به نام خدا

پایتون

\_پیشرفته

\_تابع

Input  Run  output

Def f() :

Print(‘Python’)

F()  Python

F()  Python  call

توابع عمدتا برای ننوشتن کد تکراری در برنامه بکار می رود

ابتدا از کلمه کلیدی استفاده می کنیم – سپس حتما () قرار می دهیم و : می گزاریم چون زیر آن بلاک حاوی استیتمتن وجود داره.

سپس با رعایت فرو رفتگی کد می نویسی می کنیم

Def my\_function(s) :

Print(‘Hello ’ + s)

My\_function(‘ali’)  Hello ali

My\_function(‘taha’)  Hello taha

Def f(x) :

Return 2 \* x + 1

Y = f(3)

Print(y)  7

Def f(x, y) :

Return x + 3 \* y

Z = F(1, 2)

INPUT x

Print(z)  7

FUNCTION f

OUTPUT f(x)

Def f(s, n) :

Print(s \* n)

F(‘hello’, 3)  hellohellohello

F(‘python’, 2)  pythonpython

متغیر هایی که در پارامتر های توابع تعریف می شوند – تایپ مشخصی ندارند. پس با هر نوع متغیری می تواند به تابع داد. اما باید حواسمان باشد که تابع ما رو چه تایپ هایی کار انجام می دهد

Def f(x) :

Return 2 \* x + 1

Def g(n) :

K = f(n) + 2 \* f(n + 1)

Print(k)

G(3)  20

در یک تابع می توان از یک تابع که خارچ از ان تعریف شده نیز کال یا صدا کرد – و از آن استفاده کرد

\_متغیر محلی

Def f() :

# local variable

A = ‘python’

Print(a)

F()  python

متغیر هایی هستند که داخل یک تابع نوشته میشوند و فقط داخل آن تابع شناخته می شوند – و بیرون تابع اگر از آن ها نام ببری ارور ( متغیر ناشناخته) بازمیگردد

Globalکلمه کلیدی

a = ‘python’

def f() :

global a

print(a)

f()  python

وقتی در تابع از یک متغیری که خارج از تابع تعریف شده – کد بزنی خطا می دهد.

باید از کلمه کلیدی اش استفاده کنی تا اون متغیر که خارج از تابع هست رو بشناسد. و تغیراتی که رو اون تو تابع می دهی را رسما رو متغیر اعمال کند.

Def f() :

Global s

S = s + ‘reza’

Print(s)

S = ‘ali’  
f() alireza

Print(s)  alireza

A = 5

Def f() :

A = 6

Print(a)

Print(a)  5

F()  6

A = 5

Def g() :

Global a

A = 7

Print(a)

G()  7

Print(a)  7

A = 2

B = 3

Def f(c) :

Global b

B = 5

Return a \* b \* c

? 5 input

Print(f(4))  40

Print(b)  5

اولویت متغیر در تابع:

اول\_ متغیر محلی

دوم\_ متغیر سراسر

سوم\_ اگر نه متغیر محلی تعریف شد و نه متغیری که خارج از تابع بود – برای تابع سراسری نشد

بدون سراسری کردن متغیر خارج از تابع می توان از متغیر خارجی فقط از مقدارش استفاده کرد.

نکته: اگر متغیر خارج از تابع هم تعریف نشده بود – با خطا مواجه میشویم

Def f(a, b) :

A -= 1

c = a + b

return a, b

x, y = f(2, 4)

# x, y = a, b

Print(x)  1

Print(y)  5

می توان خروجی در تابع را به عنوا لیست کرد – و بعد در چند متغیر قرار داد

Def f(s, n=1) :

Print(s \* n)

F(‘ali’, 2)  ali

F(‘sara’)  sara

در اینجا تعریف شده که در آرگومان پیشفرض برای ن 1 باشد – یعنی اگر در ورودی ن را ندادیم به طور پیشفرض 1 در نظر گرفته شود

Def op(a, b, c) :

Print(a + 2 \* b + c)

Op(a=1, c=2, b=3)  9

Def f(a, b, c=True, d=0) :

Print(a, b, c, d)

F(‘Python’, [5, 2], ‘ali’, 8)

F((‘Python’, [5, 2], ‘ali’)

(‘Python’, [5, 2])

توجه داشته باشیم – به مقادیری که در تعریف تابع مقدار پیشفرض می دهیم بهتر است چپ تر نسبت به مواردی که الزامی بر گرفتن هستند قرار دهیم.

