RESUMEN DE LO VISTO EN CLASE

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

INTRODUCCIÓN

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional.

https://g.co/kgs/6i5Zuh

1. VARIABLES EN PYTHON

En Python no es necesario declarar el tipo de vatiable, solo se crea y dependiendo del valor que se le asigne, pyton gestionara su tipo.

Ejemplo:

```
entero = 1
text = 'hola mundo'
boolean = True
flotante = 3.5
lista = [1,2,3,4]
tupla = (1,2,3,4)
complejo = 5j

print ("entero=",type(entero))
print ("text=",type(text))
print ("boolean=",type(boolean))
print ("flotante=",type(flotante))
print ("lista=",type(lista))
print ("tupla=",type(tupla))
print ("complejo=",type(complejo))
```

```
entero= <class 'int'>
text= <class 'str'>
boolean= <class 'bool'>
flotante= <class 'float'>
lista= <class 'list'>
tupla= <class 'tuple'>
complejo= <class 'complex'>
```

Es muy importante gestionar bien los datos para que la memoria ram pueda funcionar adecuadamente al ejecutar el codigo.

2. CASTEO DE DATOS

Es tomar un tipo de datos para que se comporte como otro, por ejemplo tomar un string para que se comporte como un entero.

```
a='3'
b='7'
r=a+b
print('r=',r)
a1=int(a)
b1=int(b)
r1=a1+b1
print('r1=',r1)
```

```
r= 37
```

```
r1= 10
```

3. OPERACIONES ARITMÉTICAS

Estas son algunas de las operaciones aritmericas que se pueden hacer en python.

```
print("Suma = ", 3+7)
print("Resta = ", 3-7)
print("Multiplicación = ", 3*7)
print("División = ", 10/3)
print("División entera = ", 10//3)
print("Residuo = ", 10%3)
print("Potencia = ", 3**7)
print("Raiz = ", 2**0.5)
```

```
Suma =  10
Resta =  -4
Multiplicación =  21
División =  3.3333333333333335
División entera =  3
Residuo =  1
Potencia =  2187
Raiz =  1.4142135623730951
```

4.GENERAR NUMEROS ALEATOREOS

En Pyton se pueden generar números "aleatoreos" por medio de la librería de radom, llamando algunas de sus funiones con una semilla.

```
import random

print('Generar un número entero del 3 al 7: ', random.randrange(3,7)) #con randrange
print('Seleccionar aleatoreamente de opciones: ', random.choice(["uno","dos","tres"])) #con choice
print('Generar un número decimal entre 0 y 1: ', random.random()) #con random
print('Generar un número entero entre 0 y 3: ', int(3 * random.random())) #con random
print('Generar un número decimal entre 0 y 3: ', 3 * random.random()) #con random
print('imprimir una letra aleatoria: ', random.choice("QWERTYUIOP")) #con randrange
```

```
Generar un número entero del 3 al 7:  3
Seleccionar aleatoreamente de opciones:  tres
Generar un número decimal entre 0 y 1:  0.7165026032063623
Generar un número entero entre 0 y 3:  0
Generar un número decimal entre 0 y 3:  0.9024662063034348
imprimir una letra aleatoria:  I
```

5. LIBRERIA MATH

Esta libreria nos permite hacer operaciones matemáticas más complejas.

```
import math
print("Factorial=",math.factorial(3))
print("PI=",math.pi)
print("E=",math.e)
print("seno=",math.sin(math.pi/6))
print("aseno=",math.degrees(math.asin(0.5)))
print("Logaritmo Natural=",math.log(math.exp(2)))
print("Logaritmo en base 10=",math.log10(100))
print("Logaritmo en base 2=",math.log2(8))
print("Potencia=",math.pow(3,2))
print("Raiz cuadrada=",math.sqrt(4))
```

```
Factorial= 6
PI= 3.141592653589793
E= 2.718281828459045
seno= 0.49999999999999994
aseno= 30.000000000000004
```

```
Logaritmo Natural= 2.0
Logaritmo en base 10= 2.0
Logaritmo en base 2= 3.0
Potencia= 9.0
Raiz cuadrada= 2.0
```

