# 006

type()

# 008

help("keywords")

a, b, c = 1, 2, 3

# 011

name = '''First

Second 'Test' "Test"

Third'''

name = """First

Second "Test" \\\ 'Test'

Third"""

# 012

# [Start:End:Steps] :

neme = "I Love Python"

name[0]

name[9]

name[-1]

name[-6]

neme[8:11]

neme[3:5]

neme[:10]

neme[5:]

neme[:]

name[0::1]

name[::1]

name[::2]

name[::3]

# 020

# Arithmetic Operators :

# +

# -

# \*

# /

# %

# \*\*

# //

# 021

name = ["One", "Two", "One", 1, 100.5, True]

name[1] = 2

name[-1] = False

name[0:3] = ["A"]

# 024

name = (1, 2, 3, 4, 5)

print(name[0])

print(name[-1])

print(name[-3])

# 025

name = ("Osama",)

name = "Osama",

x, y, \_, z = ("A", "B", 4, "C")

# 030

user = {

  "name": "Osama",

  "age": 36,

  "country": "Egypt",

  "skills": ["Html", "Css", "JS"],

  "rating": 10.5

}

user["country"]

user.get("country")

user.keys()

user.values()

languages = {

"One": {

"name": "Html",

    "progress": "80%"

},

"Two": {

    "name": "Css",

    "progress": "90%"

  },

"Three": {

"name": "Js",

    "progress": "90%"

  }

}

languages["One"]

languages["Three"]["name"]

# 033

print(100 > 200)

print(100 > 100)

print(100 > 90)

# True Values

print(bool("Osama"))

print(bool(100))

print(bool(100.95))

print(bool(True))

print(bool([1, 2, 3, 4, 5]))

# False Values

print(bool(0))

print(bool(""))

print(bool(''))

print(bool([]))

print(bool(False))

print(bool(()))

print(bool({}))

print(bool(None))

# 034

# Boolean Operators :

# and

# or

# not

# 035

# Assignment Operators :

# =

# +=

# -=

# \*=

# /=

# \*\*=

# %=

# //=

# 036

# Comparison Operators :

# [ == ]

# [ != ]

# [ > ]

# [ < ]

# [ >= ]

# [ <= ]

# 037

# Type Conversion :

# str()

# tuple()

# list()

# set()

# dict()

name = (("A", 1), ("B", 2), ("C", 3))

print(dict(name))

# 038

input('What\'s Is Your First Name?')

# 041

# if, elif, else

# 043

movieRate = 18

age= 18

print("Movie Is Not Good 4U" if age < movieRate else "Movie S Good 4U And Happy Watching")

# 045

# Membership Operators :

# in

# not in

# 054

# break, continue, pass

# 055

Skills = {

"HTML": "80%",

"CSS": "90%",

  "JS": "70%",

  "PHP": "80%"

}

for skill\_key, skill\_progress in Skills.items():

print(f"{skill\_key} => {skill\_progress}")

Skills = {

"HTML": {

"Main": "80%",

"Pugjs": "80%"

},

"CSS": {

    "Main": "90%",

    "Sass": "70%"

}

}

for main\_key, main\_value in Skills.items():

print(f"{main\_key} Progress Is: ")

for child\_key, child\_value in main\_value.items():

print(f"- {child\_key} => {child\_value}")

# 057

def say\_hello(name):

print(f"Hello {name}")

say\_hello("Ahmed")

# def => Function Keyword [Define]

# say\_hello() => Function Name

# name => Parameter

# print(f"Hello {name}") => Task

# say\_hello("Ahmed") => Ahmed is The Argument

# 058

def show\_details(name, \*skills):

print(f"Hello {name} Your Skills Is: ")

    for skill in skills:

print(skill)

show\_details("Osama", "Html", "CSS", "JS")

# 059

def say\_hello(name="Unknown", age="Unknown", country="Unknown"):

print(f"Hello {name} Your Age is {age} and Your Country Is {country}")

say\_hello("Osama", 36, "Egypt")

say\_hello("Mahmoud", 28, "KSA")

say\_hello("Sameh", 38)

say\_hello("Ramy")

say\_hello()