نامشخص بودن تعداد آرگومان ها

Def f(x, y, z, w) :

Print(‘Hello’, y)

F(x = ‘Ali’, y = ‘Taha’, z = ‘Amin’, w = ‘Omid’)

Def f(\*a) :

Print(‘hello’ + a[1])

0 1 2 ...

F(‘ali’, ‘Taha’, ...)

زمانی که قبل از نام گزاری آرگومان در تابع \* قرار می دهیم – به معنی این است منظور کل آرگومان ها مد نظر است و با اندیس ها می توان به آنها دسترسی پیدا کرد

Def avg(\*x) :

Print(sum(x)/len(x))

Avg(1, 2, 3, 4, 5)  3

Def fun(a, \*b) :

Print(a + len(b))

Fun(1, 2, 3, ‘ali’)

در اینجا چون آ راست ترین آمده پس راست ترین مقدار هنگام دادن مقدار به داده در آ می رود

و هر آنچه بعد از آن در سمت چپ هست – داخل ب می رود و با اندیس قابل دسترسی است.

Def my\_function() :

Pass

به معنای این است که هنگام اجرا خطا نگیریم به خاطر خالی بودن تابع – تا بعدا کامل کنیم

\_تابع بازگشتی

Recursive call

Def r() :

...

R()

...

R()

تابع محاسبه فاکتوریل ( غیر بازگشتی)

Def factorial(n) :

If n < 0 :

Return 0

Elif n == 0 0r n ==1 :

Return 1

Else :

F = 1

While (n > 1) :

F \*= n

N -= 1

Return f

فاکتوریل (بازگشتی)

Def factorial(n) :

If (n==0 or n==1) :  
 return 1

Else :

Return n \* factorial(n - 1)

#

Factorial(5)

= 5 \* Factorial(4)

= 5 \* 4 \* Factorial(3)

= 5 \* 4 \* 3 \* Factorial(2)

= 5 \* 4 \* 3 \* 2 \* Factorial(1)

= 5 \* 4 \* 3 \* 2 \* 1

= 120

تعیین زوج بودن

Def even(n) :

If n==0 :

Return True

Return not even(n-1)

Lambda عبارت

Def f(x, y) :

Return x + 2 \* y

F(2, 3)

...

F = lambda x, y : x + 2 \* y

F(2, 3)  8

Filter تابع

Even = lambda x : x%2 == 0

Lst = [12, 18, 3, 17, 20, 19]

F = filter(even, lst)

List(f)  [12, 18, 20]

تک تک موارد داخل لیست را بررسی م یکند در تابعی که در ان قرار دادیم

و انانی که جوابشان صحیح یا درست شد را باقی گذاشت و بقیه را فیلتر یا حذف کرد.

Even = lambda x : x%2 == 0

Lst = [12, 18, 3, 17, 20, 19]

m = map(even, lst)

list(m) [True, True, False, False, True, False]

مانند بالا اما جواب درست یا غلط را برمی گرداند.

A = [1, 2, 3]

B = [4, 5, 6]

F = lambda x, y : x + y

List(map(f, a, b))  [5, 7, 9]

در اینجا مپ میاد با این تابعی که به ان دادیم – مقادیر زیر هم در دو لیست را با هم جمع می کند

\_فایل

<var> = open(<file>, ‘r’)

ما نمی توانیم تمام داده ها و حتی اطلاعات دیگر را به صورت دیتا تایپ ومتغیر دریافت کنیم.

می توانیم فایل ها را در پایتون دریافت کنیم و پردازش کنیم

در اینجا با کلمه کلیدی نام فایل و جایی که هست را می نویسیم. و با یک هندل اصلاح دستور باز نوشتم در فایل یا خوندن از فایل را می دهیم و کل موضوع را داخل یک متغیر قرار می دهیم.

Write\_

F = open(‘d:a.txt’, ‘w’)

f.write(‘Ali\n’)

f.write(‘Amin\n’)

f.write(‘Taha\n’)

f.close()

در متغیر نام شده فایلی را به هندل نوشتن که د دایرکتوری د وجود دارد باز کردیم

سپس با متدی که ذکر شده شروع به نوشتن کردیم. می توانستیم همه را در یک کد بنویسیم چون از خط بعد استفاده شده.

و حتما باید فایل را ببندیم.

