201921540

1 -

Css Selector

Css Selector . css selector (F12 > Select an Element). selector .

In [2]:

```
#today_main_news > div.hdline_news > ul > li:nth-child(1)
```

selector :nth-child , .

> **Q: Selector , split join selector .**

In [24]:

```
selector = "#today_main_news > div.hdline_news > ul > li:nth-child(1)"
## CODE
## '#today_main_news > div.hdline_news > ul > li'
```

HINT

1. (seperator) .

2. 1 .

3. 2 .

4. .

In [6]:

```
selector = "#today_main_news > div.hdline_news > ul > li:nth-child(1)"
selector_list = selector.split(">")
# >
selector_list[-1] = selector_list[-1].split(":")[0]
# li

" > ".join(selector_list)
# > .
```

Out[6]:

'#today_main_news > div.hdline_news > ul > li'

2. list comprehension

PR5 3 list comprehension .

Q: list comprehension gugu_com 7 .

In [9]:

```
# gugu_com(x=2)

# 2 x 1 = 2
# 2 x 2 = 4
# 2 x 3 = 6
# 2 x 4 = 8
# 2 x 5 = 10
# 2 x 6 = 12
# 2 x 7 = 14
# 2 x 8 = 16
# 2 x 9 = 18
```

In [16]:

```
def gugu_com(x):
    [print(f"{x} x {i} = {x*i}") for i in range(1, 10)]
# list comprehension
gugu_com(7)
# x   7
```

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

3.

.

> Q : list comprehension , .

In [41]:

```
## CODE

## [[1, 2, 3, 4, 5, 6],
##  [2, 4, 6, 8, 10, 12],
##  [3, 6, 9, 12, 15, 18],
##  [4, 8, 12, 16, 20, 24],
##  [5, 10, 15, 20, 25, 30],
##  [6, 12, 18, 24, 30, 36]]
```

HINT

1. .

die = [i for i in range(1,7)]

In [43]:

```
die = [i for i in range(1,7)]
# FOR

[[j*i for i in die] for j in die]
#
```

Out[43]:

```
[[1, 2, 3, 4, 5, 6],
 [2, 4, 6, 8, 10, 12],
 [3, 6, 9, 12, 15, 18],
 [4, 8, 12, 16, 20, 24],
 [5, 10, 15, 20, 25, 30],
 [6, 12, 18, 24, 30, 36]]
```

4.

2 . 2 36 .

> **Q: 6 x 6 2 , 2 + 6 2 . (2 .)**

In [45]:

```
dice_sum = [[2, 3, 4, 5, 6, 7],
            [3, 4, 5, 6, 7, 8],
            [4, 5, 6, 7, 8, 9],
            [5, 6, 7, 8, 9, 10],
            [6, 7, 8, 9, 10, 11],
            [7, 8, 9, 10, 11, 12]]

print(dice_sum[1][5])
#      7, 8,9,10,11,12    8
print(dice_sum[5][1])
# 2    3,4,5,6,7,8   8
```

```
8
8
```

In [46]:

```
die = [i for i in range(1,7)]

dice_sum = [[j+i for i in die] for j in die]

print(dice_sum[1][5])
print(dice_sum[5][1])
#
```

```
8
8
```

07 collections

Python .

, , .

In [47]:

```
from collections import defaultdict, Counter

text = """Python is a very simple programming language so even if you are new to programming, you can learn python without facing any issues."""

text2 = """C is a very difficult programming language so even if you are good at programming, you can learn c with facing any issues."""

text3 = """R Programming is good at statistical analysis. you can learn easily"""
```

1

> Q: defaultdict text word_counter .

HINT

1. collections defaultdict dict , key 0 .

In [52]:

```
# word_dict = dict()
# word_dict["key"]

## KeyError

# word_dict = defaultdict(lambda: 0)
# word_dict["key"]
## 0

# word_dict["key"] += 1
# word_dict["key"]
## 1
```

, .