# 060

mySkills = {

"Html": "80%",

  "Css": "70%",

  "Js": "50%",

  "Python": "80%",

  "Go": "40%"

}

def show\_skills(\*\*skills):

for skill, value in skills.items():

    print(f"{skill} => {value}")

show\_skills(\*\*mySkills)

# 062

def one():

global x

    x = 2

# 064

hello = lambda name, age : f"Hello {name} your Age Is: {age}"

print(hello("Ahmed", 36))

# 065

file = open(file , mode)

file = open(R"D:\Python\Files\osama.txt")

# 066

myFile = open("D:\Python\Files\osama.txt", "r")

print(myFile)

print(myFile.name)

print(myFile.mode)

print(myFile.encoding)

print(myFile.read())

print(myFile.read(5))

print(myFile.readline(5))

print(myFile.readline())

print(myFile.readline())

myFile.close()

# 067

myFile = open("D:\Python\Files\osama.txt", "w")

myFile.write("Hello\n")

myFile.write("Third Line")

myFile.write("Elzero Web School\n" \* 1000)

myList = ["Oasma\n", "Ahmed\n", "Sayed\n"]

myFile.writelines(myList)

myFile = open("D:\Python\Files\osama.txt", "a")

myFile.write("Elzero")

# 068

myFile = open("D:\Python\Files\osama.txt", "a")

myFile.truncate(5)

print(myFile.tell())

myFile = open("D:\Python\Files\osama.txt", "r")

myFile.seek(11)

print(myFile.read())

# 076

import random

print(random)

print(dir(random))

from random import randint, random

print(f"Print Random Float {random()}")

print(f"Print Random Integer {randint(100, 900)}")

# 077

import sys

sys.path.append(R"D:\Python\Files")

print(sys.path)

import elzero as ee

from elzero import sayHello as ss

# 078

import termcolor

import pyfiglet

print(pyfiglet.figlet\_format("Elzero"))

print(termcolor.colored("Elzero", color="yellow"))

print(termcolor.colored(pyfiglet.figlet\_format("Elzero"), color="yellow"))

# 079

import datetime

print(datetime.datetime.now())

print(datetime.datetime.now().year)

print(datetime.datetime.now().month)

print(datetime.datetime.now().day)

print(datetime.datetime.now().hour)

print(datetime.datetime.now().minute)

print(datetime.datetime.now().second)

print(datetime.datetime.now().microsecond)

print(datetime.datetime.now().date().year)

print(datetime.datetime.now().date().month)

print(datetime.datetime.now().date().day)

print(datetime.datetime.now().time())

print(datetime.datetime.now().time().hour)

print(datetime.datetime.now().time().minute)

print(datetime.datetime.now().time().second)

print(datetime.datetime.now().time().microsecond)

print(datetime.datetime.min)

print(datetime.datetime.max)

print(datetime.time.min)

print(datetime.time.max)

print(datetime.date.min)

print(datetime.date.max)

print(datetime.datetime(1982, 10, 25))

print(datetime.datetime(1982, 10, 25, 10, 45, 55, 150364))

# 080

import datetime

# https://strftime.org/

date = datetime.datetime(1982, 10, 25)

print(date.strftime("%a"))

# 081

iterable = "Osama"

iterator = iter(iterable)

print(next(iterator))

print(next(iterator))

print(next(iterator))

print(next(iterator))

print(next(iterator))

for letter in iter("Elzero"):

    print(letter, end=" ")

# 082

def myGenerator():

  yield 1

  yield 2

  yield 3

  yield 4

myGen = myGenerator()

print(next(myGen))

print(next(myGen))

print(next(myGen))

print(next(myGen))

for number in myGen:

  print(number)

# 083

def myDecorator(func):

    print("Before")

    func()

    print("After")

@myDecorator

def sayHello():

    print("Hello From Say Hello Function")

@myDecorator

def sayHowAreYou():

    print("Hello From Say How Are You Function")