With open(‘d:a.txt’, ‘w’) as f :

f.write(‘ali’\n)

f.write(‘Amin\n’)

f.write(‘Taha\n’)

در اینجا با کلمه کلیدی می توان دیگر از بستن فایل و نوشتن کد آن جلوگیری کرد

Writelines\_

F = open(‘d:a.txt’, ‘w’)

f.write(‘ali’\n)

f.write(‘Amin\n’)

f.write(‘Taha\n’)

f.close()

---------

F = open(‘d:a.txt’, ‘w’)

Lst = [‘Ali\n’, ‘Amin\n’, ‘Taha\n’]

f.writelines(lst)

f.close()

Readlines\_

F = open(‘d:a.txt’, ‘r’)

S = f.readlines()

Print(s)  [‘Alil\n’, ‘Amin\n’, ‘Taha\n’]

f.close()

read\_

f = open(‘d:a.txt’, ‘r’)

s = f.read()

print(s) 

Ali

Amin

Taha

f.close()

f = open(‘d:a.txt’, ‘r’)

s = f.read(8)

print(s) 

Ali

Amin

f.close()

tell\_

lst = [‘Python\n’, ‘java\n’, ‘php\n’]

f = open(‘d:x.txt’, ‘w’)

f.writelines(lst)

f.close()

f = open(‘d:x.txt’, ‘r’)

print(f.tell())  0

print(f.read(10)) 

python

jav

print(f.tell())  11

f.close()

f = open(‘d:x.txt’, ‘r’)

f.seek(1)

print(f.read(3))  yth

f.close()

‘a’

f = open(‘d:x.txt’, ‘a’)

f.writelines(‘C++’)

f.close()

Python

Java

Php

C++

Numpy \_کتابخانه

Import numpy as np

Score = np.array([13, 12, 6, 20, 0, 7, 12])

Print(score.ndim)  1

Print(score.shape)  (7,)

Andis  0 1 2 3 4 5 6

score  13 12 6 20 0 7 12

ابتدا کتابخانه نامپای را ایمپورت می کنیم و به یک اسمی که دلخاه هست اشاره می کنیم.

اگر اشاره به اسم خاصی نکنیم... باید هر بار از خود کلمه ناپای استفاده کنیم. در متغیری نامپای را صدا زدیم ( به اسمی که خودمان مخفف کردیم) سپس از کلاس نامپای تابع مربوط به آرایه را کال (صدا) می کنیم. و در پرانتز می توانیم یک لیست از اعداد بنویسیم یا یک متغیر که در اون یک لیست تنها از اعداد ساخته ایم را بنویسیم.

با پرینت اول – معلوم می کنیم که آرایه چند بعدی هست.

و با پرینت دوم – معلوممی کنیم که آرایه چند عدد داخلش قرار گرفته است.

همانطور که لیست ها و ساختمان داده ها اندیس دارند. آرایه ها هم اندیس دارند. اندیس آمنمها هم نیز از 0 شروع می شود.

متد های آرایه

Sort\_

Score = np.array([13, 12, 6, 20, 0, 7, 12])

S = np.sort(score)

Print(s)  [0, 6, 7, 12, 12, 13, 20]

Unique\_

Score = np.array([13, 12, 6, 20, 0, 7, 12])

Uscore = np.unique(score)

Print(uscore)  [0, 6, 7, 12, 13, 20]

عملیات بر روی آرایه

Score = np.array([13, 12, 6, 20, 0, 7, 12])

X = score / 2

Print(x)  array([6.5, 6., 3., 10., 0. , 3.5, 6.])

Sum\_

Score = np.array([13, 12, 6, 20, 0, 7, 12])

Print(np.sum(score))  70

Where

Score = np.array([13, 12, 6, 20, 0, 7, 12])

Print(np.where(score < 10, 10, 20)) 

[20, 20, 10, 20, 10, 10, 20]

هر کدوم از اعدا داخل اسکور اگر کمتر از 10 بود – 10 می گیره اگر در غیر این صورت بود – 20 می گیره.