6. OPERACIONES CON CADENAS

Una cadena es basicamentre una arreglo de caracteres, por lo que se pueden hacer con este tipo de datos casi todas las operaciones que se harían con un arreglo.

```
#Suma o concatenación
x="Un divertido"+"programa"+"de"+"radio"
print(x)
#Multiplicación de una cadena con un número
y=3*"programas"
print(y)
#Obtener longitud de una cadena
print('longitud de la cadena:',len(y))
#Acceder  a una posición de la cadena
cadena = "programa"
print('acceder a una posicion determinada:',cadena[1])
print('acceder a la última posición:',cadena[-1])
```

```
Un divertidoprogramaderadio
programasprogramasprogramas
longitud de la cadena: 27
acceder a una posicion determinada: r
acceder a la última posición: a
```

7. TUPLA

Una Tupla es un conjunto de datos ordenados de cualquier tipo, los cuales no pueden ser modificados, borrados o agregar nuevos items despues de su creación. Para asignar una tupla se usan parentesis () y cada valor va separado por comas.

```
tupla = ('cadena de texto', 15, 2.8, 'otro dato', 25,25)
print(tupla)
print(type(tupla))
print(tupla[1:4])
print('cuantas veces hay un elemento=',tupla.count(25))
print('dice en que posicion esta el elemento=',tupla.index(15))

#codigo que genera error debido a que es una tupla, por lo tanto no se puede modificar el contenido de la posición [0
#tupla[0]=2
```

```
('cadena de texto', 15, 2.8, 'otro dato', 25, 25)
<class 'tuple'>
(15, 2.8, 'otro dato')
cuantas veces hay un elemento= 2
dice en que posicion esta el elemento= 1
```

8. LISTA

Una Lista es un conjunto de datos ordenados de cualquier tipo, los cuales pueden ser modificados, borrados o agregar nuevos items despues de su creación. Para asignar una lista se usan corchetes [] y cada valor va separado por comas.

```
a = [1,2,3,"hola", [10, "nunca", 90], -32]


print ("Toda la lista= ",a)
print(type(a))
print ("Longitud de la lista= ", len(a))
print("Acceder a un rango de posiciones=",a[0:4])
```

```
print("Acceder a una posición específica= ",a[4])
print("Acceder a la última posición= ",a[-1])
a[1:-1]

#Asignar un nuevo elemento
a.append("Nuevo último")
print ("Toda la lista= ",a)
```

```
Toda la lista=  [1, 2, 3, 'hola', [10, 'nunca', 90], -32]
<class 'list'>
Longitud de la lista=  6
Acceder a un rango de posiciones= [1, 2, 3, 'hola']
Acceder a una posición específica=  [10, 'nunca', 90]
Acceder a la última posición=  -32
Toda la lista=  [1, 2, 3, 'hola', [10, 'nunca', 90], -32, 'Nuevo último']
```

recorres una lista

```
for i in range(len(a)):
    print (i, "-->", a[i])
```

```
0 --> 1
1 --> 2
2 --> 3
3 --> hola
4 --> [10, 'nunca', 90]
5 --> -32
6 --> Nuevo último
```

```
for i in a:
    print (i)
```

```
1
2
3
hola
[10, 'nunca', 90]
-32
Nuevo último
```

Acceder al los elementos de una lista

```
print (a[4])
print (a[4][0])
print (a[4][1])
print (a[4][1][2:])
print (a[4][1][:3])
print (a[4][1][1:3])
```

```
[10, 'nunca', 90]
10
nunca
nca
nun
un
```

Diccionarios

Los diccionarios son estructuras de datos en donde el elemento está asociada a una clave y no se accede por medios de indice como se hace en las tuplas y listas.

