Random – Ngẫu Nhiên:

1. Cách Import?

from random import random

1. Trả Về Số Ngẫu Nhiên Thuộc Phân Phối Đều Liên Tục Từ 0 Tới 1?

random()

Progress Bar – Thanh Tiến Trình:

1. Cách Import?

from tqdm import tqdm

1. Cách Dùng?

* Chỉ dùng cho For Loop

<Bar> = tqdm(

<Iterable>, <Mô Tả>, <Sức Chứa>,

unit = <Đơn Vị>, initial = <Giá Trị Khởi Đầu>

)

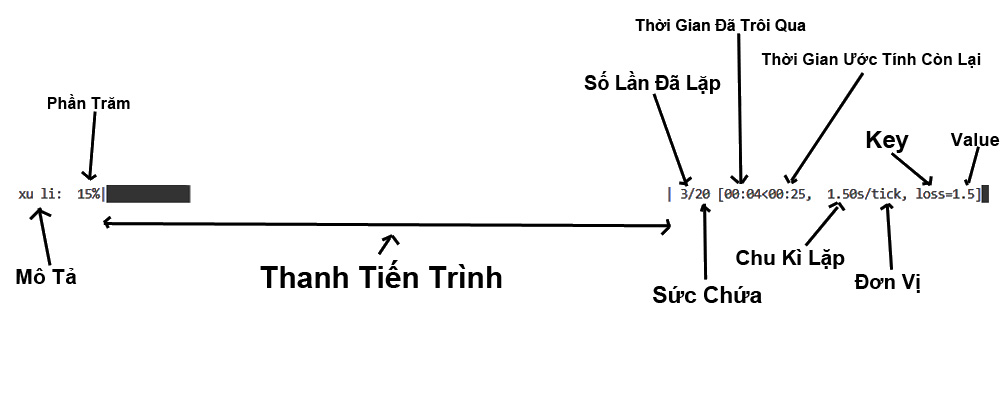
for <Biến Lặp> in <Bar>:

<Làm Gì Đó Nhưng Không Được Phép In Cái Gì Ra Màn Hình>

<Bar>.set\_postfix({<Key> : <Value>})

* <Key> phải là String
* Mặc định

|  |  |  |  |
| --- | --- | --- | --- |
| <Mô Tả> | None | <Sức Chứa> | <Kích Thước Iterable> |
| <Đơn Vị> | "it" | <Giá Trị Khởi Đầu> | 0 |

* Sẽ có 1 thanh tiến trình được in ra như sau
* 
* Thanh này sẽ được cập nhật lại mỗi lần lặp

Time – Thời Gian:

1. Cách Import?

import time

1. Trả Về Số Giây Đã Trôi Qua Tính Từ Thời Điểm Giao Thừa Giữa Năm 1969 Và 1970?

time.time()

1. Cách Đo Thời Gian Thực Thi 1 Đoạn Code Kể Cả Có Gián Đoán Hay Tạm Dừng?

* Chạy câu lệnh sau lần đầu tiên để bắt đầu tính thời gian, lưu ý thời gian bắt đầu sẽ có 1 chút Offset so với 0

time.perf\_counter()

* Tại 1 thời điểm nào đó, nếu bạn chạy câu lệnh này lần nữa thì nó sẽ trả về số giây đã trôi qua kể từ thời điểm 0

1. Trả Về Số Giây Mà CPU Đã Dành Ra Để Xử Lí Chương Trình Cho Đến Thời Điểm Hiện Tại Không Kể Thời Gian Gián Đoạn Hay Tạm Dừng?

time.process\_time()

* Lưu ý chỉ tính thời gian trên CPU chính, không tính trên những CPU khác trong máy

1. Tạm Dừng Chương Trình?

time.sleep(<Số Giây>)

Python System Utilities:

1. Cách Import?

import psutil as ps

1. Trả Về List Chứa ID Của Tất Cả Process Đang Chạy?

ps.pids()

1. Truy Cập 1 Process Với ID?

<Process> = ps.Process(<ID>)

1. Trả Về Tên Của 1 Process?

<Process>.name()

Pillow:

1. Cách Import?

from PIL import Image

1. Cách Tạo 1 PIL Image?

<PIL Image> = Image.open(<Đường Dẫn Tới Ảnh>)

