `import __ as _`
- `from __ import *`
- importing `math` and `numpy`
- operator overloading
- using datetime
    - The datetime library will be particularly useful when we have to deal with time later on Quantconnect

### installing python libraries with `pip`

- How to download and install Python Packages and Modules with Pip

---

## Optional Resources

If you want to explore more before moving onto Object-Oriented Programming, you might be interested in:
Beginner resources - Python resources for everybody

- a list of good python learning materials and tips and where to get help

How to Think Like a Computer Scientist: Interactive Edition

- gives you a solid foundation to programming, teaches debugging right at the beginning, includes case studies, exercises, etc
- based on the book "Think Python"

Programming with Mosh - Python for Beginners [Full Course]

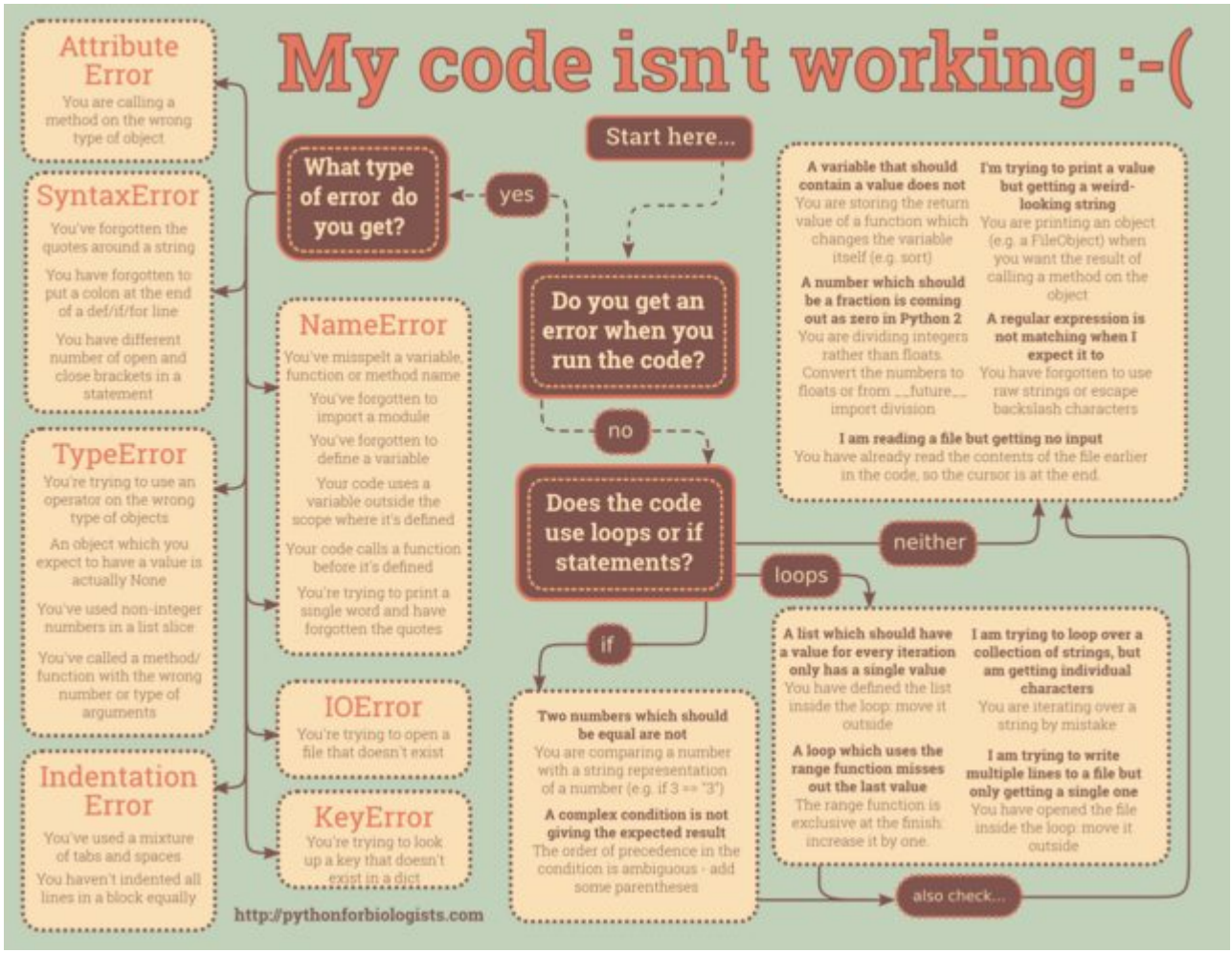
- A full course on python basics, if you do not feel like reading text-based guides you can learn the entire module 1a and 1b by following along this video

---

## Common Beginner Errors

Beginners may have trouble dealing with exceptions and errors. Here is a handy flowchart from https://pythonforbiologists.com for troubleshooting errors

HD version


My code isn't working :-(
Start here...
What type of error do you get?
yes
Do you get an error when you run the code?
no
Does the code use loops or if statements?
neither
loops
if
Attribute Error
You are calling a method on the wrong type of object
SyntaxError
You've forgotten the quotes around a string
You have forgotten to put a colon at the end of a def/if/for line
You have different number of open and close brackets in a statement
TypeError
You're trying to use an operator on the wrong type of objects
An object which you expect to have a value is actually None
You've used non-integer numbers in a list slice
You've called a method/function with the wrong number or type of arguments
Indentation Error
You've used a mixture of tabs and spaces
You haven't indented all lines in a block equally
NameError
You've misspelt a variable, function or method name
You've forgotten to import a module
You've forgotten to define a variable
Your code uses a variable outside the scope where it's defined
Your code calls a function before it's defined
You're trying to print a single word and have forgotten the quotes
IOError
You're trying to open a file that doesn't exist
KeyError
You're trying to look up a key that doesn't exist in a dict
A variable that should contain a value does not
You are storing the return value of a function which changes the variable itself (e.g. sort)
A number which should be a fraction is coming out as zero in Python 2
You are dividing integers rather than floats. Convert the numbers to floats or from __future__ import division
I'm trying to print a value but getting a weird-looking string
You are printing an object (e.g. a FileObject) when you want the result of calling a method on the object
A regular expression is not matching when I expect it to
You have forgotten to use raw strings or escape backslash characters
I am reading a file but getting no input
You have already read the contents of the file earlier in the code, so the cursor is at the end.
A list which should have a value for every iteration only has a single value
You have defined the list inside the loop: move it outside
A loop which uses the range function misses out the last value
The range function is exclusive at the finish: increase it by one.
I am trying to loop over a collection of strings, but am getting individual characters
You are iterating over a string by mistake
I am trying to write multiple lines to a file but only getting a single one
You have opened the file inside the loop: move it outside
Two numbers which should be equal are not
You are comparing a number with a string representation of a number (e.g. if 3 == "3")
A complex condition is not giving the expected result
The order of precedence in the condition is ambiguous - add some parentheses
also check...
http://pythonforbiologists.com