Las claves pueden ser de distintos tipos de datos, y no pueden repetirce

```
# Definir una variable diccionario
```

```
futbolistas = dict()

futbolistas = {
    1: "Casillas", 3: "Piqué",
    5: "Puyol", 6: "Iniesta",
    7: "Villa", 8: "Xavi Hernández",
    9: "Torres", 11: "Capdevila",
    14: "Xavi Alonso", 15: "Ramos",
    16: "Busquets"
}

# Recorrer el diccionario, imprimiendo clave - valor
print ("Vemos que los elementos no van \"ordenados\":")
for k, v in futbolistas.items():
    print ("{} --> {}".format(k,v))

# Nº de elementos que tiene un diccionario
numElemen = len(futbolistas)
print ("\nEl número de futbolistas es de {}".format(numElemen))

# Imprimir las claves que tiene un diccionario
keys = futbolistas.keys();
print ("\nLas claves de nuestro diccionario son : {}".format(keys))

# Imprimir los valores que tiene un diccionario
values = futbolistas.values();
print ("\nLos valores de nuestro diccionario son : {}".format(values))

# Obtener el valor de un elemento dada su clave
elem = futbolistas.get(6)
print ("\nEl futbolista con clave 6 es {}".format(elem))

# Insertamos un elemento en el diccionario
## si la clave ya existe, el elemento NO se inserta
futbolistas.setdefault(10, 'Cesc')
print ("\nInsertamos el elemento con clave 10 y valor Cesc")
print ("Ahora el diccionario queda : {}".format(futbolistas))

# Añadimos, de otro modo, un elemento al diccionario
## si la clave ya existe, el valor se cambia por este nuevo
futbolistas[22] = 'Navas'
print ("\nAñadimos un nuevo elemento, ahora el diccionario queda: ")
print (futbolistas)

# Eliminamos un elemento del diccionario dada su clave
futbolistas.pop(22)
print ("\nEliminamos el elemento con clave 22")
print ("Ahora el diccionario queda: {}".format(futbolistas))

# Hacemos una copia del diccionario
futbolistas_copia = futbolistas.copy()
print ("\nLa copia del diccionario tiene los valores:")
print (futbolistas_copia)

# Borramos los elementos de un diccionario
futbolistas_copia.clear()
print ("\nVaciamos el diccionario nuevo creado, ahora los valores en el: {}".format(futbolistas_copia))

# Creamos un diccionario a partir de una lista de claves
keys = ['nombre', 'apellidos', 'edad']
datos_usuario = dict.fromkeys(keys, 'null')
print ("\nCreamos un diccionario a partir de la lista de claves:")
print (keys)
print ("y con valor 'null'")
print ("Así el diccionario queda: {}".format(datos_usuario))

# Comprobamos si existe o no una clave en un diccionario
if 2 in futbolistas:
```

```
        print ("\nEl futbolista con la clave 2 existe en el diccionario.")
    else:
        print ("\nEl futbolista con la clave 2 NO existe en el diccionario.")

    if 8 in futbolistas:
        print ("\nEl futbolista con la clave 8 existe en el diccionario.")
    else:
        print ("\nEl futbolista con la clave 8 NO existe en el diccionario.")

    # Devolvemos los elementos del diccionario en una tupla
    tupla = futbolistas.items()
    print ("\nEl diccionario convertido en tupla queda así:")
    print (tupla)

    # Unimos dos diccionarios existentes
    suplentes = {
        4:'Marchena', 12:'Valdes', 13:'Mata',
        17:'Arbeloa', 19:'Llorente', 20:'Javi Martinez',
        21:'Silva', 23:'Reina'
    }

    print ("\nUnimos el diccionario: ")
    print (futbolistas)
    futbolistas.update(suplentes) # Aquí hacemos la unión de los diccionarios
    print ("con el diccionario:")
    print (suplentes)
    print ("siendo el resultado:")
    print (futbolistas)
```

```
     Vemos que los elementos no van "ordenados":
     1 --> Casillas
     3 --> Piqué
     5 --> Puyol
     6 --> Iniesta
     7 --> Villa
     8 --> Xavi Hernández
     9 --> Torres
     11 --> Capdevila
     14 --> Xavi Alonso
     15 --> Ramos
     16 --> Busquets

     El número de futbolistas es de 11

     Las claves de nuestro diccionario son : dict_keys([1, 3, 5, 6, 7, 8, 9, 11, 14, 15, 16])

     Los valores de nuestro diccionario son : dict_values(['Casillas', 'Piqué', 'Puyol', 'Iniesta', 'Villa', 'Xavi He

     El futbolista con clave 6 es Iniesta

     Insertamos el elemento con clave 10 y valor Cesc
     Ahora el diccionario queda : {1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernánde

     Añadimos un nuevo elemento, ahora el diccionario queda:
     {1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernández', 9: 'Torres', 11: 'Capdevi

     Eliminamos el elemento con clave 22
     Ahora el diccionario queda: {1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernándes

     La copia del diccionario tiene los valores:
     {1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernández', 9: 'Torres', 11: 'Capdevi

     Vaciamos el diccionario nuevo creado, ahora los valores en el: {}

     Creamos un diccionario a partir de la lista de claves:
     ['nombre', 'apellidos', 'edad']
     y con valor 'null'
     Así el diccionario queda: {'nombre': 'null', 'apellidos': 'null', 'edad': 'null'}

     El futbolista con la clave 2 NO existe en el diccionario.
```

```
El futbolista con la clave 8 existe en el diccionario.

El diccionario convertido en tupla queda así:
dict_items([(1, 'Casillas'), (3, 'Piqué'), (5, 'Puyol'), (6, 'Iniesta'), (7, 'Villa'), (8, 'Xavi Hernández'), (9

Unimos el diccionario:
{1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernández', 9: 'Torres', 11: 'Capdevi
con el diccionario:
{4: 'Marchena', 12: 'Valdes', 13: 'Mata', 17: 'Arbeloa', 19: 'Llorente', 20: 'Javi Martinez', 21: 'Silva', 23:
siendo el resultado:
{1: 'Casillas', 3: 'Piqué', 5: 'Puyol', 6: 'Iniesta', 7: 'Villa', 8: 'Xavi Hernández', 9: 'Torres', 11: 'Capdevi
```