* Để biết cấu hình ảnh thì chuyển nó sang Numpy Array
* Ví dụ

foo = Image.open("./bar/boo.png")

1. Chuyển PIL Image Sang Ảnh Trắng Đen?

<PIL Image Mới> = <PIL Image>.convert("L")

* Tóm tắt các trường hợp

|  |  |
| --- | --- |
| <PIL Image> Shape | <PIL Image Mới> Shape |
| (<Kích Thước Chiều Dọc Ảnh>,  <Kích Thước Chiều Ngang Ảnh>,  <Số Kênh Ảnh>) | (<Kích Thước Chiều Dọc Ảnh>,  <Kích Thước Chiều Ngang Ảnh>) |
| (<Kích Thước Chiều Dọc Ảnh>,  <Kích Thước Chiều Ngang Ảnh>) | (<Kích Thước Chiều Dọc Ảnh>,  <Kích Thước Chiều Ngang Ảnh>) |

Pretty Print – In Đẹp:

1. Cách Import?

from pprint import pprint

1. Cách In 1 Iterable Dài Vãi Đái Thành Từng Dòng Một Thay Vì In Nguyên 1 Dòng Không Dứt?

pprint(<Iterable>)

Operating System – Hệ Điều Hành:

1. Cách Import?

import os

1. Trả Về Tổng Số Thread Trong Máy?

os.cpu\_count()

1. Trả Về Tất Cả Các Biến Môi Trường Hiện Có Trong Phiên Làm Việc?

<Tất Cả Biến Môi Trường> = os.environ

* <Tất Cả Biến Môi Trường> có cấu trúc dữ liệu giống với Dictionary
* Để thay đổi giá trị hoặc tạo mới 1 cặp <Key> <Value>

<Tất Cả Biến Môi Trường>[<Key>] = [<Value>]

* <Key> và <Value> đều có kiểu String
* Ví dụ

os.environ["foo"] = "bar"

* Lưu ý kết thúc phiên làm việc tất cả sẽ trở về như cũ

1. Trả Về Phần Cuối Cùng Của 1 Đường Dẫn?

os.path.basename(<Đường Dẫn>)

* Ví dụ

foo = os.path.basename("bar/fool/boo.py")

* Ta có

|  |  |
| --- | --- |
| foo | "boo.py" |

1. Trả Về Đường Dẫn Tuyệt Đối Của Thư Mục Làm Việc Hiện Tại?

os.getcwd()

1. Trả Về List Tên Các Folder Và File Con Trực Tiếp Trong 1 Thư Mục?

os.listdir(<Đường Dẫn Tới Thư Mục>)

* Lưu ý thứ tự các phần tử sẽ phụ thuộc vào cách hệ thống đánh <Index> File hoặc thư mục của bạn
* Ví dụ

foo = os.listdir("./bar")

* Ta có

|  |  |
| --- | --- |
| foo | ['boo', 'dumano.js', 'foo.txt', 'gg.py', 'js.hs', 'zz'] |

1. Nối Các Đường Dẫn Với Nhau?

os.path.join(<Các Đường Dẫn>)

* Ví dụ

far = os.path.join(foo, bar, boo)

* Ta có

|  |  |
| --- | --- |
| foo | "abc" |
| bar | "huhu/hoho" |
| boo | "vcl/lol.png" |
| far | "abc/huhu/hoho/vcl/lol.png" |

1. Kiểm Tra Xem 1 Đường Dẫn Có Dẫn Tới 1 File Đang Tồn Tại Trên Máy Không?

os.path.isfile(<Đường Dẫn>)

* Ví dụ

foo = os.path.isfile("./bar")

* Ta có

|  |  |
| --- | --- |
| foo | False |

1. Chia Đường Dẫn Thành Tên Và Phần Mở Rộng?

<Tên>, <Phần Mở Rộng> = os.path.splitext(<Đường Dẫn>)

* Ví dụ

foo, bar = os.path.splitext("./boo/far/fool.js")

* Ta có

|  |  |
| --- | --- |
| foo | "./boo/far/fool" |
| bar | ".js" |

1. Di Chuyển File Hoặc Thư Mục Và Đổi Tên?

os.rename(<Đường Dẫn Tới File Hoặc Thư Mục>, <Đường Dẫn Đích>)