0 1 2 3 4 5

A = np.array([11, 20, 13, 14, 18, 6])

X = np.where(a == 13)

Print(x)  2

A = np.array([11, 20, 13, 14, 18, 16])

X = np.where(a%2 == 0)

Print(x)  [1, 3]

Copy\_

Score = np.array([13, 12, 6, 20, 0, 7, 12])

C = score[1:6].copy()

Print(c)  [12, 6, 20, 0, 7, 12]

View\_

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

B = a.copy()

A[1] = 7

Print(a)  [1, 7, 3, 4]

Print(b)  [1, 2, 3, 4]

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

B = a.view()

A[1] = 7

Print(a)  [1, 7, 3, 4]

Print(b)  [1, 7, 3, 4]

Searcsorted\_

A = np.array([11, 20, 18, 14, 15, 13])

X = np.searchsorted(a, 12)

Print(x)  1 11, 12, 13, 14, 15, 18, 20

A = np.array(12, 9, 18)

X = [True, False, True]

B = a[x]

Print(b)  [12, 18]

Zeros, full\_

A = np.zeros(3)  a[0, 0, 0]

B = np.full(3, 20)  b[20, 20, 20]

C = np.arange(3)  c[0, 1, 2]

D = np.random.random(3)  d[ , , ]

Concatenate\_

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

B = np.array([4, 5])

C = np.concatenate((a, b))

Print(c)  [1, 2, 3, 4, 5]

Split\_

A = np.array([1, 2, 3, 4, 5, 6])

B = np.array\_split(a, 2)

Print(b)  [1, 2, 3], [4, 5, 6]

Print(b[0][1])  2

\_آرایه دو بعدی

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

| # | 0 | 1 | 2 | 3 |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 | 5 | 6 | 7 | 8 |
| 2 | 9 | 10 | 11 | 12 |

Print(x.shape)  (3, 4)

Print(x.ndim)  2

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

Print(

A[0]  [1, 2, 3, 4]

A[0][0]  1

A[1][1]  6

A[-1][-1]  8

A[0, 1:3]  [2, 3]

A[1, 1:4]  [6, 7, 8]

Reshape\_

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

B = a.reshape(2, 4)

B  array([[1, 2, 3, 4], [5, 6, 7, 8]])

Z = np.arange(6).reshape((2, 3))

Z  [[0, 1, 2], [3, 4, 5]]

محور ها

A = np.arange(9).reshape(3, 3)

Print(np.sum(a, axis = 0))

# 0 ستون =

Array([9, 12, 15])

A = np.arange(9).reshape(3, 3)

Print(np.sum(a, axis = 1))

# 1 ردیف =

Array([3, 12, 21])

Concatenate\_

A = np.random.randint(0, 100, size = (2, 3)) 

[[93, 39, 85], [5, 74, 62]]

B = np.random.randint(200, 300, size = (2, 3)) 

[[258, 299, 240], [248, 203, 220]]

Print(np.concatenate((a, b), axis= 0))

[[ 93, 39, 85],

[ 5, 74, 62],

[258, 299, 240],

[248, 203, 220]]

Swapaxes\_

a.swapaxes(0, 1) 

[[1, 2, 3],

[4, 5, 6]]

[[1, 4],

[2, 5],

[3, 6]]

Transpose\_

Np.transpose(a) 

[[1, 2],

[3, 4],

[5, 6],

[7, 8]]

2x4  4x2

[[1, 3, 5, 7],

[2, 4, 6, 8]]

Flipud\_

Np.flipud(a)

[[1, 2],

[3, 4],

[5, 6],

[7, 8]]



[[7, 8]

[5, 6],

[3, 4],

[1, 2]]

A = np.array([[10, 11, 7], [15, 13, 19]])

b = np.array([[10, 20, 4], [16, 13, 5]])

a == b



Array([[True, False, False], [False, True, False]])

(a == b).sum()  2

Pandas کتابخانه\_

Mydict = {

‘Name’ : [‘Ali’, Negar’, ‘Sara’, ‘Mahsa’],

‘Age’ : [37, 45, 29, 32],

‘Score’ : [19, 20, 17.5, 18.25]

}

Import pandas as pd

Df = pd.DataFrame(mydict)

پانداز یک کتابخانه برای ایجاد دیتافریم است. دیتا فریم ها می توانند اطلاعات را در دو سطر عمودی و افقی و به صورت جدول نمایش دهند. دیتافرم ها می توانند برای نمایش بهتر داده ها و دسترسی منطقی تر به داده های عظیم باشند.

\_دسترسی به ستون ها

Df[[‘Name’, ‘Age’]]

Iloc\_

Df.iloc[:, 0:2]

در اینجا ستون های 0 تا 2 – یعنی 0 و 1 را نمایش می دهیم.

