# Python Fundamentals Cheatsheet

A guide that covers everything you need you know about the basics of python programming.

## 1. Overview of Python

### Why learn Python?

- One of the most popular
- Been around a while, hence reliable
- Beginner friendly and easy to learn
- Can do all sorts of things like AI & ML, game or web dev, data science, automation, etc.
- Fundamentals are useful for all other languages too

### Getting Python

- Go to www.python.org
- Click the download link and choose the best option for your system
- Install python and check the `python path` checkbox during the process
- Installed! 

> To check if you have python, open a terminal like cmd and type `python --version`. If you have python in your system it will show the version or else you don't have it

### Python Environment

- Open python from start menu in the terminal
- Write your first `hello world` program
- Use `exit()` function to exit and close the program

#### Notepad

- We can also use notepad and other text editors for writing python code
- Simply open notepad and write some python code
- Save the file with `.py` extension which stands for python
- Open terminal and type `python <saved file's full path>`
- Code will be executed!

#### IDEs

- Get an IDE like VS Code for your system
- Open vscode and configure it to look pretty
- Get the python extension by microsoft
- Select the python interpreter using command palette and set it as downloaded python version
- Open the python file and run it using play btn
- Code will be executed!

> We can use any one of the several IDE's available online to run python code like PyCharm, Jupyter Notebook, etc.

## Bonus - Your first ever python program

```
# first ever program
print('Hello World!!!')
```

## 2. Python Basics

### Comments

- Used to explain code as python ignores it
- Use `#` symbol for single line comment
- For multi-line comments wrap it inside `''' comment '''`
- Can be put above or right next to any code
- Can be used for line by line debugging

### Variables

- Used to assign and store values
- Can be used to store different data types
- Should only contain letters, numbers, and underscores
- Shouldn't start w number
- Case sensitive and can't be python keywords
- Can be reset later after defining

### Input Function

- Used to get input from the user in terminal
- Used as `input(<'Input details or question'>)`
- Input can pe stored in a variable
- Input function always returns a string even if user enters a number

### Math Functions

- Several functions/methods to handle calculations
- A few built in functions like `round(n)` and `abs(n)`
- Import the `math` module and use its methods for complex stuff
- Math module has methods like `math.ceil(n)` and `math.floor(n)`
- Google *python math modules* to learn more methods

## 3. Data Types in Python

### Data Types

- There are multiple data types in python
- `str` represents strings - any text or characters
- `int` is for integers like 4, 7, 27328
- `float` is for numbers with decimals like 3.2, 7839.83
- `bool` is for True/False (case sensitive)
- `type(name)` - used to find the data type of a variable

### Type Conversion

- One data type can be converted into another
- Built-in functions like `str()`, `int()` can be used for this
- `int('birthday')` - converts the string into integer

## 4. Strings and String Methods

### Basics of Strings

- Wrap text b/w single or double quotes - `print("Hello")`
- Numbers can also be used as strings `print('5')`
- Use quotes carefully and when required `print("Give me John's number")`
- For multiline strings use the quotes 3 times

- Use **formatted strings** for easy string concatenation

```
firstName = 'John'
lastName = 'Doe'

# concatenation
message = firstName + ' ' + lastName + ' is a nice guy!!'
# formatted string
msg = f'{firstName} {lastName} is a nice guy!!'
```

### String Methods

- Used to handle and do more things with strings
- `len(name)` used to find the length of a string
- `name.upper()` - to convert string to uppercase
- `name.lower()` - to convert string to lowercase
- `name.find()` - to find the index of characters
- `name.replace()` - to replace one char to another
- `'...' in name` - to check existence of a char in variable

## 5. Operators and Types of Operators

### Arithmetic Operators

- Adding(+), subtract(-), multiple(*), divide (/)
- White spaces are ignored
- Follows default mathematical BODMAS rule `2+2*5 = 12`
- Don't use commas in bw numbers
- `x*=5 => x = x * 5` - called **augmented assignment operator**

### Logical Operators

- Used to combine multiple conditions
- Used with if-else statements a lot
- `and` - both condition is true
- `or` - any one condition is true
- `not` - reverts the boolean value next to it