* Ví dụ

os.rename("./bar/foo", "./bar/fool/far/boo")

* Thư mục “foo” đã được di chuyển đến bên trong thư mục “far” và được đổi tên thành “boo”

1. Chuyển Thư Mục Làm Việc Hiện Tại Sang Thư Mục Khác?

os.chdir(<Đường Dẫn Tới Thư Mục khác>)

* Lưu ý chỉ có tác dụng đối với chương trình đang chạy
* Ví dụ

os.chdir("foo/bar/far")

* Thư mục làm việc hiện tại là "fool/boo", thư mục làm việc sau đó là "fool/boo/foo/bar/far"

1. Truy Cập Meta Data Của 1 File Hoặc Folder?

<Meta Data> = os.stat(<Đường Dẫn Tới File Hoặc Folder>)

* Dữ liệu trong <Meta Data> có thể là thời gian tạo File hay Folder, thời gian chỉnh sửa cuối cùng, kích thước,…
* <Meta Data> được lưu trong Header của File hay Folder
* Header đứng ở đầu File hay Folder, là cái được đọc đầu tiên
* Truy cập chúng thông qua thuộc tính của <Meta Data>
* Ví dụ

foo = os.stat("foo/bar/fool.py")

* ta có

|  |  |
| --- | --- |
| foo.st\_ctime | 1692603211 |
| foo.st\_atime | 1692602854 |
| foo.st\_size | 22 |
| … | … |

Shell Utilities:

1. Cách Import?

import shutil

1. Copy File Và Đổi Tên?

shutil.copyfile(<Đường Dẫn Tới File>, <Đường Dẫn Đích>)

* Ví dụ

shutil.copyfile("./bar.js", "./boo/far.js")

* File “bar.js” được Copy vào thư mục “boo” và đổi tên thành “far.js”

Requests – Yêu Cầu:

1. Cách Import?

import requests

1. Cách Gửi Request Với Phương Thức Get Tới 1 URL?

<Response> = requests.get(<URL>)

* Ví dụ

req = requests.get("https://chat.openai.com/")

* Truy cập dữ liệu trả về

<Dữ Liệu> = <Response>.content

* <Dữ Liệu> có kiểu dữ liệu là Byte String
* Để nhận được dữ liệu đã được mã hóa sang String

<Dữ Liệu Text> = <Response>.text

Beautiful Soup:

1. Cách Import?

from bs4 import BeautifulSoup as bs

1. Tạo 1 Beautiful Soup Object?

<Beautiful Soup Object> = bs(<Dữ Liệu>, <Trình Phân Tích Dữ Liệu>)

* <Dữ Liệu> có kiểu dữ liệu là Byte String hoặc String
* <Dữ Liệu> sẽ được <Trình Phân Tích Dữ Liệu> xử lí
* Ví dụ

foo = bs("<html><body><p>Hello</p></body></html>", "html.parser")

1. Trả Về String Dễ Đọc Của Dữ Liệu HTML?

<Beautiful Soup Object Của Dữ Liệu>.prettify()

1. Trả Về String Được Loại Bỏ Tag Của Dữ Liệu HTML?

<Beautiful Soup Object Của Dữ Liệu>.get\_text()

1. Trích Xuất 1 Loại Tag Của Dữ Liệu HTML?

<Tag Object> = <Beautiful Soup Object Của Dữ Liệu>.<Loại Tag>

* <Tag Object> cũng hoạt động giống <Beautiful Soup Object Của Dữ Liệu>
* Ví dụ

bar = foo.title

Copy – Sao Chép:

1. Cách Import?

import copy

1. Trả Về Nguyên Copy Của 1 Object Và Không Dùng Chung Dữ Liệu Với Object Đó?

copy.deepcopy(<Object>)

Math – Toán:

1. Cách Import?

import math

1. Trả Về Logarithm Tự Nhiên Của 1 Số?

math.log(<Số Muốn Lấy Logarithm>)

* <Số Muốn Lấy Logarithm> phải > 0

Itertools:

1. Cách Import?

import itertools as itls

1. Trả Về Các Tổ Hợp Của 1 Tập?

<Các Tổ Hợp> = itls.combinations(<Tập Hợp>, <Chập>)