# 084

def myDecorator(func):

  def nestedFunc(num1, num2):

    if num1 < 0 or num2 < 0:

      print("Beware One Of The Numbers Is Less Than Zero")

    func(num1, num2)

  return nestedFunc

def myDecoratorTwo(func):

  def nestedFunc(num1, num2):

    print("Coming From Decorator Two")

    func(num1, num2)

  return nestedFunc

@myDecorator

@myDecoratorTwo

def calculate(n1, n2):

  print(n1 + n2)

calculate(-5, 90)

# 085

def myDecorator(func):

  def nestedFunc(\*numbers):

    for number in numbers:

      if number < 0:

        print("Beware One Of The Numbers Is Less Than Zero")

    func(\*numbers)

  return nestedFunc

@myDecorator

def calculate(n1, n2, n3, n4):

  print(n1 + n2 + n3 + n4)

calculate(-5, 90, 50, 150)

from time import time

def speedTest(func):

  def wrapper():

    start = time()

    func()

    end = time()

    print(f"Function Running Time Is: {end - start}")

  return wrapper

@speedTest

def bigLoop():

  for number in range(1, 2000):

    print(number)

bigLoop()

# 086

list1 = [1, 2, 3, 4, 5]

list2 = ["A", "B", "C", "D"]

tuple1 = ("Man", "Woman", "Girl", "Boy")

dict1 = {"Name": "Osama", "Age": 36, "Country": "Egypt", "Skill": "Python"}

for item1, item2, item3, item4 in zip(list1, list2, tuple1, dict1):

  print("List 1 Item =>", item1)

  print("List 2 Item =>", item2)

  print("Tuple 1 Item =>", item3)

  print("Dict 1 Key =>", item4, "Value =>", dict1[item4])

# 087

from PIL import Image

myImage = Image.open("D:\Python\Files\game.jpg")

myImage.show()

myBox = (300, 300, 800, 800)

myNewImage = myImage.crop(myBox)

myNewImage.show()

myConverted = myImage.convert("L")

myConverted.show()

# 088

def elzero\_function(name):

  """

  Elzero Function

    It Say Hello From Elzero

  Parameter:

    name => Person Name That Use Function

  Return:

    Return Hello Message To The Person

  """

  print(f"Hello {name} From Elzero")

elzero\_function("Ahmed")

print(dir(elzero\_function))

print(elzero\_function.\_\_doc\_\_)

help(elzero\_function)

# 089

# pip install pylint

# pylint.exe D:\Python\File\elzero.py

"""

This is My Module

To Create Function

To Say Hello

"""

def say\_hello(name):

  '''This Function Only Say Hello To Someone'''

  msg = "Hello"

  return f"{msg} {name}"

say\_hello("Ahmed")

# 090

x = -10

if x < 0:

  raise Exception(f"The Number {x} Is Less Than Zero")

  print("This Will Not Print Because The Error")

else:

  print(f"{x} Is Good Number and Ok")

print('Print Message After If Condition')

y = "Osama"

if type(y) != int:

  raise ValueError("Only Numbers Allowed")

print('Print Message After If Condition')

# 091

try:

  number = int(input("Write Your Age: "))

  print("Good, This Is Integer From Try")

except:

  print("Bad, This is Not Integer")

else:

  print("Good, This Is Integer From Else")

finally:

  print("Print From Finally Whatever Happens")

try:

  print(10 / 0)

  print(x)

  print(int("Hello"))

except ZeroDivisionError:

  print("Cant Divide")

except NameError:

  print("Identifier Not Found")

except ValueError:

  print("Value Error Elzero")

except:

  print("Error Happens")

# 093

# Debugging Code

# 095

# <https://pythex.org>

# <https://www.debuggex.com/cheatsheet/regex/python>

# <https://regex101.com>

# 094

def say\_hello(name) -> str:

  print(f"Hello {name}")

say\_hello("Ahmed")

def calculate(n1, n2) -> str:

  print(n1 + n2)

calculate(10, 40)

