Break Pass and Continue

```
# Continue

prime = [2, 3, 5, 7, 11, 13, 17, 19]
for i in range(0, 20):
    if i in prime:
        continue
        print(i)
    else:
        print(i, end=" ")
```

```
0 1 4 6 8 9 10 12 14 15 16 18
```

```
# Break

while True:
    N = input()
    if N == "stop":
        break
```

```
 Rahul
 stop
```

```
# pass

def payment():
    pass
```

Statically typed vs Dynamically typed Language

```
type(23)
```

```
int
```

```
type("Rahul")
```

```
str
```

```
x = 23

type(x)
```

```
int
```

Mutability in Different Data types Revisited

```
# integer

# immutable

x = 23
id(x)
```

```
140566897683440
```

```
x = 24
id(x)
```

```
140566897683472
```

```
# Integers are immutable

l = [1, 2, 3]
id(l)
```

```
140566634710336
```

```
l
```

```
[1, 2, 3]
```

```
l.append(4)
print(id(l))
print(l)
```

```
140566634710336
[1, 2, 3, 4]
```

```
# Lists are mutable



# float

# immutable

a = 2.5
print(id(a))
a = 2.52
print(id(a))
```

```
140567173970160
140567173970128
```

```


# Strings

# Immutable
```

Lists Revisited

```
# What are lists Data types or Data Structure?

# list, tuple, set, dictionary

runs = [1, 2, 3, 4, 5]

type(runs)
```

```
list
```

```
# Indexing in a list

# Mutability

runs[0]
```

```
1
```

```
runs[1]
```

```
2
```

```
runs[-1]
```

```
5
```

```
runs
```

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

```
len(runs)
```

```
5
```

```
runs[-5]
```

```
1
```

```
runs[0] == runs[-len(runs)]
```

```
True
```

```
# Corresponding to a positive index there is always a negative index
as well


# Lists are mutable type of data
```

```
# Slicing

marks = [10,20,50,33, 100]
print(marks[::-1])
```

```
[100, 33, 50, 20, 10]
```

```
# list[start:end:jump]

a = [1,2,3,4,5,6,7,8,9,10]

a[3:8:-1]
```

```
[]
```

```
a = [1,2,3,4,5,6,7,8,9,10]

a[8:3:-1]
```

```
[9, 8, 7, 6, 5]
```

```
# [1,2,3,4,5,6,7,8,9,10]
# start = 4 -> included
# end = 9 -> excluded


# Step/size and jump
```

List methods

- append
- extend
- count
- index
- insert
- pop
- remove
- reverse
- sort
- Concatenation

```
# Max
# Min
# Sum

runs
```

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

```
max(runs)
```

```
5
min(runs)
1
sum(runs)
15

# append
runs
[1, 2, 3, 4, 5]
new = [1, 2, 3]
runs.append(12)
runs
[1, 2, 3, 4, 5, 12]
runs.append(new)
runs
[1, 2, 3, 4, 5, 12, [1, 2, 3]]

# extend
l = [1, 2, 3, 4, 5]
l.extend(new)
l
[1, 2, 3, 4, 5, 1, 2, 3]
m = max(l)
m
5
# count
l
[1, 2, 3, 4, 5, 1, 2, 3]
```

```
l.count(1)
```

```
2
```

```
# index
```

```
l
```

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

```
l.index(1)
```

```
0
```

```
l.index(12)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/zn/hkv6562d6_d30glfs8yc76900000gn/T/ipykernel_7341/340747418.py in <module>
----> 1 l.index(12)

ValueError: 12 is not in list
```

```
# Insert
```

```
l
```

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

```
l.insert(2, 34)
```

```
l
```

```
[1, 2, 34, 3, 4, 5, 1, 2, 3]
```

```
l.insert(2, new)
```

```
l
```

```
[1, 2, [1, 2, 3], 34, 3, 4, 5, 1, 2, 3]
```

```
# Deleting an object
# pop
# remove
```

```
# pop method by default removes the last element and returns it as
well

# Or pop can take index as parameter as well

l

[1, 2, [1, 2, 3], 34, 3, 4, 5, 1, 2, 3]

l.pop()

3

l.pop(0)

1

l

[2, [1, 2, 3], 34, 3, 4, 5, 1, 2]




# remove

# It needs a value not index
# It doesn't return the value

l

[2, [1, 2, 3], 34, 3, 4, 5, 1, 2]

l.remove(2)

l

[[1, 2, 3], 34, 3, 4, 5, 1, 2]


l

[[1, 2, 3], 34, 3, 4, 5, 1, 2]

del l[0]

l

[34, 3, 4, 5, 1, 2]

del l

# l
```

```
# reverse
l = [1, 2, 34, 3, 4, 5, 1, 2, 3]
l.reverse()
l
[3, 2, 1, 5, 4, 3, 34, 2, 1]

# sort
l
[3, 2, 1, 5, 4, 3, 34, 2, 1]
l.sort(reverse=True)
l.sort()
l
[1, 1, 2, 2, 3, 3, 4, 5, 34]

# List Concatenation
l
[1, 1, 2, 2, 3, 3, 4, 5, 34]
new
[1, 2, 3]
l1 = l + new
l1
[1, 1, 2, 2, 3, 3, 4, 5, 34, 1, 2, 3]
```

Iterating in a list

List comprehension

```
l = [1, 2, 3]

# make a new list with elements as square of this list?

new = []

for i in l:
    new.append(i*i)
print(new)
```

```
[1, 4, 9]
```

```
# [i*i for i in l]

new = [i * i for i in l]

# i = 1, 2, 3
# new = [1, 4, 9]

new
```

```
[1, 4, 9]
```

```
# Use of if else in List comprehension

# create list with even numbers from 0 to 10

new = []

for i in range(0, 11):
    if i % 2 == 0:
        new.append(i)
print(new)
```

```
[0, 2, 4, 6, 8, 10]
```

```
new = [i for i in range(0, 11) if i % 2 == 0]

new
```

```
[0, 2, 4, 6, 8, 10]
```

```
new = [i for i in range(0, 11, 2)]

new
```

```
[0, 2, 4, 6, 8, 10]
```

Nested Lists

```
# Printing a Continous 2D list

# n = 3
# o/p: [[1], [1, 2], [1, 2, 3]]

# 1
# 1 2
# 1 2 3

# There are 3 rows of length equal to row no
# We are printing column number

n = 3

for i in range(1, 4):
    for j in range(1, i+1):
        print(j, end=" ")
    print()
```

```
1
1 2
1 2 3
```

```
n = int(input())

final = []
for i in range(1, n+1):
    row = []
    for j in range(1, i+1):
        row.append(j)
    final.append(row)
```

```
 4
```

```
final
```

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

```
# HW: Do this question using list comprehension
```

Pair Sum

```
A = 21
B = [1, 2, 10, 5, 11, 4]

# n = len(B)
# for i in range(n):
#     print(i)
#     for j in range(n):
#         print(j)
```

```
def pair_sum(A, B):
    n = len(B)

    for i in range(n):
        for j in range(n):
            if B[i] + B[j] == A and i != j:
                return 1
    return 0

A = 31
B = [1, 2, 10, 5, 11, 4]

pair_sum(A, B)
```

```
0
```