Hướng dẫn lập trình

NGÔN NGỮ PYTHON

(Cho học phần Toán cao cấp - Phiên bản 12/2021)

LỜI NÓI ĐẦU

"Ngôn ngữ lập trình phổ biến nhất trên thế giới"

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido Van Rossum. Thiết kế bắt đầu vào cuối những năm 1980 và được phát hành lần đầu tiên vào tháng 2 năm 1991.

Tính năng chính của Python: Ngôn ngữ lập trình đơn giản, dễ học; Miễn phí, mã nguồn mở; Khả năng di chuyển: chạy được trên Windows, Mac OS, Linux. Khả năng mở rộng và có thể nhúng: kết hợp các ngôn ngữ khác vào code Python; Ngôn ngữ thông dịch cấp cao: không phải quản lý bộ nhớ, dọn dẹp những dữ liệu vô nghĩa,…; Thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến; Hướng đối tượng.

Python được dùng trong: Lập trình ứng dụng web; Khoa học và tính toán (đặc biệt trong Machine Learning, Data mining và Deep Learning); Tạo nguyên mẫu phần mềm (bản chạy thử – prototype); Ngôn ngữ tốt để dạy lập trình.

Tài liệu ngắn này chúng tôi viết dựa theo một số tài liệu trên internet nhằm giúp các bạn sinh viên làm quen với việc sử dụng một ngôn ngữ lập trình để giải các bài toán của học phần Toán cao cấp.

LÊ VĂN TUẤN[1]

[1] Bộ môn Toán – Trường Đại học Thương mại

MỤC LỤC

Chủ đề 0. Hello World

Ví dụ này sẽ thực hiện: in chữ Hello World! và in giá trị của sin0

```
import math

print("Hello World !")
print(math.sin(0))
```

- Câu lệnh import math để nhập thư viện math
- Lệnh print để in chữ/số ra cửa sổ kết quả

Chủ đề 1. Tính toán trên trường số thực

1. Các phép toán trên trường số thực là:

cộng (+), trừ (-), nhân (*), chia (/), lũy thừa (**)

2. Các hàm thông dụng trong thư viện math:

```
import math
```

Hàm toán học	Trên Python	**Hàm toán học**	Trên Python
$\boldsymbol{\pi}$	math.pi	**sin(x)**	math.sin(x)
e	math.e	**cos(x)**	math.cos(x)
$+\infty$; $-\infty$	math.inf ; -math.inf	**tan(x)**	math.tan(x)
$\sqrt[2]{x}$	math.sqrt(x)	**cot(x)**	1/math.tan(x)
|x|	abs(x)	**arcsin(x)**	math.asin(x)
ln(x) ; lg(x) ; $\log_a x$	math.log(x) ; math.log10(x) ; math.log(x,a)	**arccos(a)**	math.acos(a)
e^x ; x^y	math.exp(x) ; math.pow(x, y)	**arctan(x)**	math.atan(x)
n!	math.factorial(n)	**arccot(x)**	math.atan(1/x)

2. Các hàm thông dụng trong thư viện sympy:

```
import sympy
```

Hàm toán học	Trên Python	**Hàm toán học**	Trên Python
		sin(x)	sympy.sin(x)
		cos(x)	sympy.cos(x)
		tan(x)	sympy.tan(x)
$\sqrt[2]{x}$	sympy.sqrt(x)	**cot(x)**	sympy.cot(x)
|x|	sympy.Abs(x)	**arcsin(x)**	sympy.asin(x)
ln(x) ; $\log_a x$	sympy.log(x) ; sympy.log(x,a)	**arccos(a)**	sympy.acos(a)
e^x	sympy.exp(x)	**arctan(x)**	sympy.atan(x)
		arccot(x)	sympy.acot(x)

3. Các hàm thông dụng trong thư viện numpy:

```
import numpy as np
```

Hàm toán học	Trên Python	**Hàm toán học**	Trên Python

π	np.pi	**sin(x)**	np.sin(x)
e	np.e	**cos(x)**	np.cos(x)
$+\infty$; $-\infty$	np.Inf; np.NINF	**tan(x)**	np.tan(x)
$\sqrt[2]{x}$	np.sqrt(x)	**cot(x)**	1/np.tan(x)
|x|	np.absolute (x)	**arcsin(x)**	np.arcsin(x)
ln(x) ; $\log_a x$	np.log(x) ; np.log(x)/ np.log(a)	**arccos(a)**	np.arccos(a)
e^x	np.exp(x)	**arctan(x)**	np.arctan(x)
		arccot(x)	np.arctan(1/x)

VD1. Tính $\sqrt[2]{5,1}$

Cách 1: Dùng math	Cách 2: Dùng sympy	Cách 3: Dùng numpy

`print(math.sqrt(5.1))`	`x = sympy.Symbol('x')`
	`y = sympy.lambdify(x,sympy.sqrt(x))`
	`print(y(5.1))`
		`print(np.sqrt(5.1))`

VD2. Tính $\log_3(4)$

Cách 1: Dùng math	Cách 2: Dùng sympy	Cách 3: Dùng numpy

`print(math.log(4,3))`	`x = sympy.Symbol('x')`
	`y =sympy.lambdify(x,sympy.log(x,3))`
	`print(y(4))`
		`print(np.log(4)/np.log(3))`

VD3. Tính $\arcsin^2(1/2)$

Cách 1: Dùng math	Cách 2: Dùng sympy	Cách 3: Dùng numpy

`print((math.asin(1/2))**2)`	`x = sympy.Symbol('x')`
	`y = sympy.lambdify(x,(sympy.asin(x))**2)`
	`print(y(1/2))`
		`print((np.arcsin(1/2))**2)`

