```python
# 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"]
```

```python
# 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]))
```

```python
# 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
```

```python
# 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")
```

```python
# 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()
```

```python
# 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)
```

```python
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):
```

```python
  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
```

```python
# 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")
```

```python
# 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"
```

```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()}"
```

```python
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)
```

```python
# 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}'")
```

```python
# 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
```