Sort­\_

Sdf = df.sort\_values(‘Age’)

Sdf

دراینجا مقادیر ستون سن رو به ترتیب می کنیم. و داخل یک متغیر دیگر قرار می دهیم. اما دیتافریم اصلی تغییری ایجاد نمیشود.

Df.sort\_values(‘Age’, inplace= True)

در اینجا کار بالا صورت می گیرد اما در خود متغیر اصلی تغییر ایجاد می شود

Min, max\_

Df[‘Age’].max()  بیشترین عدد داخل ستون سن ها

Df[‘score’].min()  کمترین عدد داخل ستون امتیازات

Mean\_

Df.mean()

از ستون هایی که عدد دارند. میانگینشان را جدا میگیرد و جدا جلو هر کدام نمایش می دهد

Df[[‘score’]].mean()

میاد فقط ستون مورد نظر رو متانگینش رو می گیره

Describel\_

Df[[‘Score’]].describel()

تمام جزئیات مربوط به ستون امتیازات رو نمایش می دهد.

فیلتر کردن

Df[df[‘Age’] < 35]

ردیف هایی که سن انها کمتر از 35 است را نمایش می دهد.

Rename\_

Dfn = df.rename(columns={‘Score’ : ‘Grade’})

نام ستون امتیازات رو تغییر می دهیم

Replace\_

Df[‘Name’] = df[‘Name’].replace({‘Ali’ : ‘Alireza’})

در ستون اسم هر جا علی دید – تبدیل به علی رضا می کند

Drop\_

a = np.arange(12).reshape(4, 3)

df = pd.DataFrame(a, columns= [‘A’, ‘B’, ‘C’])

df.drop(columns ‘B’)

میاد ستون ب را کلا حذف می کند

Df.drop([‘A’, ‘C’], axis= 1)

Df.drop(columns= [‘A’, ‘C’])

چند ستون را حذف می کند

حذف سطر

Df.drop([1, 3])

Read\_excel, to\_excel

Df.to\_excel(‘d:student.xlsx’, index= False)

S = pd.read\_excel(‘d:student.xlsx’)

کد اول دیتا فرم های پانداز را به فایل اکسل تبدیل می کند.

و کد دوم فایل اکسل را می خواند و به دیتافرم های پانداز تبدیل می کند.

نکته: هر گاه از کد زیر در دیکشنری که می خواهیم آن را به دیتا فرم های پانداز تبدیل کنیم – بیاوریم... به معنای جای خالیاست و می توانیم با نامپای در او تغییراتی ایجاد کنیم.

Np.NaN

Isna\_

Frame[frame[‘col2’].isna()]

در اینجا میاد و در ستون دوم هرجا به کلمه کلیدی خالی رسید – ردیف اون رو میاورد.

Notna\_

Frame[frame[‘col1’].notna()]

در اینجا میاد و بررسی می کند در ستون اول کجا کلمه کلیدی خالی بکار رفته است – و ردیف اون رو حذف کرده – و اصلی رو نشون میده

Groupby\_

Dt = df.groupby(‘Team’)

Gt.first()

در اینجا دیتافریم دی اف را بر اساس تیم ها گروه و دسته می کنیم. دستور بعدی – اولین آیتم هر گروه را بر می گرداند.

Gt.get\_group(‘A’)

Gt.get\_group(‘B’)

آیتم های داخل هر گروه که گفتیم را بر می گرداند.

Gt.sum()

این دستور مجموع ستون های عددی هر گروه را بر می گرداند.

Gt.agg([‘max’])

از هر گروه بزرگترین آیتم را بر می گرداند.

Gt.agg([‘max’, ‘min’])

در این جا بزرگترین و کوچکترین را برمیگرداند.

matplotlib کتابخانه

Import matplotlib.pyplot as plt

X = np.array([0, 0.5, 1, 1.5, 2, 2.5, 3])

Y = 2 \* x

Plt.plot(x, y, ‘b-o’)

Plt.show()

کتابخانه مت پلات لیب برای رسم انواع نمودار مناسب است

یکی از انواع نمودار پلات یا نمودار خطی است که نیاز به دوپارامتر اصلی دارد که به عنواع ایکس و وای در تابع مورد نظر از کتابخانه قرار گرفته است. می توان از آرایه ها در اینجا استفاده کرد. پارامتر بعدی که در اینجا آورده شده است برای نوع نقطه ای که برای نقاط مختلف در خط نمودار هست تنظیم میشود. در این مدل دایره است