* <Các Tổ Hợp> là 1 Generator, chứa tất cả tổ hợp chập <Chập> của <Tập Hợp>, xếp theo thứ tự như khi bạn liên kê bằng tay
* Ví dụ

foo = itls.combinations([99, 54, 75], 2)

* Ta có

|  |  |
| --- | --- |
| foo | [[99, 54],  [99, 75],  [54, 75]] |

1. Trả Về Tích Đề Các Của Các Tập Hợp?

<Tích Đề Các> = itls.product(<Tập Hợp 1>, <Tập Hợp 2>, …, repeat = <Mũ>)

* Ban đầu, tính tích Đề Các D của các <Tập Hợp>, mỗi phần tử trong D là 1 Tuple, ứng với 1 kết hợp, nếu chỉ có 1 tập thì D = tập đó luôn và mỗi phần tử trong nó là Tuple với chỉ 1 giá trị, thứ tự sắp xếp như khi bạn liệt kê bằng tay, sau đó D sẽ lũy thừa theo <Mũ>, nếu <Mũ> = 1 thì không làm gì, mũ bằng 2 thì lấy D nhân với chính nó, lưu ý Tuple trong D kết hợp với chính nó = phép nối, ví dụ (1, 2) với (2, 3) thành (1, 2, 2, 3), tương mũ 3, 4, …

foo = itls.product([99, 54, 75], ["ab", "bar"], [5, 6])

bar = itls.product([1, 2], repeat = 3)

* Ta có

|  |  |
| --- | --- |
| foo | [[99, "ab", 5], [99, "ab", 6],  [99, "bar", 5], [99, "bar", 6],  [54, "ab", 5], [54, "ab", 6],  [54, "bar", 5], [54, "bar", 6],  [75, "ab", 5], [75, "ab", 6],  [75, "bar", 5], [75, "bar", 6]] |
| bar | [[1, 1, 1], [1, 1, 2],  [1, 2, 1], [1, 2, 2],  [2, 1, 1], [2, 1, 2],  [2, 2, 1], [2, 2, 2]] |

combinations(Array, K):

Usage:

Trả về Itertool Array mà mỗi phần tử của nó là một Tuple tổ hợp chập

K của Array ban đầu

Ex:

list(Itertools.combinations([1, 2, 3], 2)) [(1, 2), (1, 3), (2, 3)]

cmath

phase(Complex Number):

Usage:

Trả về Phase của Complex Number

Ex:

cmath.phase(complex(4, 4)) 0.7853…

polar(Complex Number):

Usage:

Trả về Tuple gồm 2 giá trị là Modulus và Phase của Complex Number

Ex:

cmath.polar(complex(4, 4)) (5.6568…, 0.7853…)

rect(Modulus, Phase):

Usage:

Tạo Complex Number bằng Modulus và Phase

Ex:

cmath.rect(2, 0.1) 1.99 + 0.2j

PIL:

1. Cách Import?

* Import PIL

.Image

.new(Mode, Size Tuple):

Trả về một ảnh mới với chế độ màu Mode, kích thước Size

Ex:

image = PIL.Image.new(‘RGB’, (300, 200))

.open(File Name):

Trả về một Image File Object tương ứng với File Name

Ex:

image = PIL.Image.open(‘foo.png’)

.show():

Hiện ảnh của Image File qua cửa sổ mới

Ex:

image.show()

.save(File Name):

Lưu ảnh với File Name

Ex:

image.save(‘bar.png’)

.thumbnail((X, Y)):

Resize ảnh Inplace với độ phân giải bảo toàn Ratio

Kích thước mới có X mới không vượt quá X cũ và Y tương tự

Không thể phóng to

Ex:

Resize (64, 128) => (16, 32)

image.thumbnail((32, 32))

image.thumbnail((16, 48))

Resize (64, 128) => (64, 128)

image.thumbnail((100, 150))

.rotate(Degree):

Trả về ảnh bị xoay ngược chiều kim đồng hồ góc Degree

Ex:

image = image.rotate(90)

.convert(Mode):

Trả về ảnh với cấu hình màu theo Mode

Ex:

Màu đen trắng

image = image.convert(mode = ‘L’)