MATRICES Y VECTORES

La matrices basicamente son arreglos o vectores de dos dimenciones y que pueden tener 1 o varias capas.

Para trabajar con Matrices y vectores en Python su puede usar la libreria de **NumPy** el cual trabaja con vectores y matrices de forma natural.

```
import numpy as np

a = np.array([[1,2,3,4,5],
              [5,4,3,2,1],
              [9,8,7,6,5],
              [7,6,5,6,7],
              [2,2,2,3,3],
              [4,3,4,3,4],
              [5,1,1,4,1]])

# shape sirve para mirar la forma de la matriz,
print("a shape", a.shape)
# shape[0] es para ver la forma de las filas
print("a rows", a.shape[0])
# shape[1] es para ver la forma de las columnas
print("a cols", a.shape[1])

v = np.array([2,3,4,5,6,7,3,12])
print("v shape", v.shape)
print("v elems", v.shape[0])
```

```
a shape (7, 5)
a rows 7
a cols 5
v shape (8,)
v elems 8
```

Para mostrar los datos de la matriz, se usan los indices así:

Porción de la Matriz

arreglo [fila : fila]

arreglo [fila : fila , Columna : Columna]

arreglo [: , Columna]

arreglo [: , Columna : Columna]

Elemento especifico

arreglo [fila , Columna]

```
print("una fila        " ,a[1])
print("una fila        ",a[1,:])
print("una columna     ",a[:,0])
print("un elemento     ",a[6,4])
print("varias filas    \n",a[1:4])
```

```
print("varias columnas \n",a[:,1:3])
print("una porcion     \n",a[2:5,1:5])
```

```
una fila         [5 4 3 2 1]
una fila         [5 4 3 2 1]
una columna      [1 5 9 7 2 4 5]
un elemento      1
varias filas
 [[5 4 3 2 1]
 [9 8 7 6 5]
 [7 6 5 6 7]]
varias columnas
 [[2 3]
 [4 3]
 [8 7]
 [6 5]
 [2 2]
 [3 4]
 [1 1]]
una porcion
 [[8 7 6 5]
 [6 5 6 7]
 [2 2 3 3]]
```

Operaciones con los elementos

```
# Valor maximo
print('Valor maximo: ',np.max(a))
# Valor minimo
print('Valor minimo: ',np.min(a))
# Valor media
print('Valor media: ',np.mean(a))
# Valor suma
print('Valor suma: ',np.sum(a))
```

```
Valor maximo:  9
Valor minimo:  1
Valor media:  3.942857142857143
Valor suma:  138
```