- split
- lower

In [53]:

```
def word_counter(text):
    word_count = defaultdict(lambda: 0)
    for word in text.lower().split():
        word_count[word] += 1

    return word_count
```

In [54]:

```
word_counter(text)
```

Out[54]:

```
defaultdict(<function __main__.word_counter.<locals>.<lambda>()>,
            {'python': 2,
             'is': 1,
             'a': 1,
             'very': 1,
             'simple': 1,
             'programming': 1,
             'language': 1,
             'so': 1,
             'even': 1,
             'if': 1,
             'you': 2,
             'are': 1,
             'new': 1,
             'to': 1,
             'programming,': 1,
             'can': 1,
             'learn': 1,
             'without': 1,
             'facing': 1,
             'any': 1,
             'issues.': 1})
```

2

> Q: 1 word_counter text text2 text text3 .

HINT collections Counter dict Counter , defaultdict Counter . Counter({"a": 1, "b": 2, "c": 3}) - Counter({"a": 1, "b": 1, "c": 1})

Counter({'b': 1, 'c': 2}) dictionary .values() value . sum(Counter({"a": 1, "b": 2, "c": 3}).values()) #

6

1. Counter(A) Counter(B) .(Counter dict value)

In [56]:

```
def text_similarity(text_count_1, text_count_2):
    text1_count = Counter(text_count_1)
    text2_count = Counter(text_count_2)

    word_total = sum(text1_count.values())
    word_diff = sum((text1_count - text2_count).values())

    return (1 - word_diff / word_total) * 100
```

In [57]:

```
text_similarity(word_counter(text), word_counter(text2))
#   word_counter        text
```

Out[57]:

73.91304347826086

In [59]:

```
text_similarity(word_counter(text), word_counter(text3))
#   word_counter        text
```

Out[59]:

21.739130434782606

2 -

--

1.

- (lambda) , . , . ' ' . .

1.1.

In [61]:

```
def f(x,y):
    return x + y

print(f(1,4))
# 1   4
```

5

1.2. lambda

In [62]:

```
f=lambda x,y: x + y
print(f(1,4))
# 1    4
```

5

1.3. lambda

In [63]:

```
print((lambda x, y:x + y)(1, 4))
#      1    4
```

5

2.

2.1. map

. . .

In [66]:

```
ex = [1,2,3,4,5]
f = lambda x:x**2
print(list(map(f, ex)))
#
```

[1, 4, 9, 16, 25]

-
- ex , f . 'map(,)' map(f,ex) . f ex .
- 2.x 3.x 3.x map() list() .
- (generator) , .
- list , . , .

In [67]:

```
ex=[1,2,3,4,5]
f=lambda x:x**2
for value in map(f,ex):
    print(value)
#       map
```

1
4
9
16
25

- map() . . , .

In [69]:

```
ex = [1, 2, 3, 4, 5]
[x**2 for x in ex]
#
```

Out[69]:

[1, 4, 9, 16, 25]

- map() 2 .

In [73]:

```
ex=[1,2,3,4,5]
f=lambda x,y:x+y
list(map(f,ex,ex))
# 1+1 2+2 3+3
```

Out[73]:

[2, 4, 6, 8, 10]

In [76]:

```
[x+y for x,y in zip(ex,ex)] #
```

Out[76]:

[2, 4, 6, 8, 10]

2.2. reduce

- map() .
- lambda . .

In [79]:

```
from functools import reduce
print(reduce(lambda x,y:x+y, [1,2,3,4,5]))
#
```

15

-

In [81]:

```
x=0
for y in [1,2,3,4,5]:
    x += y
print(x)
#           1   5
```

15

3.

3.1.

-

In [83]:

```
def asterisk_test(a, *args):
    print(a,args)
    print(type(args))

asterisk_test(1,2,3,4,5,6)

#       tuple
```

1 (2, 3, 4, 5, 6)
<class 'tuple'>

-

In [85]:

```
def asterisk_test(a,**kargs):
    print(a,kargs)
    print(type(kargs))
asterisk_test(1,b=2,c=3,d=4,e=5,f=6)
#
```

1 {'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
<class 'dict'>

3.2.

