Python fundamentals

Welcome👨🏽‍💻👩🏽‍💻

Getting started with Python

Damilola Omifare
Software Engineer

Class Ethics

Please!!!

- No phones unless it is very important.
- No talking please.
- Stop me if you have questions or doubts (*no question is stupid*)
- Have fun learning

Scope

This session is designed to imparting a basic level understanding of variables, control flow, functions in python programming.

Variables

- **Variables** are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.

```
x = 5
```

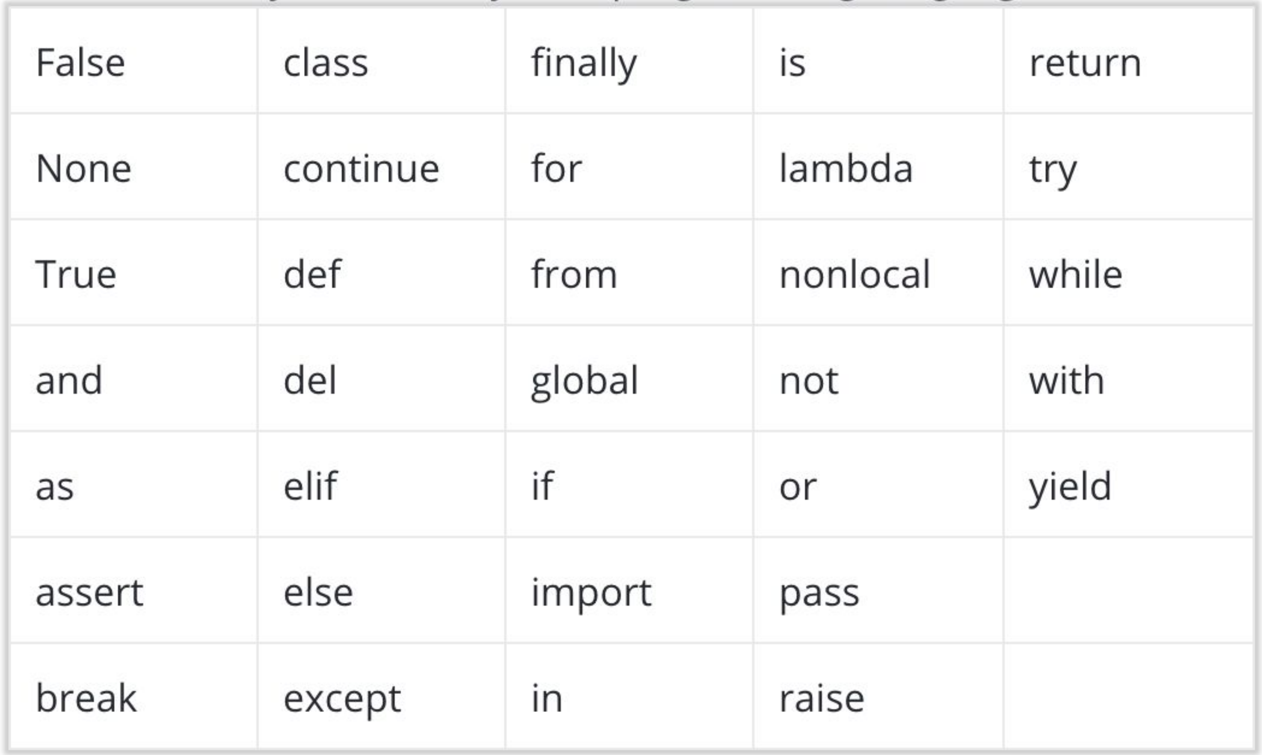
```
y = "John"
```

Python allows you to assign value to multiple variables in one line.

e.g
```
x, y, z = "Orange", "Banana", "Cherry"
```

They are reserved words.

Keywords in Python programming language



False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Data Types

- Integer **1,2,3,4,5**
- floats : 1.2,1.4, 53.5
- complex
- strings : `"Hello World!"`
- booleans : true / false. 1 or 0
- bytes, bytearray, memoryview
- ...

Getting the Data Type: You can get the datatype of any object by using the type() function:

Strings

In python, strings are shown as variable type **str**. You can define a string with either double quotes " or single quotes '.

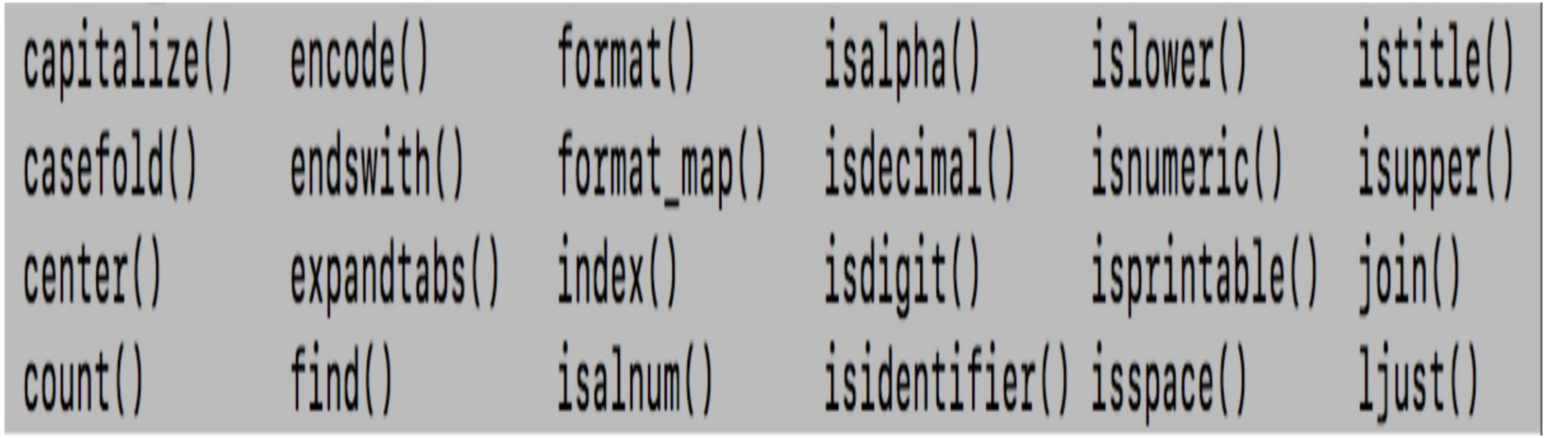
```
>>> my_string = 'this is a string!'

>>> my_string2 = "this is also a string!!!"

>>> this_string = 'David\'s laptop is a macbook'
```