Una matriz puede tener una o varios canales o dimentisones

```
m = np.random.randint(10, size=(3,3,3))

print(m)
```

```
[[[5 4 3]
  [2 3 8]
  [8 6 7]]

 [[9 3 7]
  [1 8 1]
  [6 1 7]]

 [[7 9 7]
  [1 1 7]
  [4 7 4]]]
```

Operaciones por ejes

axis(0)= columnas axis(1)= filas axis(2)= capas

```
# Valor maximo axis = 0
print('Valor maximo: ', np.max(m, axis = 0))
```

```
# Valor minimo axis = 1
print('Valor minimo: ', np.min(m,  axis = 1))

# Valor media axis = 2
print('Valor media: ',np.mean(m,  axis = 2))
```

```
Valor maximo:  [[9 9 7]
 [2 8 8]
 [8 7 7]]
Valor minimo:  [[2 3 3]
 [1 1 1]
 [1 1 4]]
Valor media:  [[4.         4.33333333 7.        ]
 [6.33333333 3.33333333 4.66666667]
 [7.66666667 3.         5.        ]]
```

Generación de matrices y vectores

```
print("matrix identidad\n", np.eye(3))
print("vector de ceros", np.zeros(4))
print("matriz de ceros\n", np.zeros((3,2)))
print("matriz de unos\n", np.ones((2,3)))
print("vector rango", np.arange(10))
print("vector rango", np.arange(5,10))
print("vector espacio lineal", np.linspace(0,9,4))
print("matriz aleatoria según distribución uniforme [0,1]\n", np.random.random(size=(3,5)))
print("vector aleatorio de enteros entre 0 y 5", np.random.randint(5, size=10))
```

```
matrix identidad
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
vector de ceros [0. 0. 0. 0.]
matriz de ceros
 [[0. 0.]
 [0. 0.]
 [0. 0.]]
matriz de unos
 [[1. 1. 1.]
 [1. 1. 1.]]
vector rango [0 1 2 3 4 5 6 7 8 9]
vector rango [5 6 7 8 9]
vector espacio lineal [0. 3. 6. 9.]
matriz aleatoria según distribución uniforme [0,1]
 [[0.35289119 0.30965251 0.24902438 0.89445822 0.90058872]
 [0.8890108  0.93654041 0.67578808 0.09631306 0.5908885 ]
 [0.51738606 0.11375485 0.30942529 0.78238528 0.04922465]]
vector aleatorio de enteros entre 0 y 5 [3 4 3 3 1 0 3 3 3 4]
```

Operaciones Vectoriales

Caba resaltar que existen 2 clases de multiplicación, la normal con * que es la multiplicacion entre elementos y la dot Que es una mulriplicacion matricial

```
v = np.array([10,12,13,15,20])
a =  np.random.randint(10, size=5)
print(v)
print(a)
print(v+1)
print(v*2)
print(v.dot(a))
```

```
[10 12 13 15 20]
[7 6 0 5 7]
[11 13 14 16 21]
[20 24 26 30 40]
357
```

```
a = np.array([[1,2,3],[4,5,6]])
b = np.array([[6,5,4],[3,2,1]])

print(a)
print(b)
print("--")

print("a+b\n",a+b)
print("a**2\n", a**2)
print("a*b\n",a*b)
print("a x b'\n",a.dot(b.T))
print("a' x b\n",a.T.dot(b))
```

```
[[1 2 3]
 [4 5 6]]
[[6 5 4]
 [3 2 1]]
--
a+b
 [[7 7 7]
 [7 7 7]]
a**2
 [[ 1  4  9]
 [16 25 36]]
a*b
 [[ 6 10 12]
 [12 10  6]]
a x b'
 [[28 10]
 [73 28]]
a' x b
 [[18 13  8]
 [27 20 13]
 [36 27 18]]
```

Los array tambien pueden funcionar con expresiones boolean

```
a = np.array([1,8,4,10,-4,5])
print(a)
print(a>4)
i = np.array([False, True, False, True, False, True])
print(i)

a[a>4]
```

```
[ 1  8  4 10 -4  5]
[False  True False  True False  True]
[False  True False  True False  True]
array([ 8, 10,  5])
```

Expresiones compactas:

Son son formas más cortas que realizar elñ codigo

```
a=15
if a > 7:
    s = "mayor que 7"
else:
    s = "menor que 17"

print(s)
```

```
mayor que 7
```

```
a = 15
s = "mayor que 7" if a > 7 else "menor que 7"
print(s)
```

```
mayor que 7
```

```
l=[]
for i in range(3):
    l.append(i)
print(l)
```

```
[0, 1, 2]
```

```
l=[i for i in range(3)]
print(l)
```

```
[0, 1, 2]
```

```
a = [10, -4, 20, 5]

#o = ["10A", "-4B", "20A", "5A"]

o = []
for i in a:
    if i<0:
        o.append(str(i)+"B")
    else:
        o.append(str(i)+"A")

print(o)
```

```
['10A', '-4B', '20A', '5A']
```

```
a = [10, -4, 20, 5]
def convert(x):
    return str(x)+"B" if x<0 else str(x)+"A"

o = [convert(i) for i in a]
print(o)
```

```
['10A', '-4B', '20A', '5A']
```

```
r = []
for i in range(10):
    r.append("el numero "+str(i))
print(r)
```

```
['el numero 0', 'el numero 1', 'el numero 2', 'el numero 3', 'el numero 4', 'el numero 5', 'el numero 6', 'el nu
```

```
r = ["el numero "+str(i) for i in range(10)]
print(r)
```

```
['el numero 0', 'el numero 1', 'el numero 2', 'el numero 3', 'el numero 4', 'el numero 5', 'el numero 6', 'el nu
```

```
frutas = {'Fresa':'roja', 'Limon':'verde', 'Papaya':'naranja', 'Manzana':'amarilla', 'Guayaba':'rosa'}
for nombre, color in frutas.items():
    print (nombre, "es de color", color)

r = [nombre+" es de color "+color for nombre, color in frutas.items()]
```

```
Fresa es de color roja
Limon es de color verde
Papaya es de color naranja
Manzana es de color amarilla
Guayaba es de color rosa
```

```
r
```

```
['Fresa es de color roja',
 'Limon es de color verde',
 'Papaya es de color naranja',
 'Manzana es de color amarilla',
 'Guayaba es de color rosa']
```

MATPLOIB

Es una libreria que nos permite graficar

```
x = np.linspace(-4,4,100)
x
```

```
array([-4.        , -3.91919192, -3.83838384, -3.75757576, -3.67676768,
       -3.5959596 , -3.51515152, -3.43434343, -3.35353535, -3.27272727,
       -3.19191919, -3.11111111, -3.03030303, -2.94949495, -2.86868687,
       -2.78787879, -2.70707071, -2.62626263, -2.54545455, -2.46464646,
       -2.38383838, -2.3030303 , -2.22222222, -2.14141414, -2.06060606,
       -1.97979798, -1.8989899 , -1.81818182, -1.73737374, -1.65656566,
       -1.57575758, -1.49494949, -1.41414141, -1.33333333, -1.25252525,
       -1.17171717, -1.09090909, -1.01010101, -0.92929293, -0.84848485,
       -0.76767677, -0.68686869, -0.60606061, -0.52525253, -0.44444444,
       -0.36363636, -0.28282828, -0.2020202 , -0.12121212, -0.04040404,
        0.04040404,  0.12121212,  0.2020202 ,  0.28282828,  0.36363636,
        0.44444444,  0.52525253,  0.60606061,  0.68686869,  0.76767677,
        0.84848485,  0.92929293,  1.01010101,  1.09090909,  1.17171717,
        1.25252525,  1.33333333,  1.41414141,  1.49494949,  1.57575758,
        1.65656566,  1.73737374,  1.81818182,  1.8989899 ,  1.97979798,
        2.06060606,  2.14141414,  2.22222222,  2.3030303 ,  2.38383838,
        2.46464646,  2.54545455,  2.62626263,  2.70707071,  2.78787879,
        2.86868687,  2.94949495,  3.03030303,  3.11111111,  3.19191919,
        3.27272727,  3.35353535,  3.43434343,  3.51515152,  3.5959596 ,
        3.67676768,  3.75757576,  3.83838384,  3.91919192,  4.        ])
```