.filter(Filter):

Trả về ảnh đã được Filter Object tác dụng

Ex:

Image = image.filter(PIL.ImageFilter.GaussianBlur())

.ImageFilter

.GaussianBlur(Kernel Size):

Trả về Filter Object với Kernel là Matrix vuông phân phối chuẩn

Ex:

Filter = PIL.ImageFilter.GaussianBlur(15)

.ImageFont

.truetype(Font File, Font Size):

Trả về Font Object

Ex:

font = PIL.ImageFont.truetype(‘arial.ttf’, 50)

.ImageDraw

.Draw(Image):

Trả về Draw Object trên Image

Ex:

draw = PIL.ImageDraw.Draw(PIL.Image.open(‘foo.png’))

.text((X, Y), Text, Fill, Font):

Vẽ Text lên ảnh với tọa độ (X, Y), màu Fill

Ex:

draw.text((50, 40), ‘FooBar’, (255, 125, 126),

PIL.ImageFont.truetype(‘arial.ttf’, 50))

matplotlib

rules:

Nếu không tạo Figure mà vẫn Plot thì sẽ có một Figure mới tự động tạo

Plot trả về dùng .remove() để xóa

.pyplot

.figure():

Trả về một Figure (một cửa sổ)

.savefig(File Name, bbox\_inches = ‘tight’):

Save Figure hiện tại với định dạng ảnh

Ex:

matplotlib.pyplot.savefig(‘foo.png’, bbox\_inches = ‘tight’)

.pause(Time):

Pause khoảng Time giây, nếu không Pause và không có Show thì các

Figure sẽ chạy liên hoàn và đóng

Ex:

matplotlib.pyplot.pause(0.1)

.show():

Hiện tất cả các Figure đã khai báo, tạm ngưng chương trình cho đến

khi tắt tất cả Figure đóng

plot(X\_List, Y\_List):

Usage:

Trả về một Array gồm các đoạn đường Plot

Plot đường

Ex:

new\_plot = matplotlib.pyplot.plot([1, 2, 3], [4, 5, 6])

new\_plot.pop(0).remove()

.grid():

Tạo các đường kẻ theo các mốc trên trục

axes(Projection Type):

Usage:

Trả về Axes

Set projection = “3d” để Plot 3d

Ex:

axes = matplotlib.pyplot.axes(projection = “3d”)

axes.scatter(3, 5, 7)

scatter(X, Y, Alpha):

Usage:

Plot điểm

Ex:

matplotlib.pyplot.scatter(4, 5, alpha = 0.5)

.gca():

Trả về Current Axis Object

Ex:

axis = matplotlib.pyplot.gca()

.add\_patch(Patch):

Vẽ hình lên Plot

Ex:

rect = matplotlib.patches.Rectangle((0, 0), 1, 2)

axis.add\_patch(rect)

.add\_collection(Collection):

Vẽ hình có thứ tự Layer trước sau

Ex:

rect = matplotlib.patches.PatchCollection(

[matplotlib.patches.Rectangle((0, 0), 1, 2)], zorder = 1

)

axis.add\_collection(rect)

.xticks(Array):

Đặt mốc cho trục là các phần tử trong Array

Ex:

matplotlib.pyplot.xticks([1, 2, 3])

.xlim(Left, Right):

Đặt Boundary cho trục

Ex:

matplotlib.pyplot.xlim(-10, 10)

.quiver(X, Y, U, V , angles = ‘xy’, scale\_units = ‘xy’, scale = 1):

Plot Vector có tọa độ gốc (X, Y), hướng (U, V)

X, Y, U, V có thể là mảng nhiều chiều

Ex:

matplotlib.pyplot.quiver([1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12])

.patches

.Rectangle((X, Y), Width, Height):

Trả về một hình chữ nhật có tọa độ Bottom Left Corner là (X, Y), chiều

rộng Width, chiều cao Height

Ex:

rect = matplotlib.patches.Rectangle((1, 2), 5, 6)

.Polygon(XY):

Trả về một đa giác có tọa độ là các Row của XY

Ex:

poly = matplotlib.patches.Polygon([[1, 2], [3, 4], [5, 6]])

.PatchCollection(Patch Array, zorder):