# 100

import re

is\_email = re.search(r"[A-z0-9\.]+@[A-z0-9]+\.(com|net)", "os@osama.com")

print(is\_email)

print(is\_email.span())

print(is\_email.string)

print(is\_email.group())

if is\_email:

  print("This is A Valid Email")

  print(is\_email.span())

  print(is\_email.string)

  print(is\_email.group())

else:

  print("This is Not A Valid Email")

email\_input = input("Please Write Your Email: ")

search = re.findall(r"[A-z0-9\.]+@[A-z0-9]+\.com|net", email\_input)

empty\_list = []

if search != []:

  empty\_list.append(search)

  print("Email Added")

else:

  print("Invalid Email")

for email in empty\_list:

  print(email)

# 101

import re

string = "How-To\_Write\_A\_Very-Good-Article"

search = re.split(r"-|\_", string,2)

print(search)

for counter, word in enumerate(search, 1):

  if len(word) == 1:

    continue

  print(f"Word Number: {counter} => {word.lower()}")

my\_string = "I Love Python"

print(re.sub(r"\s", "-", my\_string, 1))

# 102

import re

my\_web = "https://www.elzero.org:8080/category.php?article=105?name=how-to-do"

search = re.search(r"(https?)://(www)?\.?(\w+)\.(\w+):?(\d+)?/?(.+)", my\_web)

print(search.group())

print(search.groups())

print(f"Protocol: {search.group(1)}")

print(f"Sub Domain: {search.group(2)}")

print(f"Domain Name: {search.group(3)}")

print(f"Top Level Domain: {search.group(4)}")

print(f"Port: {search.group(5)}")

print(f"Query String: {search.group(6)}")

# 104

class Member:

  def \_\_init\_\_(self):

    print("A New Member Has Been Added")

member\_one = Member()

member\_two = Member()

member\_three = Member()

print(member\_one.\_\_class\_\_)

# 105

class Member:

  def \_\_init\_\_(self, first\_name, middle\_name, last\_name):

    self.fname = first\_name

    self.mname = middle\_name

    self.lname = last\_name

member\_one = Member("Osama", "Mohamed", "Elsayed")

member\_two = Member("Ahmed", "Ali", "Mahmoud")

member\_three = Member("Mona", "Ali", "Mahmoud")

print(member.fname, member\_one.mname, member\_one.lname)

# 106

class Member:

  def \_\_init\_\_(self, first\_name, middle\_name, last\_name, gender):

    self.fname = first\_name

    self.mname = middle\_name

    self.lname = last\_name

    self.gender = gender

  def full\_name(self):

    return f"{self.fname} {self.mname} {self.lname}"

  def name\_with\_title(self):

    if self.gender == "Male":

      return f"Hello Mr {self.fname}"

    elif self.gender == "Female":

      return f"Hello Miss {self.fname}"

    else:

      return f"Hello {self.fname}"

  def get\_all\_info(self):

    return f"{self.name\_with\_title()}, Your Full Name Is: {self.full\_name()}"

member\_one = Member("Osama", "Mohamed", "Elsayed", "Male")

member\_two = Member("Ahmed", "Ali", "Mahmoud", "Male")

member\_three = Member("Mona", "Ali", "Mahmoud", "Female")

print(member\_one.full\_name())

print(member\_one.name\_with\_title())

print(member\_one.get\_all\_info())

# 107

class Member:

  not\_allowed\_names = ["Hell", "Shit", "Baloot"]

  users\_num = 0

  def \_\_init\_\_(self, first\_name, middle\_name, last\_name, gender):

    self.fname = first\_name

    self.mname = middle\_name

    self.lname = last\_name

    self.gender = gender

    Member.users\_num += 1

  def full\_name(self):

    if self.fname in Member.not\_allowed\_names:

      raise ValueError("Name Not Allowed")

    else:

      return f"{self.fname} {self.mname} {self.lname}"

  def name\_with\_title(self):

    if self.gender == "Male":

      return f"Hello Mr {self.fname}"

    elif self.gender == "Female":

      return f"Hello Miss {self.fname}"

    else:

      return f"Hello {self.fname}"

  def get\_all\_info(self):

    return f"{self.name\_with\_title()}, Your Full Name Is: {self.full\_name()}"

print(Member.users\_num)

member\_three = Member("Mona", "Ali", "Mahmoud", "Female")

member\_four = Member("Shit", "Hell", "Metal", "DD")

print(Member.users\_num)

print(member\_one.full\_name())

print(member\_one.name\_with\_title())

print(member\_one.get\_all\_info())

# 108

class Member:

  users\_num = 0

  @classmethod

  def show\_users\_count(cls):

    print(f"We Have {cls.users\_num} Users In Our System.")