-

In [87]:

```
def asterisk_test(a,args):
    print(a,*args)
    print(type(args))
asterisk_test(1,(2,3,4,5,6))
#
```

```
1 2 3 4 5 6
<class 'tuple'>
```

In [88]:

```
def asterisk_test(a,args):
    print(a,args)
    print(type(args))
asterisk_test(1,(2,3,4,5,6))
#
```

```
1 (2, 3, 4, 5, 6)
<class 'tuple'>
```

-

In [91]:

```
a,b,c=([1,2], [3,4], [5,6])
print(a,b,c)
data=([1,2], [3,4], [5,6])
print(*data)
# 2
```

```
[1, 2] [3, 4] [5, 6]
[1, 2] [3, 4] [5, 6]
```

- zip

In [92]:

```
for data in zip(*[[1,2],[3,4],[5,6]]):
    print(data)
    print(type(data))

#           1,3,5 / 2,4,6
```

```
(1, 3, 5)
<class 'tuple'>
(2, 4, 6)
<class 'tuple'>
```

-

In [93]:

```
def asterisk_test(a,b,c,d):
    print(a,b,c,d)
data={"b":1, "c":2, "d":3}
asterisk_test(10, **data)
```

```
10 1 2 3
```

4.

4.1.

In [95]:

```
vector_a=[1,2,10]  #
vector_b=(1,2,10)   #
vector_c={'x':1, 'y':2, 'z':10}  #
```

- :

In [97]:

```
u=[2,2]
v=[2,3]
z=[3,5]
result=[]

for i in range(len(u)):
    result.append(u[i]+v[i]+z[i])
print(result)
```

[7, 10]

In [98]:

```
u=[2,2]
v=[2,3]
z=[3,5]
result=[sum(t) for t in zip(u,v,z)]
print(result)
```

[7, 10]

-

In [100]:

```
def vector_addition(*args):
    return [sum(t) for t in zip(*args)]   # unpacking    zip(u,v,z)          .

vector_addition(u,v,z)
```

Out[100]:

[7, 10]

-

In [101]:

```
a = [1, 1]
b = [2, 2]

[x + y for x, y in zip(a, b)]
```

Out[101]:

[3, 3]

- :

In [103]:

```
u=[1,2,3]
v=[4,4,4]

alpha=2

result=[alpha*sum(t) for t in zip(u,v)]
result
```

Out[103]:

[10, 12, 14]

4.2.

In [104]:

```
matrix_a=[[3,6], [4,5]] #
matrix_b=[(3,6), (4,5)] #
matrix_c={(0,0):3, (0,1):6, (1,0):4, (1,1):5}  #
```

- : elemnet-wise

In [106]:

```
matrix_a=[[3,6], [4,5]]
matrix_b=[[5,8], [6,7]]

result=[[sum(row) for row in zip(*t)] for t in zip(matrix_a, matrix_b)]
print(result)
```

[[8, 14], [10, 12]]

with map

PR6 split . .

> pins = ["891120-1234567", "931120-2335567", "911120-1234234", "951120-1234567"]
>
> Q: lambda map .

CODE

['1', '2', '1', '1']

HINT

1. lambda .
2. map lambda .
3. 2.1. map lambda list .

In [109]:

```
pins = ["891120-1234567", "931120-2335567", "911120-1234234", "951120-1234567"]


list(map(lambda x: x.split("-")[1][0], pins))
# -
```

Out[109]:

['1', '2', '1', '1']

.

v1 * w1+⋯+vn * wn

> Q: (list) dot , a=[1, 2], b=[3,4] .

HINT

1. 4.1. . .

In [111]:

```
a = [1, 2]
b = [3, 4]

dot = lambda a,b : sum([x*y for x, y in zip(a, b)])
dot(a,b)

# 3+8 = 11
```

Out[111]:

11