Flow Control and Loops Cheat Sheet

if ... elif ... else Statements

Execute a specific block of code if a test condition is evaluated to True

Syntax

```
a, b = 3, 5
# if a is less that b execute the indented block of code under the if clause, otherwise go and test the elif condition

if a < b:
   print('a is less than b')
elif a == b:
   print('a is equal to b')
else:
   print('a is greater than b')
```

Boolean Variables

```
# True is 1 and False is 0
True == 1     # => True
bool(True)    # => 1

False == 0     # => True
bool(False)    # => 0

1 is True     # => False
0 is False    # => False

True > False                # => True
a = (True + True ) * 10    # => 20
```

```
id(True)    # => 10714848 (you'll get another value)
id(4 > 2)   # => 10714848  - the address of True and False is constant during program execution

# The next 2 expressions are equivalent
(4 > 2) == True     # => True
(4 > 2) is True     # => True
```

Truthiness of objects

```
bool(0)       # => False
bool(0.0)     # => False
bool(10)      # => True
bool(-1.5)    # => True

bool('')       # => False (empty string)
bool('py')    # => True

bool([])        # => False (empty list)
bool([1,2])    # => True

bool(())        # => False (empty tuple)
bool((3,4))    # => True

bool({})                          # => False (empty dictionary)
bool({1:'abc',2:55,'a':5})    # => True
```

```
b = 0
if b:   # it tests the truthiness of b or bool(b)
   print('The truthiness of b is True')
else:
   print('The truthiness of b is False')

my_str = 'some string'
if my_str:   # it tests the truthiness of my_str or bool(my_str)
   print('The truthiness of my_str is True')
else:
   print('The truthiness of my_str is False')

name = 'Andrei'

# Pythonic version
print('Hello Andrei') if name == 'Andrei' else print('You are not Andrei!')
```

```
# equivalent to:
if name == 'Andrei':
    print('Hello Andrei')
else:
    print('You are not Andrei')
```

Boolean Operators

```
# expression1 and expression2    => True when both expressions are True and False otherwise
# expression1 or expression2     => True when any expression is True

a, b = 3, 5
a < 10 and b < 10    # => True
a < 10 and b > 10    # => False

a < 10 or b < 10     # => True
a < 10 or b > 10     # => True

# The next 2 expressions are equivalent
2 < a < 6
a < 2 and a < 6    # more readable

a != 7 or b > 100         # => True
not a == a                # => False
a == 3 and not b == 7     # => True

not a > 10 and b < 10    # => True

a < 10 or b > 10          # => True
not a < 10 or b > 10      # => False
not (a < 10 or b > 10)    # => False

# !!
# Python considers 4 > 2 and 2 == True
4 > 2 == True      # => False
(4 > 2) == True    # => True
```

or / and operators

```
your_age = 14
#if ANY expression is True execute the indented block of code under the if clause
if your_age < 0 or your_age > 99:
    print('Invalid age!')
elif your_age <= 2:
    print('You are an infant')
elif your_age < 18:
    print('You are a child')
else:
    print('You are an adult')

a = 3
if 1 < a <= 9:
    print('a is greater than 1 and less than or equal to 9')

# equivalent to:
if a > 1 and a <= 9:
    print('a is greater than 1 and less than or equal to 9')

# The following 3 examples test if number a is divisible by 6
a = 12

# 1st example - nested if
if a % 2 == 0:
    if a % 3 ==0:
        print('Example 1: a is divisible by 2 and 3 (or by 6)')

# 2nd example - and operator. It returns True if both expressions are True, False otherwise
if a % 2 == 0 and a % 3 == 0:
    print('Example 2: a is divisible by 2 and 3 (or by 6)')

# 3rd example
if not (a % 2  and a % 3 ):
    print('Example 2: a is divisible by 2 and 3 (or by 6)')
```

For Loops

It iterates over a sequence and executes the code indented under the for clause for each element in the sequence

```
movies = ['Star Wars', 'The Godfather', 'Harry Potter ', 'Lord of the Rings']

for m in movies:
    print(f'One of my favorites movie is {m}')
else: #the code below gets executed when "for" has finished looping over the entire list
    print('This is the end of the list')
```

range()

```
for i in range(100):
   pass    # => empty instruction or "do nothing"

for i in range (10):  # => from 0 (default, included) to 10 excluded
   print(i, end=' ')
# it prints:  0 1 2 3 4 5 6 7 8 9

for i in range (3, 9):  # => from 3 included to 9 excluded
   print(i, end=' ')
# it prints: 3 4 5 6 7 8

for i in range (3, 20, 3):  # => from 3 included to 20 excluded in steps of 3
   print(i, end=' ')
# it prints: 3 6 9 12 15 18

for i in range (8, -4, -2):  # => from 8 included to -4 excluded in steps of -2
   print(i, end=' ')
# it prints: 8 6 4 2 0 -2
```

for and continue

```
# for ... continue -> it prints out all letters of the string without 'o'
for letter in 'Python Go and Java Cobol':
   if letter == 'o':
      continue    # go to the beginning of the for loop and do the next iteration
   print(letter, end='')
```

for and break

```
sequence = [1, 5, 19, 3, 31, 100, 55, 34]
for item in sequence:
   if item % 17 == 0:
      print('A number divisible by 17 was found! Breaking the loop...')
      break   # breaks out the loop
else: # belongs to for, not to if
   print('There is no number divisible by 17 in the sequence')

# it prints out the numbers from 0 to 4
for number in range(10):
    if number == 5:
        break
    print(number)

# it prints out the letters Pytho
for letter in 'Python':
    print(letter)
    if letter == 'o':
        break
```

While Loops

```
a = 10

# Infinite Loop, it prints out 10 indefinitely
while a:  # it tests the truthiness of a or bool(a) which is always True
    print(a)

# Printing out the numbers from 10 to 1
while a:    # => "while a:" is equivalent to "while a > 0:"
    print(a)
    a -= 1
else:   # => executes the block of code below after finishing the while loop (and if no "break"
statement was executed)
    print('Finishing printing numbers. a is now 0')

# Printing out only odd numbers between 1 and 20
a = 0
while a < 20:
    a += 1
    if a % 2 == 0:
        continue    # go the the beginning of the while loop
    print(f'Odd number {a}')   #it reaches this line only if the continue statement wasn't executed


# printing out numbers greater than 2
a = 7
while a > 0:
    a -= 1
    if a == 2:
        break   # => it breaks out the while loop and executes the next instruction after while
    print(a)

print('Loop ended.')
```