  @staticmethod

  def say\_hello():

    print("Hello From Static Method")

  def \_\_init\_\_(self, first\_name, middle\_name, last\_name, gender):

    self.fname = first\_name

    self.mname = middle\_name

    self.lname = last\_name

    self.gender = gender

    Member.users\_num += 1

def full\_name(self):

    return f"{self.fname} {self.mname} {self.lname}"

member\_three = Member("Mona", "Ali", "Mahmoud", "Female")

member\_four = Member("Shit", "Hell", "Metal", "DD")

print(Member.users\_num)

Member.show\_users\_count()

print(member\_one.full\_name())

print(Member.full\_name(member\_one))

Member.say\_hello()

# 109

class Skill:

  def \_\_init\_\_(self):

    self.skills = ["Html", "Css", "Js"]

  def \_\_str\_\_(self):

    return f"This is My Skills => {self.skills}"

  def \_\_len\_\_(self):

    return len(self.skills)

profile = Skill()

print(profile)

print(len(profile))

profile.skills.append("PHP")

profile.skills.append("MySQL")

print(len(profile))

my\_string = "Osama"

print(type(my\_string))

print(my\_string.\_\_class\_\_)

print(dir(str))

print(str.upper(my\_string))

# 110

class one :

    def \_\_init\_\_ (self,name\_pa1):

        self.name1 = name\_pa1

    def say\_hello (self):

        print(f"Hello {self.name1}")

class two(one) :

    def \_\_init\_\_ (self):

        # one.\_\_init\_\_(self,'bloote')

        super().\_\_init\_\_("Elzero")

member1=one("Osama")

member2=two()

member1.say\_hello()

member2.say\_hello()

# 111

class one :

    def \_\_init\_\_ (self,name\_pa1):

        self.name1 = name\_pa1

    def say\_hello (self):

        print(f"Hello {self.name1}")

class two(one) :

    def \_\_init\_\_ (self,name\_pa2):

        self.name2 = name\_pa2

        one.\_\_init\_\_(self,self.name2)

        # super().\_\_init\_\_(self.name2)

member1=one("Osama")

member2=two("Elzero")

member1.say\_hello()

member2.say\_hello()

class BaseOne:

  def \_\_init\_\_(self):

    print("Base One")

  def func\_one(self):

    print("One")

class BaseTwo:

  def \_\_init\_\_(self):

    print("Base Two")

  def func\_two(self):

    print("Two")

class Derived(BaseOne, BaseTwo) : pass

my\_var = Derived()

print(Derived.mro())

print(my\_var.func\_one)

print(my\_var.func\_two)

my\_var.func\_one()

my\_var.func\_two()

class Base : pass

class DerivedOne(Base) : pass

class DerivedTwo(DerivedOne) : pass

# 113

class Member:

  def \_\_init\_\_(self, name):

    self.name = name  # Public

one = Member("Ahmed")

print(one.name)

one.name = "Sayed"

print(one.name)

class Member:

  def \_\_init\_\_(self, name):

    self.\_name = name  # Protected

one = Member("Ahmed")

print(one.\_name)

one.\_name = "Sayed"

print(one.\_name)

class Member:

  def \_\_init\_\_(self, name):

    self.\_\_name = name  # Private

  def say\_hello(self):

    return f"Hello {self.\_\_name}"

one = Member("Ahmed")

# print(one.\_\_name)

print(one.say\_hello())

print(one.\_Member\_\_name)