Strings formatting

In python, these are methods you can use on strings.



capitalize()	encode()	format()	isalpha()	islower()	istitle()
casefold()	endswith()	format_map()	isdecimal()	isnumeric()	isupper()
center()	expandtabs()	index()	isdigit()	isprintable()	join()
count()	find()	isalnum()	isidentifier()	isspace()	ljust()

Python also has : **.split(), .format()** methods.

Data Structures

There are four collection data types in the Python programming language:

- **list** is a collection which is ordered and changeable. Allows duplicate members.
- **tuple** is a collection which is ordered and **unchangeable**. Allows duplicate members.
- **set** is a collection which is unordered and unindexed. No duplicate members.
- **dictionary** is a collection which is unordered, changeable and indexed. **No duplicate members.**

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

List

- A list is a container organising data which is ordered and changeable. In Python lists are written with square brackets.
- A list is one of the most common and basic data structures in Python.

```
mylist = ["hello", True, 1, 4.3]
print(mylist)
```

Accessing a list :
You access the list items by referring to the index number.
The number could be positive or negative.

```
print(mylist[0]) // hello
print(mylist[1]) // True
print(mylist[-1]) // 4.3
```

Slicing and dicing with List

- When using slicing, it is important to remember that the `lower` index is **`inclusive`** and the `upper` index is **`exclusive`**.

```
mylist = ["hello", True, 1, 4.3]
print([0:1]) // Expected output is hell
```

```
print(mylist[len(mylist)]) // throws an erro
print(mylist[len(mylist)] - 1) //  4.3
```

List method

- `len()` returns how many elements are in a list.
- `max()` returns the greatest element of the list. How the greatest element is
- `min()` returns the smallest element in a list. min is the opposite of max, which returns the largest element in a list.
- `sorted()` returns a copy of a list in order from smallest to largest, leaving the list unchanged.

These operations can be performed on a list: `join`, `append, pop, count, reverse, copy, clear, remove, delete …`

`From more visit :`

[`https://docs.python.org/3/tutorial/datastructures.html`](https://docs.python.org/3/tutorial/datastructures.html)

Loop through a list

You can loop through the list items by using a `for` loop:

```
mylist = ["apple", "python", "banana", "cherry"]
for x in mylist:
  print(x)
```

Python list comprehension.

- List comprehensions provide a concise way to create lists

```
mylist = ["apple", "python", "banana", "cherry"]
print([i for i in mylist])
```

Tuple

- It's a data type, **immutable** ordered sequences of elements.
- Elements can be mixed.

```
dimension = 52, 40, 100
length, width, height = dimension

print("The dimension are {} x {} x {}".format(length, width, height))
length, width = size

print("The size are {} x {}".format(length, width))
```

The parentheses are optional when defining tuples. It is know that programmers do frequently omit them if parentheses don't clarify the code.

Tuple

- provides a convenient way to **swap** variable values.
- used to **return more than one value** from a function
- You can **iterate** over a tuple

```
(a ,b) = (b, a)
```

```
a = b
b = a
```

```
hello = a
a = b
b = hello
```

```
def multipy and_addition (x, y):
    add = x + y
    multipy = x * y
    return (add, multipy)

(added, multiplied) =
multipy_and_addition(2,6)
```

return more than one value

Set

- A **set** is a data type, **mutable** unordered collections of **unique** elements.
- No duplicate members

```
numbers = [1,4, 4, 2, 6, 3, 1, 1, 6]
get_unique_number = set(numbers)

print(get_unique_number)



fruit = {"apple", "banana", "mango", "grapes", "watermelon"}

print(fruit)
```

More on Strings

- can **number, letter, spaces, special characters** like @
- enclose in **single or double quotation mark**. Never mixed them up

```
hi ="Hi there."
```

```
hi ='Hi there."
```

DON'T DO THIS

- **string concatenation**

```
her_name = "Pythoniana"
salute = "Hi there."
greetings = her_name + " " + salute
```

- **Operation** can be done on **strings** like, `her_name * 3`

INPUT : `input("")`

- type a description inside the quotation marks as this is printed.
- user types something and hit enter
- bind input to a variable / assign it to a variable.
- input **takes in the values as string**, so it must be casted when **working with floats, or integers (numbers)**

```
user_input = input("What is your name...")
print(user_input, "Hi there.")
```

INPUT : input("")

- type a description inside the quotation marks as this is printed.
- user types something and hit enter
- bind input to a variable / assign it to a variable.
- input **takes in the values as string**, so it must be casted when **working with floats, or integers (numbers)**

```
user_input = input("What is your name...")
print(user_input, "Hi there.")
```