### Comparison Operators

- Used to compare 2 values (a variable with a value)
- Operators are almost like math
- `>`, `<`, `==` - used to compare which value is **greater**, **lesser**, or **equal**
- `>=` and `<=` - checks the condition **greater than or equal to** and **less than or equal to**

## 6. If-Else Statements

- Used to run programs based on conditions
- Used a lot with comparison operators and boolean values
- If a condition is met, the program runs and discards the rest of the conditions
- Starts with `if (condition)` and continues till the last `else` statement

- `elif` statements are used to created *nested if-else* conditions

```
 # nested if-else statements
if john_age > voting_age:
print('John can vote')
elif john_age == voting_age:
print('Depends on the govt.')
else:
print('John cannot vote')
```

## 7. Loops in Python

### While Loops

- Used to execute a block of code multiple times
- Code block keep reiterating as long as the condition given is true
- Can use used to create patterns and small games
- `else` can be used with while loops

### For Loops

- Used to iterate over a collection - For eg, each character of a string
- In each iteration, loop variable holds one character value at a time
- `for i in 'Python'` - Here `i` is the loop variable and `Python` is iterable string
- Built-in `range()` func is used to loop over a range of numbers
- `range()` func can take arguments in multiple ways

### Nested Loops

- One loop can be nested inside another loop
- Used to create complex programs like getting graph coordinates
- Can be used to create various patterns or complex programs
- Use break and continue statements for further control of loops
    - `break` - stop and exit the loop
    - `continue` - skip over current iteration

## 8. Lists and List Methods

### Basics of Lists

- Used to store a collection of values of different data types
- Items are put inside the square brackets `['John', 'Peter', 'Josh']`
- Single items can be accessed using their index
- We can also get a range of items using index - `names[2:4]`
- Range doesn't modify the list, they return a new list
- List items can also be updated using their index

### 2-D Lists

- Used to create a maths like matrix by nesting multiple lists inside a parent list
- Used a lot in data science and ML

- Each child list represents one row of the parent matrix
- Individual items can be referenced using 2 brackets indexes
- Items can also be modified using the same way `items[2][1]`

### List Methods

- Built-in functions/operations for lists
- `sort()` - to sort the list in ascending order
- `reverse()` - to reverse direction of the list
- `append(n)` - to add n to the end of the list
- `remove(n)` - to remove n from the list
- `list.copy()` - to create a copy of the list that doesn't get affected when the original list is manipulated

## 9. Tuples in Python

- Used to store a collection of items just like lists
- Defined using *parenthesis* - `('John', 'Peter', 'Josh')`
- Immutable - items can't be added or removed
- Single items can be accessed using indexes
- Since they're immutable, there aren't a lot of methods for tuples unlike lists
- `count()` and `index()` can be used to get count and index of the specified item

## Bonus - Unpacking in Python

- Used to destructure list/tuple items into separate individual variables
- Makes it easy to access single items when they are being repeated many times

```
 # a list or a tuple
coordinates = [1, 2, 3]

# □ product of items w/o destructuring
product1 = coordinates[0] * coordinates[1] * coordinates[2]

# □ product of items with destructuring w/o unpacking
a = coordinates[0]
b = coordinates[1]
c = coordinates[2]
product2 = a * b * c

# □ destructuring using unpacking
x, y, z = coordinates
product3 = x * y * z

# All 3 products will return the same value but it can easily be observed that the 3rd way makes things much more simple than the ot
```

## 10. Sets in Python

### Basics of Sets

- Set is a collection of items with no duplicates
- Defined by curly braces - `{1, 2, 3}`
- A list can be converted into set using **set** function
- Sets are unordered so items cannot be accessed using indexes
- Has several built-in methods like `add()`, `remove()`, `len()`, etc.