Chủ đề 2. Ma trận và định thức

- Nhập thư viện:

```
import numpy as np
```

1. Khai báo biến vectơ và ma trận

VD: Khai báo vectơ:

Khai báo vectơ dòng:	Khai báo vectơ cột:
A = np.array([1, 2, 3])	A = np.array([[1], [2], [3]])
print(A) *###(xem vectơ A)*	print(A) *###(xem vectơ A)*

VD: Khai báo ma trận cỡ 3x3:

B = np.array([[11,4,20], [4,9,8], [3,6,9]])

print(B)

2. Các phép toán trên các phần tử của ma trận

Python đang nhớ các biến A và B được khai báo ở trên, ta khai báo thêm biến C:

C = np.array([[0,4,17], [-2,5,8], [3.5,8,-9.2]])

Ta có thể thực hiện các phép toán cộng (+), trừ (-), nhân (*)

VD: D=B+C

VD: D=B*C *###(lưu ý các phép toán này là cho từng phần tử)*

3. Phép nhân hai ma trận

VD: print(np.dot(B,C))

4. Phép lũy thừa ma trận

VD: print(np.linalg.matrix_power(B, 3)) *(tính B^3)*

5. Ma trận chuyển vị

VD: print(B.T)

6. Tìm hạng của ma trận

VD: print(np.linalg.matrix_rank(B))

7. Tìm ma trận nghịch đảo

VD: print(np.linalg.inv (B))

8. Tính định thức (của ma trận vuông)

VD: print(np.linalg.det (B))

Chủ đề 3. Hệ phương trình tuyến tính

- Nhập thư viện:

```
import numpy as np
```

VD1: Giải hệ PTTT: $\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_1 - x_2 \quad\quad = -1; \\ x_1 + x_2 + 2x_3 = 9 \end{cases}$

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`a = np.array([[1,1,1],[1,-1,0],[1,1,2]])`
`b = np.array([6,-1,9])`
`x = np.linalg.solve(a, b)`
`print(x)`
	[1. 2. 3.]

Ghi chú: Trong ví dụ này hệ có nghiệm duy nhất: X=(1, 2, 3)

VD2: Giải hệ PTTT: $\begin{cases} x_1 + x_2 = 6 \\ x_1 + x_2 = 9; \end{cases}$

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`a = np.array([[1,1],[1,1]])`
`b = np.array([6,9])`
`x = np.linalg.solve(a, b)`
`print(x)`
	Traceback (most recent call last):
	……
	numpy.linalg.LinAlgError: Singular matrix

Ghi chú: Trong ví dụ này hệ vô số nghiệm hoặc vô nghiệm.

Chủ đề 4. Dạng toàn phương

- Nhập thư viện:

```
import numpy as np
import scipy.linalg as la
```

VD1: Tìm các giá trị riêng và vectơ riêng của DTP:

$$F(x_1; x_2; x_3) = 4x_1^2 + 4x_2^2 - 8x_3^2 - 10x_1x_2 + 4x_2x_3 + 4x_1x_3$$

Giá trị riêng

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`A = np.array([[4,-5,2],[-5,4,2],[2,2,-8]])`
`results = la.eig(A)`
`print(results[0])`
	[9.00000000e+00+0.j -9.86623977e-17+0.j -9.00000000e+00+0.j]

Ghi chú:Vì python chỉ cho các giá trị xấp xỉ (e+00=10^0=1; e-17 = 10^{-17}) nên DTP thực chất có các giá trị riêng là: 9; 0; -9

Vectơ riêng

Ta thực hiện như sau:	Màn hình hiện kết quả là:
`print(results[1])`	[[-7.07106781e-01 6.66666667e-01 -2.35702260e-01] [7.07106781e-01 6.66666667e-01 -2.35702260e-01] [-1.47451495e-17 3.33333333e-01 9.42809042e-01]]

VD2: Xét tính xác định dấu của DTP:

$$F(x_1; x_2; x_3) = -2x_1^2 - 4x_2^2 - 9x_3^2 + 2x_1x_2 - 8x_2x_3 + 2x_1x_3$$

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`A = np.array([[-2,1,1],[1,-4,-4],[1,-4,-9]])`
`results = la.eig(A)`
`print(results[0])`
	[-11.41363115+0.j -2.1791917 +0.j -1.40717715+0.j]

Ghi chú: Các giá trị riêng là (-11.41363115, -2.1791917, -1.40717715) đều âm nên DTP là xác định âm.

Chủ đề 5. Đồ thị của hàm số

- *Ghi chú: Tham khảo Chủ đề 1 khi cần vẽ các hàm phức tạp.*
- Nhập thư viện:

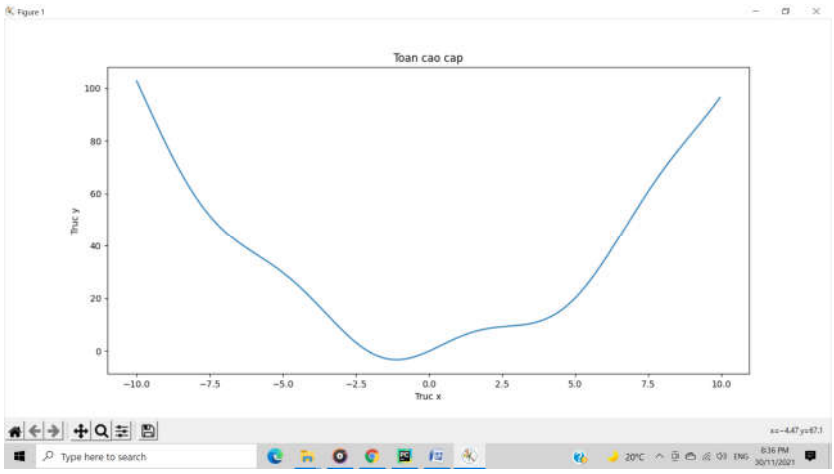
```
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

1. Hàm 1 biến (2D)

VD: Vẽ đồ thị hàm số: $y = x^2 + 5\sin(x)$ trên đoạn [-10; 10]

Thực hiện như sau:

```
x = np.arange(-10, 10, 0.05)
y = x**2 + 5*np.sin(x)
plt.plot(x,y)
plt.xlabel("Truc x")
plt.ylabel("Truc y")
plt.title('Toan cao cap')
plt.show()
```



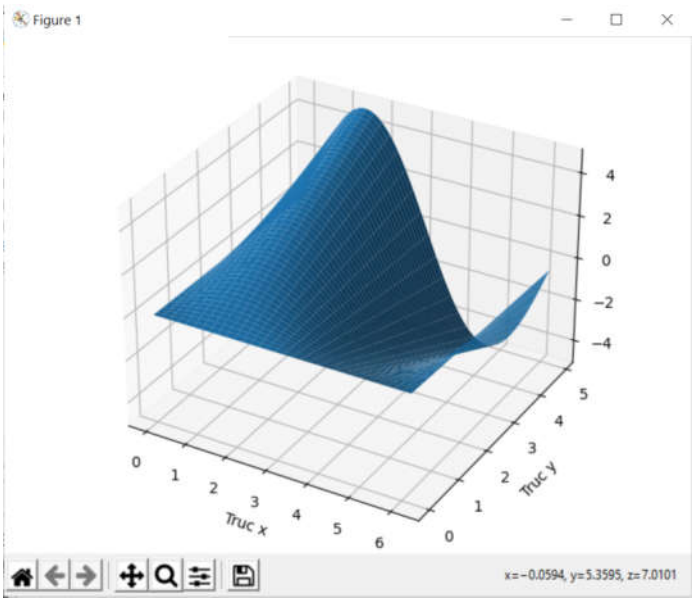
2. Hàm 2 biến (3D)

VD: Vẽ đồ thị hàm z=sin(x)*y với $x \in [0; 2\pi]$, $y \in [0; 5]$

Thực hiện như sau:

```
x = np.arange(0, 2*np.pi, 0.1)
y = np.arange(0, 5, 0.1)
```

```
X, Y = np.meshgrid(x, y)
z = np.sin(X)*Y
fig = plt.figure()
ax = Axes3D(fig)
ax.plot surface(X, Y, z)
plt.xlabel("Truc x")
plt.ylabel("Truc y")
plt.title('Toan cao cap')
plt.show()
```



Chủ đề 6. Giới hạn

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính các hàm phức tạp.*
- Nhập thư viện:

```
import math
import sympy
```

VD1: Tính giới hạn: $\lim_{x\to 0}\frac{\sin x}{x}$
 	
Thực hiện như sau:
 	
`x = sympy.symbols('x')`
`y = sympy.sin(x) / x;`
`result = sympy.limit(y, x, 0)`
`print("Gia tri : {}".format(result))`
	VD2: Tính giới hạn: $\lim_{x\to\infty} x^2\left(1-cos\frac{1}{x}\right)$
	
	Thực hiện như sau:
	
	`x = sympy.symbols('x')`
	`y = (x**2)*(1-sympy.cos(1/x));`
	`result = sympy.limit(y, x, math.inf)`
	`print("Gia tri : {}".format(result))`

Màn hình hiện kết quả là: Gia tri : 1	Màn hình hiện kết quả là: Gia tri : 1/2

VD3: Tính giới hạn: $\lim_{x\to+\infty}\sin x$
 	
Thực hiện như sau:
 	
`x = sympy.symbols('x')`
`y = sympy.sin(x);`
`result = sympy.limit(y, x, +math.inf)`
`print("Gia tri : {}".format(result))`
	VD4: Tính giới hạn: $\lim_{x\to-\infty} x^3$
	
	Thực hiện như sau:
	
	`x = sympy.symbols('x')`
	`y = x**3;`
	`result = sympy.limit(y, x, -math.inf)`
	`print("Gia tri : {}".format(result))`

Màn hình hiện kết quả là: Gia tri : AccumBounds(-1, 1) (Nghĩa là: Không tồn tại giới hạn)	Màn hình hiện kết quả là: Gia tri : -oo (Nghĩa là: Âm vô cùng)

Chủ đề 7. Đạo hàm

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính đạo hàm của các hàm phức tạp.*
- Nhập thư viện:

```
import sympy
```

1. Hàm một biến

VD 1: Tính đạo hàm cấp 1 và cấp 2 của hàm số $f(x) = x^3 + \sin(x)$

Đạo hàm cấp 1	Đạo hàm cấp 2

`x = sympy.Symbol('x')`
`y = x**3 + sympy.sin(x)`
`y1 = y.diff(x)`
`print(y1)`
	`x = sympy.Symbol('x')`
	`y = x**3 + sympy.sin(x)`
	`y1 = y.diff(x)`
	`y2 = y1.diff(x)`
	`print(y2)`

`Kết quả: 3*x**2 + cos(x)`	`Kết quả: 6*x - sin(x)`

VD 2: Tính đạo hàm của hàm số $f(x) = x^3 + \sin(x)$ tại $x = 2$.