# 114

class Member:

  def \_\_init\_\_(self, name):

    self.\_\_name = name  # Private

  def say\_hello(self):

    return f"Hello {self.\_\_name}"

  def get\_name(self):  # Getter

    return self.\_\_name

  def set\_name(self, new\_name):  # Setter

    self.\_\_name = new\_name

one = Member("Ahmed")

one.\_Member\_\_name = "Sayed"

print(one.\_Member\_\_name)

print(one.get\_name())

one.set\_name('Abbas')

print(one.get\_name())

# 115

class Member:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def say\_hello(self):

    return f"Hello {self.name}"

  @property

  def age\_in\_days(self):

    return self.age \* 365

one = Member("Ahmed", 40)

print(one.name)

print(one.age)

print(one.say\_hello())

# print(one.age\_in\_days())

print(one.age\_in\_days)

class Member:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def say\_hello(self):

    return f"Hello {self.name}"

  @property

  def age\_in\_days(self):

    return self.age \* 365

one = Member("Ahmed", 40)

print(one.name)

print(one.age)

print(one.say\_hello())

# print(one.age\_in\_days())

print(one.age\_in\_days)

# 116

from abc import ABCMeta, abstractmethod

class Programming(metaclass=ABCMeta):

  @abstractmethod

  def has\_oop(self):

    pass

  @abstractmethod

  def has\_name(self):

    pass

class Python(Programming):

  def has\_oop(self):

    return "Yes"

  def has\_name(self):

    return "Python"

class Pascal(Programming):

  def has\_oop(self):

    return "No"

  def has\_name(self):

    return "Pascal"

one = Pascal()

print(one.has\_oop())

print(one.has\_name())

# 118

import sqlite3

db = sqlite3.connect("app.db")

db.execute("create table if not exists skills (name text, progress integer, user\_id integer)")

db.close()

# 119

import sqlite3

db = sqlite3.connect("app.db")

cr = db.cursor()

cr.execute("create table if not exists users (user\_id integer, name text)")

cr.execute("create table if not exists skills (name text, progress integer, user\_id integer)")

cr.execute("insert into users(user\_id, name) values(1, 'Ahmed')")

cr.execute("insert into users(user\_id, name) values(2, 'Sayed')")

cr.execute("insert into users(user\_id, name) values(3, 'Osama')")

my\_list = ["Ahmed", "Sayed", "Mahmoud", "Ali", "Kamel", "Ibrahim", "Enas"]

for key, user in enumerate(my\_list,1):

    cr.execute(f"insert into users(user\_id, name) values({key}, '{user}')")

db.commit()

db.close()

# 120

cr.execute("select \* from skills")

print(cr.fetchone())

print(cr.fetchone())

print(cr.fetchone())

print(cr.fetchall())

print(cr.fetchmany(2))

# 122

cr.execute("update users set name = 'Mahmoud' where user\_id = 1")

cr.execute("delete from users where user\_id = 4")

# 124

cr.execute(f"delete from skills where name = '{sk}' and user\_id = '{userid}'")

# 126

cr.execute(f"select name from skills where name = '{sk}' and user\_id = '{uid}'")

# 127

cr.execute("insert into skills values('php', 'Python', 'sqlite')")

cr.execute("insert into skills values(?, ?, ?)", my\_tuple)

cr.execute("select \* from skills order by name asc") # desc

cr.execute("select \* from skills order by name limit 3 offset 2")

cr.execute("select \* from skills where user\_id not in(1, 2, 3)")

# 128

print(\_\_name\_\_)

# 129

import timeit

print(timeit.timeit("'Elzero' \* 1000"))

name = "Elzero"

print(timeit.timeit("name = 'Elzero'; name \* 1000"))

print(timeit.timeit(stmt="random.randint(0, 50)", setup="import random"))

print(timeit.repeat(stmt="random.randint(0, 50)", setup="import random", repeat=4))

# 130

import logging

logging.basicConfig(filename="my\_app.log",filemode="a",format="(%(asctime)s) | %(name)s | %(levelname)s => '%(message)s'",datefmt="%d - %B - %Y, %H:%M:%S")

my\_logger = logging.getLogger("Elzero")

my\_logger.warning("This Is Warning Message") # logging.warning("This Is Warning Message")

