ZIP AND ENUMERATE

ZIP

- **zip** can combine elements from two or more lists into nested tuples.

ENUMERATE

- **enumerate** can generate an integer number next to each output.</font></b>

BOTH CAN BE EASILY REPLACED

In [7]:

```
for count,i in enumerate(range(10, 40, 5), 200):
    print(count, i)
```

```
200 10
201 15
202 20
203 25
204 30
205 35
```

In [9]:

```
for i in range(10, 40, 5):
    print(190+i, i)
```

```
200 10
205 15
210 20
215 25
220 30
225 35
```

In [15]:

```
tup1 = (33, 49, 55)
mylist = [10, 20, 30]
z = list(zip(tup1, mylist))
z
```

Out[15]:

```
[(33, 10), (49, 20), (55, 30)]
```

In [18]:

```
for t, m in z:
    print(t + m)
```

```
43
69
```

```
85
```

In [19]:

```python
for i in range(3):
    print(tup1[i] + mylist[i])
```

```
43
69
85
```

In [20]:

```python
groceries = ["apples", "chips", "bread", "icecream"]
prices = [2, 3, 1.2, 4.25]

for food in range(4):
    print(groceries[food], "=" , prices[food])
```

```
apples = 2
chips = 3
bread = 1.2
icecream = 4.25
```

In [23]:

```python
zip_shop = (zip(groceries, prices))
zip_shop

for g, p in zip_shop:
    print(g, "=", p)
```

```
apples = 2
chips = 3
bread = 1.2
icecream = 4.25
```

In [24]:

```python
num1 = [100, 2, 90, 10] ; num2 = [12, 7, 90, 50]

zip_num = zip(num1, num2)

for i, j in zip_num:
    if i > j:
        print(i)
    elif i < j:
        print(j)
    else:
        print(i, j)
```

```
100
7
90 90
50
```

In [25]:

```python
for i in range(4):
    if num1[i] > num2[i]:
        print(num1[i])
    elif num1[i] < num2[i]:
```

```
    elif num1[i] < num2[i]:
        print(num2[i])
    else:
        print(num1[i], num2[i])
```

```
100
7
90 90
50
```

In [43]:

```
numbers = list(range(10,70, 10))
numbers
values = [2, 5]
import itertools
from itertools import cycle as cy

val1 = [2] ; val2 = [5]



zip_val = list(zip(numbers, cy(val1), cy(val2)))
calc = []

for n, v, z in zip_val:
    s = n *v
    t = n * z
    calc.append(s)
    calc.append(t)
calc
# len(calc)
```

Out[43]:

```
[20, 50, 40, 100, 60, 150, 80, 200, 100, 250, 120, 300]
```

In [44]:

```
zip_val
```

Out[44]:

```
[(10, 2, 5), (20, 2, 5), (30, 2, 5), (40, 2, 5), (50, 2, 5), (60, 2, 5)]
```

In [35]:

```
numbers
```

Out[35]:

```
[10, 20, 30, 40, 50, 60]
```