Trả về một Collection gồm các Patch trong Patch Array với thứ tự ưu

tiên được vẽ lên trên zorder

Ex:

coll = matplotlib.patches.PatchCollection(

[

matplotlib.patches.Rectangle((0, 0), 1, 2),

matplotlib.patches.Rectangle((1, 2), 3, 4)

], zorder = 1, color = [0.5, 1, 1]

)

os

.chdir(Path): Đổi Current Working Directory sang Path

Ex: os.chdir(‘C:/Users/Huan/bar’)

.mkdir(Path): Tạo Folder Path, tối đa tạo ra 1 Folder, Error nếu Path đã tồn tại

Ex: os.mkdir(‘C:/Users/Huan/dummy’)

os.mkdir(‘C:/Users/Huan/dummy’) => Error

os.mkdir(‘C:/Users/Huan/dummy/foo/bar’) => Error

.makedirs(Path): Tạo Folder Path, có thể tạo nhiều Folder chồng lên nhau, Error nếu

Path đã tồn tại

Ex: os.mkdir(‘C:/Users/Huan/dummy/foo/bar’)

os.mkdir(‘C:/Users/Huan/dummy’) => Error

.rmdir(Path): Xóa Folder cuối trong Path, Error nếu Path không rỗng

Ex: os.rmdir(‘C:/Users/Huan/dummy/foo’) => C:/Users/Huan/dummy

os.rmdir(‘C:/Users’) => Error

.removedirs(Path): Xóa đệ quy từng Folder trong Path ngược trở lại, dừng xóa nếu

Folder không rỗng, Error nếu Path không rỗng

Ex: os.removedirs(‘C:/Users/Huan/dummy/foo’) => C:/Users/Huan

os.removedirs(‘C:/Users’) => Error

.rename(Src, Des): Cut Src đến Des và đổi tên thành Des

Ex: os.rename(‘C:/Users/Huan/dummy.py’, ‘C:/Users/foo.py’)

.stat(Path): Trả về Object thông tin của File Path

Ex: os.stat(‘C:/Users/Huan/foo.py’) = {‘st\_mtime’ = 12345, st\_size = 20, …}

.walk(Path): Trả về Generator Object, Search vét cạn trong Path, trả về 3 giá trị là

Directory hiện tại đang search, Array tên tất cả Folder con trực tiếp,

Array tên tất cả File con trực tiếp, Path = ‘.’ = Current Working Directory

Ex: for r, d, f in os.walk(‘C:/Users/Huan’)

Scipy

Stats

Norm

Pdf(num | arr, mean, std): trả về normal likelihood tại ví trí num

Ex: scipy.stats.norm.pdf(2) ~= 0.054

ctypes

.windll

.user32

.MessageBoxW(0, Content, Title, 1):

Hiện lên một Message Box

.GetSystemMetrics(Num):

Trả về độ phân giải màn hình theo chiều ngang nếu Num = 0,

chiều cao nếu Num = 1

Ex: ctypes.windll.user32.GetSystemMetrics(0) = 1280

Time

Perf\_counter(): trả về giá trị float thời gian hiện tại

Ex: start = time.perf\_counter(), time.sleep(1), end = time.perf\_counter()

end – start ~= 1

Queue

Queue(): trả về một queue

Ex: q = queue.Queue()

Put(val): Thêm val vào queue

Ex: q.put(1)

Get(): trả về giá trị đầu tiên của queue vào xóa nó khỏi queue, nếu queue

rỗng sẽ đợi cho đến một luồng nào đó put val vào queue

Ex: q.get() = 1

q.get() = loading

Task\_done(): sau khi get vào xử lý giá trị get, gọi hàm này để thông báo cho

queue biết, kết hợp với join để kết thúc queue

Ex: a = q.get()

Processing a…

q.task\_done()

Join(): chờ cho đến khi tất cả task đều done (số task done = size queue)

Ex: q.join()

Empty(): check nếu queue rỗng trả true, ngược lại false

Ex: q.empty() = True

Multiprocessing: Chạy task trên nhiều cpu, không share memory

Process(target = callback, args = [] – list tham số cho callback): trả về một process

Ex: p = multiprocessing.process(target = foo)

Start(): bắt đầu chạy task parallel

Ex: p.start()

Join(): chờ task chạy xong rồi mới chạy tiếp

Ex: p.join()

Cpu\_count(): trả về số lượng cpu

Lock(): trả về lock object, dùng nó như một context wrapper, code bên trong chỉ chạy

khi không còn task nào đang chạy với cùng lock này(switch khi time sleep)

Ex: lock = mp.Lock()

def foo():

with lock:

Processing...

