Abedias

1

Age = 20

Price = 19.95

Fi

Filter( \_\_\_function, \_\_\_ ... builtins

Gerada

age = 20

price = 19.95

f

(- -

H 1}

\_Lada

2

Age = 20

Price = 19.95

Fisrt\_name=”Mosh”

Is\_online=False

Gerada

age = 20

price = 19.95

first\_name = "Mosh"

is\_online = Fhlig

5

Def finally\_instead\_of contexto\_manager (host, port):

With socket.socket(socket.AF\_INET , socket.SOCK\_STREAM) as s:

s.connect((host,port))

s.sendall(b’Hello, world’)

gerada

def f1nully\_1nstond-of\_contoxt.lanagor(host. port):

with socket.socket(socket.AF\_INET, sockot.socx\_STREAn) as s:

s.connect((nost, port)) .

s.sendellgb'uello, world'g

6

def bare\_esxcept():

while True:

try :

s=input(“Input a number: ”)

x=int(s)

break

except: # oops! Can’t CTRL-C to exit

print(“Not a number , try again”)

gerada

a" boro\_oxcopt():

will. Inn:

try:

5 l input('1nput a nunber: ")

x - mm

brook

O - a oopu can't cTRL-c to exit

print('No( a nulbcr, try again")

7

Def caret\_and\_exponentiaion(x,p):

Y=x^p # bitwise xor

Y=x\*\*p # exponentiaion

gerada

def caret\_and\_exponentiation(x, p):

9 y = xlp k bitwise xor

y = x \*\* p x exponentiation

8

Def append (n,l[):

l.append(n)

return l

l1=append(0) # [0

l2= apeend(1) # [0,1] oops

gerada

def append(n, -):

1. append(n)

return 1

11 = append(0) : : [0]

12 = append(1) x [0, 1] oops

9

Def append (n,l==none):

If l is None :

l=[]

l.append(n)

return l

l1=append(0) # [0

l2=append(1) # [1

gerada

hef append(n, 1=None):

if 1 is None:

1 = I] k

l.append(n)

return 1

11 = append(e) a [0]

12 = append(1) x [1]

10

def never\_using\_compehensioins():

squares= {}  
 for in range (10):

squares[i] =i\*i

gerada

def never\_using\_conprehensions()1

squares = {}

for i in range(10):

squares[i] = i \* i

11

def never\_using\_compehensioins():

dict \_comp={ i: i \* i for in range(10)  
 list\_comp={ x \* x for x in range(10)

set\_comp={ i% 3 for i in range(10)

gen\_comp={ 2\*x+5 for x in range(10)

gerada

def never\_using\_conprehensions() :l

dict\_comp = {i: i \* i for i in range(1e)}

list\_comp = [x\*x for x in range(1e)]

set\_comp = {1963 for i in range(19)}

gen\_comp = (2\*x+5 for x in range(19))

12

def always\_using\_compehensinos(a,b,n):

“”” matrix product of a, b of length n x n”””

Return[

Sum(a[n\* i+k ] \* b[n\*k+j ] for k in range(n))

For i in range(n)

For j in range(n)

]

gerada

def always\_using\_conprehensions(a, b, n)1

"""matrix product of a, b of length n x n"""

return [

sum(a[n \* i + k] \* b[n \* k + j] for k in r

for i in range(n)

for j in range(n)

13

def always\_using\_compehensinos(a,b,n):

“”” matrix product of a, b of length n x n”””

C=[]

For i in range(n):

For j in range(n):

Ij\_entry =sum(a[n\* i+k ] \* b[n \*k + j ] for k in range(n)):

c.append(ij\_entry)

return c

gerada

def always\_using\_conprehensions(a, b, n):

"""matrix product of a, b of length n x n"""

c = [l

for i in range(n):

for j in range(n):

ij\_entry = sun(a[n \* i + k] \* b[n \* k + j] for k in range(n))

c.append(ij\_entry)

0 return d

14

def checking\_type\_equality():

Point = namedtuple(‘ Point ‘ , ´[‘X’ ,’Y’ ] )

P=point(1,2)

If type(p) ==tuple :

Print (“ it’s a tuple”)

Else:

Print(“ it’s not a tuple”)

Gerada

def checking\_type\_equality():

Point = namedtup1e('Point', ['x', 'v'])

p = Point(1, 2)

ifq

print("it's a tu le")

else:

print("it's not a tuple")

15

def checking\_type\_equality():

Point = namedtuple(‘ Point ‘ , ´[‘X’ ,’Y’ fecha )

P=point(1,2)

If is instance(p,tuple): # Liskov

Print (“ it’s a tuple”)

Else:

Print(“ it’s not a tuple”)

Gerada

def checking\_type\_equa11ty():

Point = namedtup1e('Point', ['x', 'y'])

p = Point(1, 2)

if -(p, tuple): it Liskov :)

print("it's a tuple")

else:

print("it's not a tuple")

16

Def equality\_for\_singleton(x):

I f x is None:

pass

If x is True:

pass

If x is False:

Pass

gerada

def equality\_for\_singletons(x):

0 if x . None: it

pass

if x is True:

pass

if x is False:

pass

17

Def checking\_bool\_or\_len(x):

If bool(x):

Pass

If len(x) !=0:

Pass

Gerada

def checking\_bool\_or\_len(x)1

if bool(x):

pass

if len(x) l: 6:

pass

18

Def range\_len\_pattern():

A = [1,2,3]

For i in range(len(a)):

V=a[i]

...

Gerada

def range\_len\_pattern():

a= [1. 2, 3]

0 for i in \_a)):

v = a[i]

19

Def range\_len\_pattern():

A = [1,2,3]

For i, v in enumerate(a):

...

Gerada

def range\_len\_pattern():

a = [11 21 3]

for i, v in enumerate(a)1

20

Def range\_len\_pattern():

A = [1,2,3]

For i, v in enumerate(a):

...

Gerada

def range\_len\_pattern():

a = [11 21 3]

for i, v in enumerate(a)1

21

def for\_key\_in\_dict\_keys():

D={”a”:1 , “b”:2 , “c”:3}

For key in d.keys()

...

Gerada

def for\_key\_in\_dict\_keys():

d = {"a": 1, "b": 2, "c": 3}

for key in d-:

22

Def not\_using\_in\_dict\_items():

D={”a”:1 , “b”:2 , “c”:3}

For key in d[key]:

val =d[key]

...

gerada

def not\_using\_dict\_items():

d = {"a": 1, "b": 2, "c": 3}

for key in d: \*

val = d[key]

23

Def not\_using\_in\_dict\_items():

D={”a”:1 , “b”:2 , “c”:3}

For key in d:

val =d[key]

...

Gerada

def not\_using\_dict\_items():

d = {"a": 1, "b": 2, "c": 3}

for key in d:

' \_ .

val =d[key]

...