# 131

import unittest

assert 3 \* 8 == 24, "Should Be 24"

def test\_case\_one():

  assert 5 \* 10 == 50, "Should Be 50"

def test\_case\_two():

  assert 5 \* 50 == 250, "Should Be 250"

if \_\_name\_\_ == "\_\_main\_\_":

  test\_case\_one()

  test\_case\_two()

  print("All Tests Passed")

class MyTestCase(unittest.TestCase):

  def test\_one(self):

    self.assertTrue(100 > 99, "Should Be True")

  def test\_two(self):

    self.assertEqual(40 + 60, 100, "Should Be 100")

  def test\_three(self):

    self.assertGreater(100, 101, "Should Be True")

if \_\_name\_\_ == "\_\_main\_\_":

  unittest.main()

# 132

import string

import random

print(string.digits)

print(string.ascii\_letters)

print(string.ascii\_lowercase)

print(string.ascii\_uppercase)

def make\_serial(count):

    all\_chars = string.ascii\_letters + string.digits

    chars\_count = len(all\_chars)

    serial\_list = []

    while count > 0:

      random\_number = random.randint(0, chars\_count - 1)

      random\_character = all\_chars[random\_number]

      serial\_list.append(random\_character)

      count -= 1

    print("".join(serial\_list))

make\_serial(10)

# 142

import numpy as np

print(np.\_\_version\_\_)

# 143

import numpy as np

my\_list = [1, 2, 3, 4, 5]

my\_array = np.array(my\_list)

print(type(my\_list))

print(type(my\_array))

a = np.array(10)

b = np.array([10, 20])

c = np.array( [ [1, 2], [3, 4] ] )

d = np.array( [ [ [5, 6], [7, 9] ], [ [1, 3], [4, 8] ] ] )

print(d[1][1][-1])

print(d[1, 1, -1])

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)

my\_custom\_array = np.array([1, 2, 3], ndmin=3)

print(my\_custom\_array)

print(my\_custom\_array.ndim)

print(my\_custom\_array[0, 0, 0])

# 144

print(id(my\_list[0]))

print(id(my\_list[1]))

print(id(my\_array[0]))

print(id(my\_array[1]))

my\_list\_of\_data = [1, 2, "A", "B", True, 10.50]

my\_array\_of\_data = np.array([1, 2, "A", "B", True, 10.50])

print(my\_list\_of\_data)

print(my\_array\_of\_data)

print(type(my\_list\_of\_data[0]))

print(type(my\_array\_of\_data[0]))

# 145

import numpy as np

import time

import sys

elements = 150000

my\_list1 = range(elements)

my\_list2 = range(elements)

my\_array1 = np.arange(elements)

my\_array2 = np.arange(elements)

list\_start = time.time()

list\_result = [(n1 + n2) for n1, n2 in zip(my\_list1, my\_list2)]

print(f"List Time: {time.time() - list\_start}")

array\_start = time.time()

array\_result = my\_array1 + my\_array2

print(f"Array Time: {time.time() - array\_start}")

my\_array = np.arange(100)

print(my\_array.itemsize)

print(my\_array.size)

print(f"All Bytes: {my\_array.itemsize \* my\_array.size}")

my\_list = range(100)

print(sys.getsizeof(my\_list[1]))

print(len(my\_list))

print(f"All Bytes: {sys.getsizeof(1) \* len(my\_list)}")

# 146

import numpy as np

a = np.array(["A", "B", "C", "D", "E", "F"])

print(a.ndim)

print(a[1])

print(a[1:4])

print(a[:4])

print(a[2:])

print(a[2::2])

b = np.array([["A", "B", "X"], ["C", "D", "Y"], ["E", "F", "Z"], ["M", "N", "O"]])

print(b[:3,:2])

print(b[2:,0])

print(b[2:,:2:2])