Value(‘i’ - integer| ‘d’ - double, val): tạo biến giá trị có thể share giữa các cpu

Ex: shared\_value = multiprocessing.Value(‘i’, 1)

Value: trả về giá trị hiện tại

Ex: shared\_value.value = 1

Array(): như value nhưng là mảng:

Ex: shared\_array = multiprocessing.Value(‘d’, [1.2, 1.3, 1.5])

shared\_array[:] = [1.2, 1.3, 1.5]

Queue(): như value nhưng là queue:

Ex: shared\_queue = multiprocessing.Queue()

shared\_queue.put(1)

Pool(): trả về một pool, hàm tự động chia nhỏ dữ liệu để xử lí trên nhiều cpu, đặt

trong \_\_main\_\_ để không bị lỗi

Ex: pool = multiprocessing.Pool()

Map(func, arr\_args): chia array tham số thành nhiều batch chạy trên nhiều

Cpu, sau khi hoàn thành xong hết trả về array result

Ex: res = pool.map(foo, [1, 2, 3])

Loading …

Res = [1, 4, 9]

Close(): stop pool running

Ex: pool.close()

Join(): chờ cho đến khi pool hoàn thành xong

Ex: pool.join()

Threading: Chạy multi thread trên 1 cpu, share memory

Thread(target = callback, args = [] – list tham số cho callback): trả về một thread

Ex: p = threading.Thread(target = foo)

Name: trả về tên của thread

Ex: p.name = Thread-56

Daemon: giá trị true nếu muốn kill thread khi main thread die(tránh chạy

ngầm khi chương trình chính đã chạy xong)

Ex: p.daemon = True

Current\_thread(): trả về thread hiện tại đang chạy

Ex: threading.current\_thread().name = Thread-56

Functools

Lru\_cache(None – infinite cache | maxsize = num): Least Recently Used Cache

Cache một function, nếu gọi function đó với args cũ sẽ không chạy code bên

trong mà chỉ trả về giá trị trước đó đã trả về, số lượng cache = maxsize

Ex: @functools.lru\_cache(maxsize = 4)

def foo(a): print(a) return a

foo(1) foo(1) foo(1) => 1

Collections

Defaultdict(callback): Giống dict nhưng có giá trị default được trả về từ callback cho

key không tồn tại

Ex: a = collections.defaultdict(list), print(a[1]) => []

Pygad: Thư viện genetic algorithm

Torchga

TorchGA(model = torch\_model, num\_solutions = số cá thể): trả về torchga

Ex: torchga = pygad.torchga.TorchGA(model = model, num\_solutions

= 10)

Population\_weights: trả về 2d list các chuỗi parameters của các cá thể

Ex: torchga.population\_weights = 10 \* 30 array

Model\_weights\_as\_dict(model = torch\_model, weights\_vector = 1 list):

Convert chromosome sang state dict để model load state dict

Ex: pygad.torchga.model\_weights\_as\_dict(model = model,

weights\_vector = [1, 2, 3, 4, 5, 6]) = order dict

GA(num\_generations = số đời, num\_parents\_mating = số cha mẹ,

initial\_population = đời đầu, fitness\_func = fitness callback (solution => fitness),

on\_generation = callback sau mỗi đờ, parent\_selection\_type = ‘sss’ – steady

state selection, crossover\_type = ‘single\_point’, mutation\_type = "random",

mutation\_percent\_genes = num, keep\_parents = num – số parent giữ lại | -1 –

giữ tất cả parent): trả về một PA

Run(): chạy các đời

Ex: ga\_instance.run()

Plot\_result(title = hiệu): plot kết quả fitness

Ex: ga\_instance.plot\_result(title = ‘title’)

Best\_solution(): trả về solution, fitness, index của con mạnh nhất

Ex: ga.best\_solution() = [1, 2, 3], 10, 1