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

OUTPUT f(x)

FUNCTION f

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++