```
x = sympy.Symbol('x')
y = x**3 + sympy.sin(x)
y1 = y.diff(x)
y1 = sympy.lambdify(x, y1)
print(y1(2))
```

2. Hàm nhiều biến

VD1: Tính các đạo hàm riêng cấp 1 và cấp 2 của hàm số $z = x^3 + \sin(xy)$

Đạo hàm cấp 1:

z'_x	z'_y

`x, y = sympy.symbols('x y',real=True)`
`z = x**3 + sympy.sin(x*y)`
`z_x = sympy.diff(z, x)`
`print(z x)`
	`x, y = sympy.symbols('x y',real=True)`
	`z = x**3 + sympy.sin(x*y)`
	`z_y = sympy.diff(z, y)`
	`print(z y)`

Đạo hàm cấp 2:

z''_{xx}	z''_{xy}	z''_{yy}
`x, y = sympy.symbols('x y', real=True)` `z = x**3 + sympy.sin(x*y)`	`x, y = sympy.symbols('x y', real=True)` `z = x**3 + sympy.sin(x*y)`	`x, y = sympy.symbols('x y', real=True)` `z = x**3 + sympy.sin(x*y)`

`z_x = sympy.diff(z, x)`
`z_xx = sympy.diff(z_x,x)`
`print(z_xx)`
	`z_x = sympy.diff(z, x)`
	`z_xy = sympy.diff(z_x,y)`
	`print(z_xy)`
		`z_y = sympy.diff(z, y)`
		`z_yy = sympy.diff(z_y,y)`
		`print(z_yy)`

VD2: Tính các đạo hàm riêng cấp 1 và cấp 2 của hàm số $z = x^3 + \sin(xy)$ tại (1,2)

Đạo hàm cấp 1:

z'_x	z'_y

`x, y = sympy.symbols('x y',real=True)`
`z = x**3 + sympy.sin(x*y)`
`z_x = sympy.diff(z, x)`
`z_x = sympy.lambdify((x, y), z_x)`
`print(z_x(1,2))`
	`x, y = sympy.symbols('x y',real=True)`
	`z = x**3 + sympy.sin(x*y)`
	`z_y = sympy.diff(z, y)`
	`z_y = sympy.lambdify((x, y), z_y)`
	`print(z_y(1,2))`

Đạo hàm cấp 2:

z''_{xx}	$z''_{xy} = z''_{yx}$	z''_{yy}

`x, y = sympy.symbols('x y',`
`real=True)`
`z = x**3 + sympy.sin(x*y)`
`z_x = sympy.diff(z, x)`
`z_xx = sympy.diff(z_x,x)`
`z_xx = sympy.lambdify((x,y), z_xx)`
`print(z_xx(1,2))`
	`x, y = sympy.symbols('x y',`
	`real=True)`
	`z = x**3 + sympy.sin(x*y)`
	`z_x = sympy.diff(z, x)`
	`z_xy = sympy.diff(z_x,y)`
	`z_xy = sympy.lambdify((x, y), z_xy)`
	`print(z_xy(1,2))`
		`x, y = sympy.symbols('x y',`
		`real=True)`
		`z = x**3 + sympy.sin(x*y)`
		`z_y = sympy.diff(z, y)`
		`z_yy = sympy.diff(z_y,y)`
		`z_yy = sympy.lambdify((x, y), z_yy)`
		`print(z_yy(1,2))`

Chủ đề 8. Giá trị nhỏ nhất – lớn nhất

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính cho các hàm phức tạp.*
- Nhập thư viện:

```
import math

from scipy.optimize import fmin
```

1. Hàm một biến

VD: Tìm giá trị nhỏ nhất, lớn nhất của hàm $y = \sin^2(x) + x^2 + 1$

Nhỏ nhất (min)	Lớn nhất (max)

`def f(x): return (math.sin(x))**2+x**2+1`
`min_loc = fmin(f, 1.0)`
`print(min_loc)`
`print(f(*min_loc))`
	`def f(x): return -((math.sin(x))**2+x**2+1)`
	`min_loc = fmin(f, 1.0)`
	`print(min_loc)`
	`print(-f(*min_loc))`

Kết quả: Optimization terminated successfully. Current function value: 1.000000 Iterations: 17 Function evaluations: 34 [-8.8817842e-16] `1.0`	Kết quả: Warning: Maximum number of function evaluations has been exceeded. [6.338253e+28] `4.0173451106474974e+57`
Ghi chú. Hàm số đạt min tại x = -8.8817842e-16 ≈ 0 Giá trị min: 1.0	*Ghi chú.* Giá trị max vượt quá giá trị lớn nhất của Python (hàm số tiến ra vô cùng)

2. Hàm nhiều biến

VD: Tìm giá trị nhỏ nhất, lớn nhất của hàm $f = x^2 + (y-1)^2$

Nhỏ nhất (min)	Lớn nhất (max)
`def f(x,y): return x**2 + (y-1)**2` `min loc = fmin(lambda vec: f(vec[0],`	`def f(x,y): return -(x**2 + (y-1)**2)` `min loc = fmin(lambda vec: f(vec[0],`

vec[1]), [0.1, 0.1]) print(min loc) print(f(*min_loc))	vec[1]), [0.1, 0.1]) print(min loc) print(-f(*min_loc))
Kết quả: Optimization terminated successfully. Current function value: 0.000000 Iterations: 51 Function evaluations: 94 [-1.03073585e-05 9.99952728e-01] 2.340897791033299e-09	Kết quả: Warning: Maximum number of function evaluations has been exceeded. [1.02325529e+43 -1.21372471e+43] 2.520179052869857e+86
Ghi chú. Hàm số đạt min tại (x, y) = (-1.03073585e-05, 9.99952728e-01) ≈ (0, 1) Giá trị min: 2.340897791033299e-09 ≈ 0	*Ghi chú.* Giá trị max vượt quá giá trị lớn nhất của Python (hàm số tiến ra vô cùng)

Chủ đề 9. Tích phân hàm một biến

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính cho các hàm phức tạp.*
- Nhập thư viện:

```
import math
from scipy import integrate
```

1. Tích phân thông thường

VD: Tính $\int_1^{10} xe^x dx$

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`f = lambda x: x*math.exp(x)`
`result = integrate.quad(f, 1, 10)`
`print(result)`
	(198238.1921532605, 2.2008860528861583e-09)
	
	(Giá trị tích phân là: 198238.1921532605)

2. Tích phân suy rộng

VD1: Xét tích hội tụ, phân kỳ của $\int_1^{+\infty} \frac{1}{x^2} dx$

Ta thực hiện như sau:	Màn hình hiện kết quả là:

`f = lambda x: 1/x**2`
`result = integrate.quad(f, 1, math.inf)`
`print(result)`
	(1.0, 1.1102230246251565e-14)
	
	(Giá trị tích phân là 1; hội tụ)

Chủ đề 10. Phương trình vi phân

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính cho các hàm phức tạp.*
- Nhập thư viện:

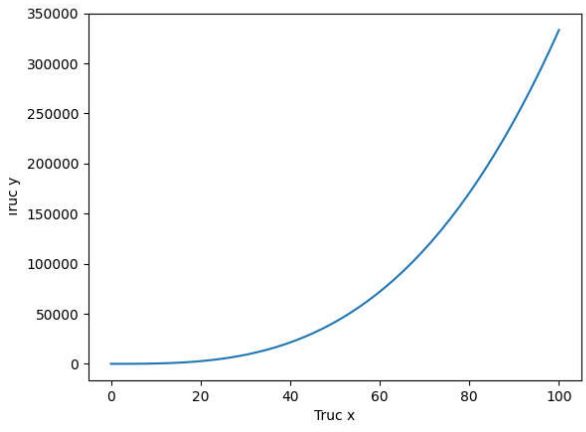
```
import math
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
```

1. Phương trình vi phân cấp 1

VD1: Vẽ đồ thị nghiệm riêng của phương trình vi phân $dy/dx = x^2$ với điều kiện ban đầu $y(0) = 5$ trên miền [0, 100].

Thực hiện như sau:

```
def model(y,x):
    dydx = x**2
    return dydx
y0 = 5
x = np.linspace(0,100)
y = odeint(model,y0,x)
plt.plot(x,y)
plt.xlabel('Truc x')
plt.ylabel('Truc y')
plt.show()
```



Lưu ý. Các dòng dydx = x**2 và return dydx phải lùi vào 1 tab.

VD2: Vẽ đồ thị nghiệm riêng của phương trình vi phân

$$y' + \frac{x}{1-x^2}y = x\sqrt{y}$$

với điều kiện ban đầu y(4) = 2 trên miền [4, 10].

Ta thực hiện như sau:

```
def model(y,x):
    dydx = -x/(1-x**2)*y+x*math.sqrt(y)
    return dydx
y0 = 2
x = np.linspace(4,10)
y = odeint(model,y0,x)
plt.plot(x,y)
plt.xlabel('Truc x')
plt.ylabel('Truc y')
plt.show()
```

2. Phương trình vi phân cấp 2

VD1: Vẽ đồ thị nghiệm riêng của phương trình vi phân: y" – (1 – y^2)y' + y = 0, với điều kiện ban đầu y(5) = 2, y'(5) = 0 trên miền [5, 100].

Đặt U[0] = y, y[1] = y', ta đưa PTVP về hệ:

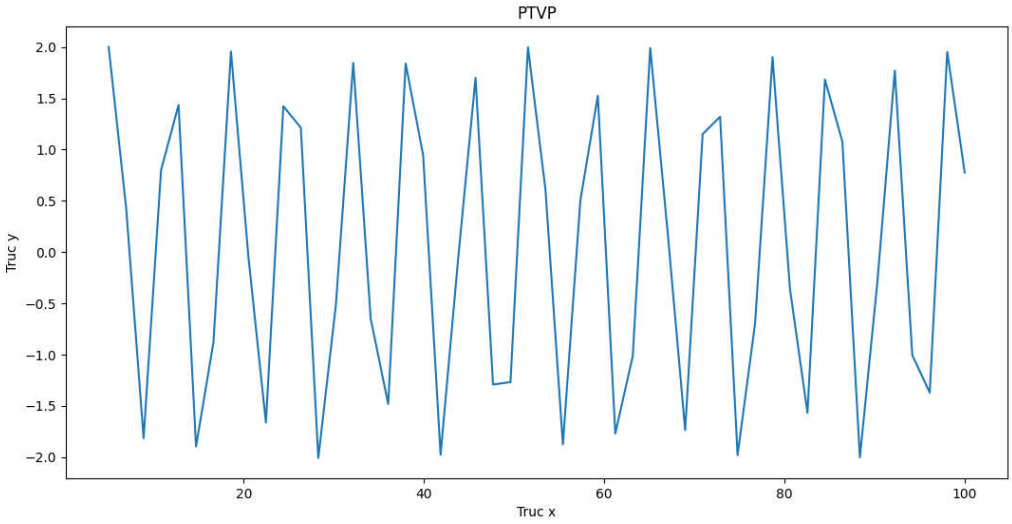
U[0]' = U[1]

U[1]' = (1 – $U[0]^2$)U[1] - U[0]

Ta thực hiện như sau:

```
def model(U, x):
    return [U[1], (1 - U[0]**2)*U[1] - U[0]]
U0 = [2, 0]
xs = np.linspace(5, 100)
Us = odeint(model, U0, xs)
ys = Us[:,0]
plt.xlabel("Truc x")
plt.ylabel("Truc y")
plt.title("PTVP")
plt.plot(xs,ys)
plt.show()
```

Lưu ý. Dòng return [U[1], (1 - U[0]**2)*U[1] - U[0]] phải lùi vào 1 tab.



VD2: Vẽ đồ thị nghiệm riêng của phương trình vi phân: $y'' - (1 - y^2)y' + y = e^x \sin x$, với điều kiện ban đầu $y(4) = 2$, $y'(4) = 0$ trên miền [4, 10].

Ta thực hiện như sau:

```
def model(U, x):
    return [U[1], (1 - U[0]**2) * U[1] - U[0] + math.e**x*math.sin(x)]
U0 = [2, 0]
xs = np.linspace(4, 10)
Us = odeint(model, U0, xs)
ys = Us[:,0]
plt.xlabel("Truc x")
plt.ylabel("Truc y")
plt.title("PTVP")
plt.plot(xs,ys)
```

Chủ đề 11. Phương trình sai phân

- *Ghi chú: Tham khảo Chủ đề 1 khi cần tính cho các hàm phức tạp.*
- Nhập thư viện:

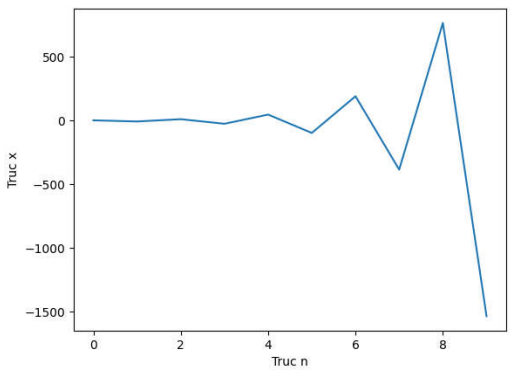
```
import math
import numpy as np
import matplotlib.pyplot as plt
```

1. Phương trình sai phân cấp 1

VD1: Vẽ đồ thị nghiệm riêng của phương trình sai phân x(n+1) + 2x(n) = 0 với điều kiện ban đầu x(0) = 3 trên miền [0, 10].

Ta thực hiện như sau:

```
N = 10
x = np.zeros(N, float)
x[0] = 3
for n in range(1, N):
    x[n] = -2*x[n-1]
plt.plot(x)
plt.xlabel('Truc x')
plt.ylabel('Truc y')
plt.show()
```



Lưu ý. Dòng x[n] = -2*x[n-1] phải lùi vào 1 tab.

VD2: Vẽ đồ thị nghiệm riêng của phương trình sai phân x(n+1) = (n+1)x(n) + (n+1)!n; với điều kiện ban đầu x(0) = 3 trên miền [0, 10].

Ta thực hiện như sau:

```
N = 10
x = np.zeros(N, float)
x[0] = 3
```

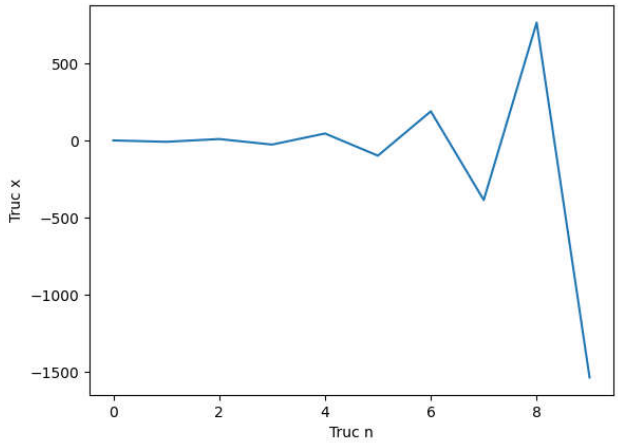
```
for n in range(1, N):
    x[n] = n*x[n-1] + math.factorial(n)*(n-1)
plt.plot(x)
plt.xlabel('Truc n')
plt.ylabel('Truc x')
plt.show()
```

2. Phương trình sai phân cấp 2

VD1: Vẽ đồ thị nghiệm riêng của phương trình sai phân x(n+2) – 5x(n+1) + 6x(n) = 0, với điều kiện ban đầu x(0) = 2, x(1) = 5 trên miền [0, 10].

Ta thực hiện như sau:

```
N = 10
x = np.zeros(N, float)
x[0] = 2
x[1] = 5
for n in range(2, N):
    x[n] = 5*x[n-1] - 6*x[n-2]
plt.plot(x)
plt.xlabel('Truc n')
plt.ylabel('Truc x')
plt.show()
```



Lưu ý. Dòng x[n] = 5*x[n-1] - 6*x[n-2] phải lùi vào 1 tab.

VD2: Vẽ đồ thị nghiệm riêng của phương trình sai phân: x(n+2) – 5x(n+1) + 6x(n) = n^2 + 2n + 3, với điều kiện ban đầu x(0) = 2, x(1) = 5 trên miền [0, 10].

Ta thực hiện như sau:

```
N = 10
x = np.zeros(N, float)
x[0] = 2
```

```
x[1] = 5
for n in range(0, N-2):
    x[n+2] = 5*x[n+1] - 6*x[n] + n**2 + 2*n + 3
plt.plot(x)
plt.xlabel('Truc n')
plt.ylabel('Truc x')
plt.show()
```

Phụ lục: Download, cài đặt và chạy “Hello World”

1. Download và cài đặt

Để lập trình python cần cài đặt: python và PyCharm (Pycharm là một công cụ để soạn thảo và chạy python (IDE), có thể sử dụng IDE khác hoặc không cần dùng IDE)

Bước 1:

@ Download python tại: https://www.python.org/downloads/

@ Cài đặt như các phần mềm khác

Bước 2:

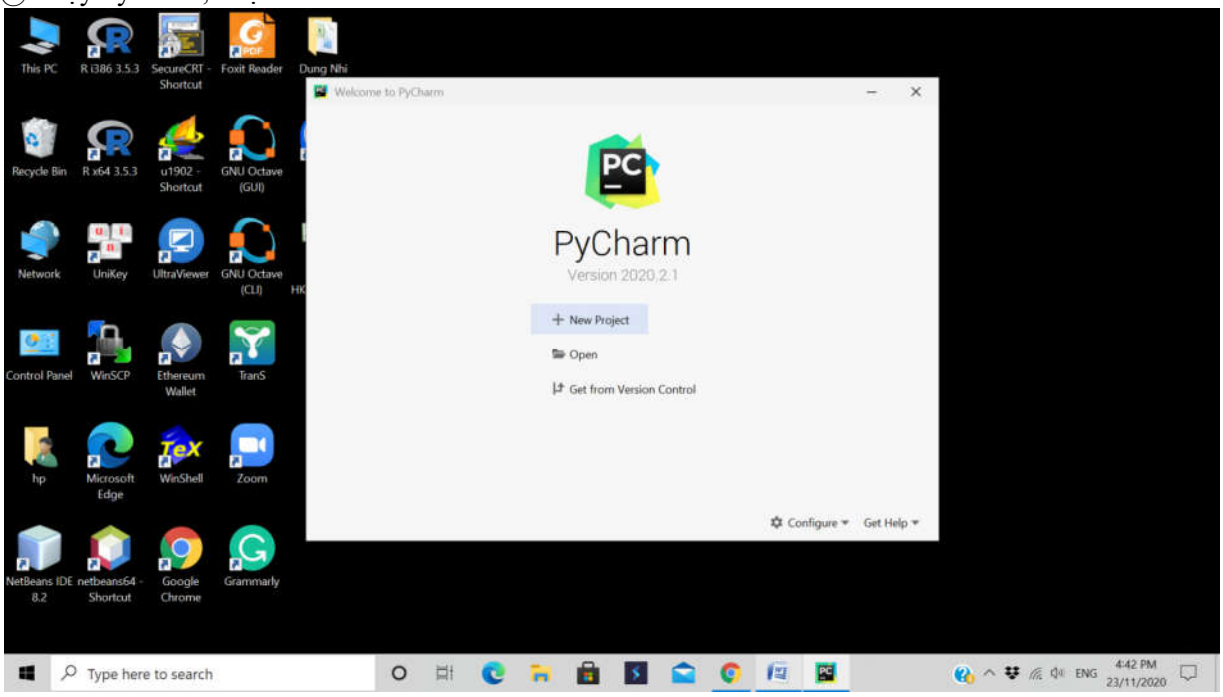
@ Download PyCharm tại: https://www.jetbrains.com/pycharm/download/

(chọn bản Community)

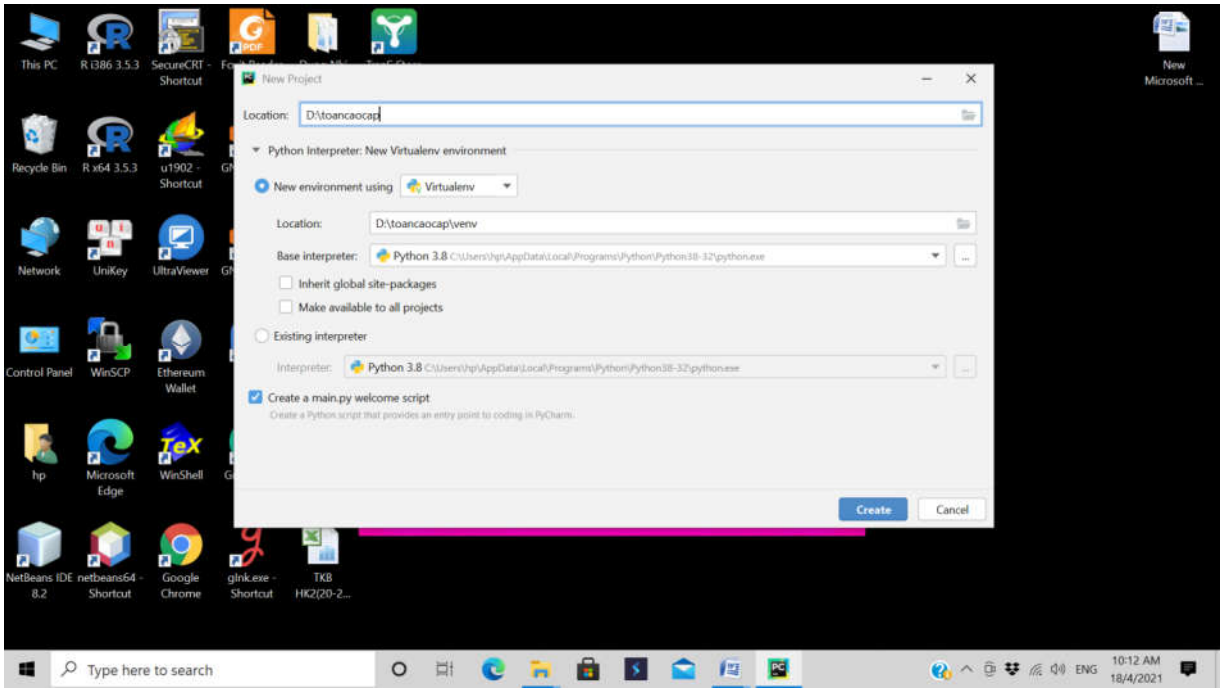
@ Cài đặt như các phần mềm khác

2. Chạy “Hello World”

@ Chạy Pycharm, được cửa sổ:



@ click vào + New Project; Tại Location gõ: D:\toancaocap; click nút Create

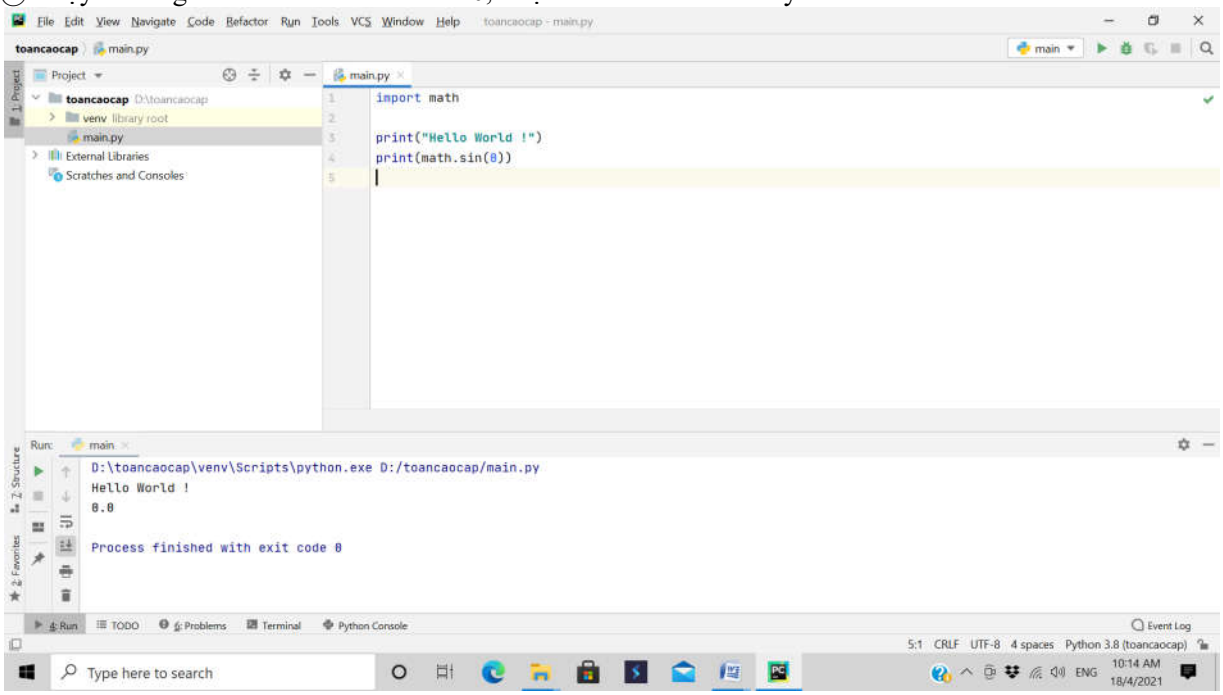


@ Trong cửa sổ bên phải, đây chính là file main.py; xóa hết các câu lệch trong file này, gõ các câu lệnh:

```
import math

print("Hello World !")
print(math.sin(0))
```

@ Chạy chương trình: nhấn nút Shift+F10, được cửa sổ như dưới đây



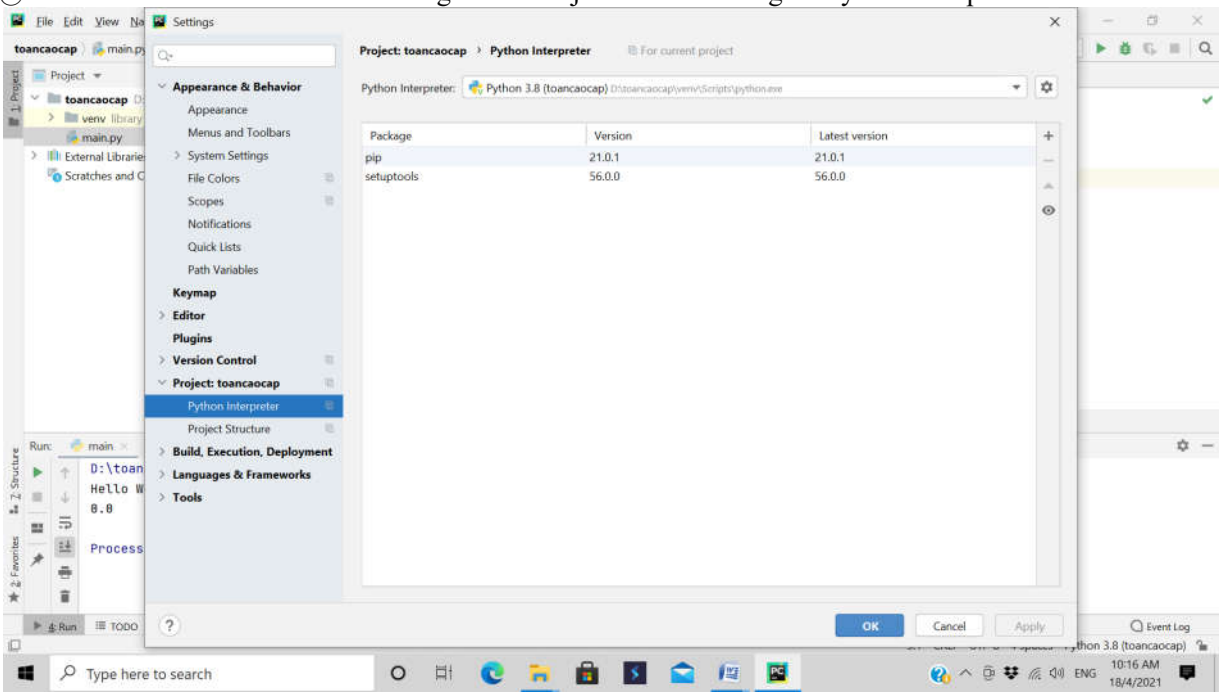
Kết luận: Bạn đã chạy thành công "Hello World"

3. Cài đặt thư viện

Để chạy được hết các ví dụ trong tài liệu này, cần phải cài đặt các thư viện: numpy, scipy, sympy, matplotlib

Mô tả cài numpy (các thư viện khác cài tương tự)

@ Trên thanh Menu vào File → Settings… → Project: toandaicuong → Python Interpreter



@ click vào dấu + ở bên phải; gõ numpy và click Install Package