### Mathematical Operations on Sets

- Sets support powerful mathematical operations
- `set1 | set2` - returns union of 2 sets
- `set1 & set2` - returns intersection of 2 sets
- `set1 - set2` - returns difference b/w 2 sets
- `set1 ^ set2` - returns symmetric difference b/w 2 sets

## 11. Dictionaries in Python

- Used to store a collection of key-value pairs
- Every key must be unique

- Values can be accessed using the `[]` brackets to target specific keys
- Items can also be modified the same way
- New key value pairs can also be added later using `[]` brackets

# 12. Functions in Python

### Basics of Functions

- Block of code that performs a specific task when called and executed
- Makes code more reusable and well designed
- Can be built-in like `print()` or can be manually created
- Created by using the **def** keyword followed by function name and parenthesis
- Cannot be called before defining it
- Can be called as many as required

### Parameters and Arguments

- **Parameters** act as placeholders for info passed to the functions
- **Arguments** are actual information passed into the function when called
- Make functions more dynamic
- We can also have multiple parameters for a single function
- Order/Position of the argument matters

### Keyword Arguments

- Order of the arguments doesn't matter when using keyword argument
- Are used to improve the readability of the code
- Used by writing parameter name followed by the argument value when calling a function
- Must come after the positional arguments if we have both

### Return Statement

- Functions can also return values using return statement
- Useful when performing calculations inside functions
- By default, all functions return the value **None**

# Bonus - Try and Except

- Used to handle errors in python programs
- Main code is written under try block and except block customizes the results based on the type of the error

```
# normal code
try:
  age = int(input('Age: '))
  print(age)
# code when there's a value error
except ValueError:
  print('Please provide a number...')
```

- We can have multiple exceptions (except blocks) to handle different kinds of errors

# 13. Object Oriented Programming

### Classes

- Used to define new types or create blueprint for objects
- Can have their own methods just like strings, lists, dictionaries, etc.
- Every method in a class should have the **self** parameter
- Defined by using *class keyword* followed by a *PascalCased* name by convention - `class EmailClient`

### Objects

- Objects are the instances of a class
- Can have as many instances as we want but **each object is a different instance of the parent class**
- Defined by calling the class and is stored in a variable - `point1 = Point()`
- Have access to all the methods defined in their parent class - `point1.methodName()`
- Can also have attributes along with methods

### Constructors

- A function that gets called at the time of creating an object
- Lets us have pre defined attributes for our objects using constructor parameters

- Defined by using `__init__(self, other_parameters)` method which creates the objects
- Attributes can also be modified later as they're not set in stone by constructors

### Inheritance

- Allows us to reference methods of one class to another
- A parent class has the base methods and the child classes inherit the required methods from that parent class
- Child classes can have their own methods as well along with the inherited ones
- `class Child(Parent)` - *Child* class will inherit all the methods of the *Parent* class

## 14. Modules and Packages

### Modules in Python

- Used to organize code into multiple files where each file is a module
- A module should have all the related functions and classes
- Makes the code more reusable as a module can be imported into other programs too
- Imported module act as an object so the module functions are the methods of that object
- Specific functions can also be directly imported from a module

### Packages in Python

- Another way to organize code when there are a lot of modules
- Can be called as a container for multiple similar modules
- A directory is converted into a package by adding a special `__init__.py` file to it
- Directory should be accurate when importing modules from packages
- Ways to import modules and their function from packages -
    - `import package.module` - imports the whole module
    - `from package.module import func_1, func_2` - imports specific functions from modules

### Built-In Modules

- Python comes with a standard library that contains several modules to perform basic common tasks
- It makes code reusable and we don't have to start from scratch every time
- Search for `python 3 module index` to find the standard library on google
- This will give us list of all the modules already built into python
- We have modules for *date-and-time*, *sending-emails*, *generating-random-values*, and much more

### PyPi and Pip

- Python Package Index ( `PyPi` ) - Directory with tons of cool external packages
- These packages are built by python devs for their projects and then published for the whole community to use
- Not every package is complete or bug free, some might even be in development mode so choose carefully
- Visit pypi.org to find all these packages.

#### Pip

- `pip` - A tool installed with python, used to install external packages from `PyPi`
- `pip install <package_name>` - Command to install a package
- After installing, we can normally import the package and its modules into our program

## Conclusion

Thank you for using this cheatsheet!

If you found it helpful please check out more of my work on yodkwtf.com or follow me on twitter. I also run a small youtube channel called Yodkwtf Academy.

If you'd like a detailed version of this cheatsheet checkout the full repo on github.