Plt.figure(figsize= (4, 2))

Plt.plot(x, x, label= ‘Linear’)

Plt.plot(x, x\*\*2, label= ‘Quadratic)

Plt.xlabel(‘X’)

Plt.ylabel(‘Y’)

Plt.title(‘plot’)

Plt.legend()

کد اول تنظیم اندازه و سایز کلیاست

با آرگومان لیبل در رسم پلات یک عنوان برای شناسایی پلات است

در ایکس لیبل و وای لیبل عنوان برای طول و عرض می گذاریم

و با تابع تایتل نامی برای کل نمودار می گذاریم

تابع لجند با تابع شو هر دو نمودار را نمایش می دهند

کد زیر خروجی یکسانی با کد بالا دارد

Fig, ax = plt.subplots(figsize= (4, 2))

Ax.plot(x, x, label= ‘Linear’)

Ax.plot(x, x\*\*2, label= ‘Quadratic’)

Ax.set(xlabel= ’X’, ylabel= ‘Y’, title= ‘Plot’)

Ax.legend()

Color= ‘Green’

Marker= ‘D’

Markercolor= ‘red’

آرگومان های دیگری نیز وجود دارد برای تنظیم در تابع رسم پلات

کالر می توان رنگ خط رو تنظیم کرد

مارکر می توان نوع نشانه در نقاط وصل را مشاهده کرد

مارکر کالر نیر می توان رنگ مارکر را مشخص کرد

Plt.annotate(‘Local Max’, xy= (1, 1), xytext= (1.5, 1.5), arrowprops= dict(facecolor= ‘g’))

Plt.ylim(-2, 2)

زمانی که می خواهی به یک جای خاص از نمودار اشاره کنیم از فلش استفاده می کنیم. فلش را از تابع انوتیت میاوریم

ابتدا نامی برای ان مشخص می کنیم سپس در ایکس وای جایی که می خواهیم فلش اشاره کند را می نویسیم

در ایکس وای موقعیت فلش و در ایکس وای تکست موقعیت نام فلش را مینویسیم

در پارامتر دیگر در قالب دیکشنری موارد دیگر مانند رنگ را می دهیم

scatterنمودار نقطه ای

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

Plt.scatter(x, x\*2)

Plt.show()

در اینجا مانند پلات نمودار رسم می کند اما دیگر نقاط را به هم متصل نمی کند

X = np.array([10, 15, 20, 30])

Plt.scatter(x, x\*2, color= ‘red’, marker= ‘\*’, s= 200)

Plt.show()

پارامتر های تنظیمی اول برای رنگ نقاط و مارکر نیز نوع نقطه که استار قرار داده ایم اس هم برای سایز در نظر گرفته می شود

Barنمودار میله ای

X = [‘A’, ‘B’, ‘C’, ‘D’, ‘E’]

Y = [13, 20, 8, 35, 16]

Plt.bar(x, y)

این نمودار نیز میله ای عما می کند ابتدا لیستی از عناوین و سپس لیستی از مقادیر به آن دادیم

X = [‘A’, ‘B’, ‘C’, ‘D’, ‘E’]

Y = [13, 20, 8, 35, 16]

C = plt.cm.Dark2(range(5))

Plt.barh(x, y, color= c)

سی متغیری است که در آن از توابع نوشته شده در رنج پنج عدد رنگ مختلف گرفته ایم و در آرگومان کالر متغیر را قرار داده ایم

در کنار اسم تابع بار یک اچ آورده ایم که آن تابع دیگری است که میله ها را در کنار تنظیم می کند

pieنمودار دایره ای

Population = [1400000000, 333000000, 60000000, 85000000]

Plt.pie(population)

نمودار دایره ای نسبت مقادیر را به هم نشان میدهد. در قالب دایره و صدم دایره

Population = [1400000000, 333000000, 60000000, 85000000]

Labels = [‘China’, ‘USA’, ‘Italy’, ‘Iran’]

P = plt.pie(population, autopct= ‘%1.1f%%’, explode=[0, 0, 0, 0.3])

Plt.legend(p[0], labels)

نمودار را داخل یک متغیر قرار دادیم و در پارامتر دوم آن به منظور اینکه برای قسمت ها درصد بزند تنظیم شد و در پارامتر آخر تنظیم شد که مقدار اخری به مقداری بیرون بیاید

هیستوگرام