# 147

# https://numpy.org/devdocs/user/basics.types.html

# https://docs.scipy.org/doc/numpy/reference/arrays.dtypes.html#specifying-and-constructing-data-types

import numpy as np

my\_array1 = np.array([1, 2, 3])

my\_array2 = np.array([1.5, 20.15, 3.601])

my\_array3 = np.array(["Osama\_Elzero", "B", "Ahmed"])

print(my\_array1.dtype)

print(my\_array2.dtype)

print(my\_array3.dtype)

my\_array4 = np.array([1, 2, 3], dtype=float) # float Or 'float' Or 'f'

my\_array5 = np.array([1.5, 20.15, 3.601], dtype=int) # int Or 'int' Or 'i'

# my\_array6 = np.array(["Osama\_Elzero", "B", "Ahmed"], dtype=int) # Value Error

print(my\_array4.dtype)

print(my\_array5.dtype)

# print(my\_array6.dtype)

my\_array7 = np.array([0, 1, 2, 3, 0, 4])

print(my\_array7.dtype)

my\_array7 = my\_array7.astype('float')

print(my\_array7.dtype)

my\_array7 = my\_array7.astype('bool')

print(my\_array7.dtype)

my\_array8 = np.array([100, 200, 300, 400], dtype='f')

print(my\_array8.dtype)

print(my\_array8[0].itemsize) # 4 Bytes

my\_array8 = my\_array8.astype('float') # Change To Float64

print(my\_array8.dtype)

print(my\_array8[0].itemsize) # 8 Bytes

# 148

import numpy as np

my\_array3 = np.array([[1, 4], [5, 9]])

my\_array4 = np.array([[2, 7], [10, 5]])

print(my\_array3 + my\_array4)

print(my\_array3 - my\_array4)

print(my\_array3 \* my\_array4)

print(my\_array3 / my\_array4)

my\_array5 = np.array([10, 20, 30])

print(my\_array5.min())

print(my\_array5.max())

print(my\_array5.sum())

my\_array6 = np.array([[6, 4], [3, 9]])

print(my\_array6.min())

print(my\_array6.max())

print(my\_array6.sum())

my\_array7 = np.array([[6, 4], [3, 9]])

print(my\_array7.ravel())

my\_array8 = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print(my\_array8.ndim)

print(my\_array8.ravel())

x = my\_array8.ravel()

print(x.ndim)

# 149

my\_array3 = np.array([[[1, 2, 3], [1, 2, 3]], [[1, 2, 3], [1, 2, 3]]])

print(my\_array3.ndim)

print(my\_array3.shape)

my\_array4 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

print(my\_array4.ndim)

print(my\_array4.shape)

reshaped\_array4 = my\_array4.reshape(3, 4)

print(reshaped\_array4.ndim)

print(reshaped\_array4.shape)

print(reshaped\_array4)

my\_array5 = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]])

print(my\_array5.ndim)

print(my\_array5.shape)

reshaped\_array5 = my\_array5.reshape(-1)

reshaped\_array5 = my\_array5.reshape(2, 5, 2)

print(reshaped\_array5.ndim)

print(reshaped\_array5.shape)

print(reshaped\_array5)

# 152

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

# -- Outro and Resources --

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

# Documentations => https://docs.python.org/3/

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

# Useful Websites:

# - Real Python     => https://realpython.com/

# - Programiz       => https://www.programiz.com/python-programming

# - GeeksforGeeks   => https://www.geeksforgeeks.org/python-programming-language/

# - W3Schools       => https://www.w3schools.com/python/default.asp

# - LearnPython     => https://www.learnpython.org/

# - TutorialsPoint  => https://www.tutorialspoint.com/python/index.htm

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

# Collection

# - https://wiki.python.org/moin/BeginnersGuide/Programmers

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

# Resources

# - https://awesome-python.com/

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

# The Next Level ?

# - GUI With Tkinter & PyQt

# - Parsing Html With BeautifulSoup

# - Manage HTTP Requests With Requests Module

# - Web Development With Django & Flask & Web.py

# - The Binary Number System

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

م 2023/07/01 السبت