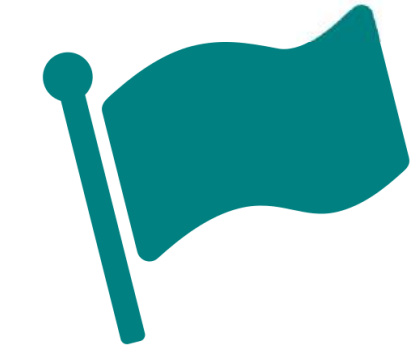
lightweight learn python

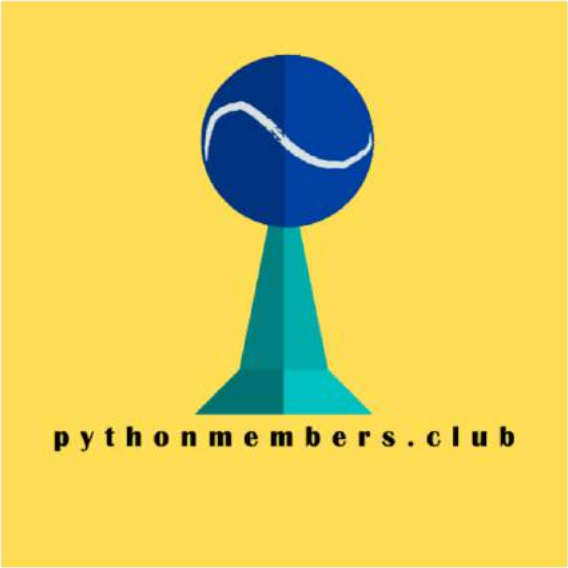


a lightweight guide to
a heavily used language

draft 1

dedicated to friends, families and
all python users, particularly the
core devs and all users of the official
mailing list

an initiative of



www.pythonmembers.club

liked it or suggestions? contact

Abdur-Rahmaan Janhangeer
arj.python@gmail.com
Mauritius

COMMENTS

comments are used to provide more information but are not executed

sometimes comments are used to generate documentations by special libraries or tools

file

```
# -*- coding: utf-8 -*-
# the above is a special comment stating unicode support

# comment but for single line
# comment
# comment

"""
multi-line comment but more used for
documentation
"""

'''
same as above but with single quote
'''
```

DATA TYPES

some basic types of data supported by python

integers, floats and strings are the basic data types supported by python

shell

```
In [3]: 1
Out[3]: 1          integer

In [4]: 1.0
Out[4]: 1.0        float

In [5]: "1.0"
Out[5]: '1.0'      string

In [6]: 'abc'
Out[6]: 'abc'      string
```

PRINT

print is used to output characters to the screen

text numbers or unicode characters

shell

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> print(1)
1
> print(1.0)
1.0
> print("abcd")
abcd
> print(1+1)
2
```

```
In [2]: print("👆")
👆
```

also written as

file

```
print(1)
print(1.0)
print("1.0")
```

STRINGS

adding strings produces longer ones

shell

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> 'a'
=> 'a'
> "a"
=> 'a'
> 'a' + 'b'
=> 'ab'
> 'abc' + 'def'
=> 'abcdef'
> 'a' * 2
=> 'aa'
> 'abc' * 4
=> 'abcabcabcabc'
> 'abc' + "def"
=> 'abcdef'
```

OPERATIONS

python supports arithmetic operations, just like maths

shell

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> 1 + 1   addition
=> 2
> 10 - 2   substraction
=> 8
> 1 * 10   multiplication
=> 10
> 2 * 10
=> 20
> 10 / 2   division
=> 5.0
> 10 // 2   floor/rounded division
=> 5
> 2 ** 2   exponentiation/
=> 4
> 2 ** 2 ** 2   to the power of
=> 16
```

VARIABLES

variables are declared just like maths

x = 1

shell

```
In [13]: x = 1

In [14]: print(x)
1

In [15]: x = x + 1

In [16]: print(x)
2

In [17]: x += 1

In [18]: print(x)
3

In [19]: y = 4

In [20]: print( x + y + 1)
8

In [21]: z = x + y

In [22]: print(z)
7
```

same

CONDITIONALS AND EQUALITY

objects can be checked whether they are equal or not, whether they are greather than or not

shell

```
In [1]: if 1 == 1: print("one equals one")
one equals one
In [2]: if 1 == 2 : print("one equals two")

In [3]: if 1 != 2 : print("one is not equal to two")
one is not equal to two

In [4]: if 1 > 2: print("1 greater than 2")

In [5]: if 1 >= 1: print("one greater equals one")
one greater equals one

In [6]:
```

equal to

not equal to

greater

greater equal to

also written as

file

```
# -*- coding: utf-8 -*-

if 1 == 1:
    print('one equals one')

if 1 == 2:
    print('one equals two') # does not execute

if 1 != 2:
    print('one is not equal to two')

if 1 > 2:
    print('one is greater than two')

if 1 >= 1:
    print('one greater equals one')
```

strings can also be compared

shell

```
In [7]: if 'a' == 'b': print(1)

In [8]: if 'a' == 'a': print(1)
1
```

FUNCTIONS

instead of writing the same thing over and over again, functions allow us to specify a block of code by using a name

file

```
print(1)
print(2)
print(3)

print(1)
print(2)
print(3)

print(1)
print(2)
print(3)
```

replaced by

file

```
def print_nums():
    print(1)
    print(2)
    print(3)

print_nums()
print_nums()
print_nums()
```

def used to define a functtion

shell

```
In [7]: def print_country(): print("mauritius")

In [8]: print_country()
mauritius

In [9]: def add_two(x): print(x+2)

In [10]: add_two(5)
7

In [11]: def add_two(x): return x + 2

In [12]: print(add_two(10))
12
```

also written as

file

```
# -*- coding: utf-8 -*-

def print_country():
    print("mauritius")

def add_two(x):
    print(x+2)
add_two(5) # prints 7

def add_two(x):
    return x+2

print(add_two(5)) # prints 7
```

CLASSES

classes allow us to represent real world objects as we think of them

attribute

file

```
class Car:
    wheels = 4

    def move(self):
        print('moving')
```

method

valourdigest.com

wheels is an attribute while move is a method

shell

```
In [23]: class Boy: pass

In [24]: Boy.name = 'a'

In [25]: Boy.age = 10

In [26]: Boy.address = "vacoas"

In [27]: print(Boy.name, Boy.address)
a vacoas
```

OOP (OBJECT-ORIENTED PROGRAMMING)

making use of classes in your code is called object oriented programming

a class defines the object

file

```
class Plane:
    def __init__(self):
        self.owner = 'someone'

    def take_off(self):
        print('taking off')
```

actually creating the object is called instantiation

boeing = Plane()

file

```
boeing = Plane()

boeing.take_off()

print(boeing.owner)
```

the output of the above

```
taking off
someone
```

THE STANDARD LIBRARY

python provides some useful futilities which can be imported

```
letters = 'abcdefghijklmnopqrstuvwxyz'
print(letters)
```

instead of writing by hand

```
import string

letters = string.ascii_lowercase
print(letters)
```

some maths

```
import math

print(math.sin(100), math.cos(100))
```

end of draft 1
Monday 10 sept 2018

BTC

38hU4b1ToHHmaPQHeTacLZ9yXSc
Rty8f2j

the book shall remain free forever.
if you want to adapt and sell, contact
for permissions and supportive royalties