Programming in Python

1 Repeating Actions(posted) [...]

2 Conditions and Control Flow

You will often need to make a choice among options while programming. This means you may have to set conditions for deciding what the computer does given different options

2.1 if True: Output

The general format for working with **"if elif else"** in python is:

```
if condition #1:
    option #1
elif condition #2:
    option #2
    :
    :
    :
elif condition #(n-1):
    option #(n-1)
else:
    last option
```

Note: use **colon (:)** at the end of each conditional statement and use **indentation** before each option.

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

In [1]:

```
# Example 1: General form
a = 200
b = 100 # Try b=200, b=300

if a > b:
  print("'a' is greater than 'b'")

elif a == b:
  print("'a' and 'b' are equal")

else:
  print("'b' is greater than 'a'")
```

```
'a' is greater than 'b'
```

In [2]:

```
a = 200
b = 100 # Try b=200, b=300
a > b
```

Out[2]:

```
True
```

In [6]:

```
a = 200
b = 300
a > b
```

Out[6]:

```
False
```

In [ ]:

```
a = 200
b = 200
a > b
```

In [3]:

```
# Example 5: Using AND, Or
a = 10
b = 1
c = 5

if a > b or a > c:
  print("At least one of the conditions is Tr

if a > b and a > c:
  print("Both of the conditions are True")
```

```
At least one of the conditions
is True
Both of the conditions are Tru
e
```

In [5]:

```
a = 10
b = 1
c = 5 # Try different values

a > b or a > c
```

Out[5]:

```
True
```

In [7]:

```
a = 10
b = 1
c = 5 # Try different values
a > b and a > c
```

Out[7]:

```
True
```

2.1.1 Short_Hand If

One-Line if Statements

In [8]:

```
# Example 1:
a=10
b=2

if a > b: print("a is greater than b")
```

```
a is greater than b
```

In [9]:

```
# Example 2: excute many options in one line using ;
a=10
b=2

if a > b: print("a =",a,"and b=",b); print("a is greater than b")
```

```
a = 10 and b= 2
a is greater than b
```

In [11]:

```
# Example 3:
x = 2

if x == 1: print('One'); print('Two'); print('Three')
elif x == 2: print('Two'); print('Three'); print('Four')
else: print('Three'); print('Four'); print('Five')
```

```
Two
Three
Four
```

In [12]:

```
# C.W. 1:
# Given: x=1 or 2 or 3 or ...
# Requirment: Test if x is odd or even
```

```
'x' is an even number
```

In [14]:

```
# C.W. 2:  Use input function to inter a number
# Requird: let your code decide if user input is an even or an odd number
```

```
Please enter an integer then press Enter
3
You just entered an odd number =  3
```

In [16]:

```
# C.W. 2: Using "loop" anf "if" together
# Given: nums = [1, 2, 3, 4, 5]
# Requird: Find Even & odd numbers in the "nums" list
```

```
Num 1 is odd
Num 2 is even
Num 3 is odd
Num 4 is even
Num 5 is odd
```

In [18]:

```
# H.W. 1:
# Given: A heterogeneous list (myList) consists of: strings, integers, and floats
# myList = [1, 5, -2,"Lam", 5.25, 1.245, 0.5,"Ram", -3, 0, 10, 2, -1, 0, 1, 1.25, -9.3, 3,

# Requird: Create a sorted lists (subsets) without any element repeated:
#          1- integer list
#          2- float list
#          2- string list
```

```
Int_List = {0, 1, 2, 3, 5, 6, 10, -1, -3, -2}
Float_List = {0.5, 1.245, 1.25, 5.25, -9.3}
String_List = {'Sam', 'Ram', 'Lam'}
```

2.1.2 Nested If

In [21]:

```
# Example 1:
x =15

if x > 10:
  print("x is above ten,")

  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
else:
   print("x is less tham ten,")
```

```
x is above ten,
but not above 20.
```

In [22]:

```
# H.W. 1: Use "loop" anf "Nested-if" together

# Given:I have a heterogeneous list (myList): integers, and floats
# myList = [1, 5, -2, 5.25, 1.245, 0.5, -3, 0, 10, 2, -1, 0, 1, 1.25, -9.3, 3, 5, 6]

# Requird: I want to create a sorted lists (subsets) without any element repeated:
#    1- integer list (only +ve and even numbers)
#    2- float list
```

```
intList = {0, 10, 2, 6}
floatList = {0.5, 1.245, 1.25, 5.25, -9.3}
```

2.2 Output if True

Creating a new value that meet certain conditions, like function.

In [26]:

```
# Example 2:
a=20
b=10

a if a > b else b
```

Out[26]:

```
20
```

In [32]:

```
# Boolean results: True or False
a=20
b=10 # Try different values

a > b
```

Out[32]:

```
True
```

In [33]:

```
# Example 1:
a=20
b=10

a if True else b   # Try the next line
# a if False else b
```

Out[33]:

```
20
```

In [34]:

```
# Example 3: Assigning the output to a variable
a=10
b=20

m = a if a > b else b
m
```

Out[34]:

```
20
```

2.2.1 Ternary Operators

In [36]:

```
# Example 1:
a = 200
b = 100 # Try different values

"A > B" if a > b else "A < B" if a < b else print("A = B")
```

Out[36]:

```
'A > B'
```

In [37]:

```
# Example 2:
a = 200
b = 100

print("a > b") if a > b else print("a = b") if a == b else print("a < b")
```

```
a > b
```

In [47]:

```
# Example 2:
a = 500
b = 400

a-b if a > b else 0 if a == b else (b-a)*-1
```

Out[47]:

```
100
```

In [53]:

```
a = 500
b = 400

a > b and a > 0 # and/or also can be used
```

Out[53]:

```
True
```

2.3 True & False expressions in Python

2.3.1 Item in a list

In [48]:

```
# Example 1: if True: Output
A_list=['foo', 'bar', 'baz']

if 'foo' in A_list:
    print('Yes {foo} is in the list')
```

```
Yes {foo} is in the list
```

In [49]:

```
'foo' in ['foo', 'bar', 'baz']
```

Out[49]:

```
True
```

In [51]:

```
# Example 2: Output if True
A_list=['foo', 'bar', 'baz']

"Yes"  if 'foo' in A_list else "no"
```

Out[51]:

```
'Yes'
```

In [58]:

```
# Example 3: all in one line
c='qux'

f'Yes {c} is in the list' if c in ['foo', 'bar', 'baz'] else f'{c} is not in the list'
```

Out[58]:

```
'qux is not in the list'
```

2.3.2 Not

In [60]:

```
# Example 2:
raining =True # try False
'library' if raining else 'beach'
```

Out[60]:

```
'library'
```

In [62]:

```
# Example 2: using not True or not False
raining = True

'beach' if not raining else 'library'
```

Out[62]:

```
'library'
```

In [63]:

```
raining = True
not raining
```

Out[63]:

```
False
```

2.3.3 is methods

In [64]:

```
# Example 1:
'Upper' if "ABC".isupper( ) else 'Lower' # Tr
```

Out[64]:

```
'Upper'
```

In [65]:

```
"ABC".isupper( ) # You can use the other meth
```

Out[65]:

```
True
```

2.3.4 is vs. ==

In [66]:

```
list_1 = [1, 2, 3]
list_2 = list_1

list_3 = [1, 2, 3]
list_4 = list_1.copy()

print(list_1)
print(list_2)
print(list_3)
print(list_4)
```

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

2.4 Break, Continue & Pass

Note: they control loops

In [67]:

```
print(list_1 == list_2)
print(list_1 == list_3)
print(list_1 == list_4)
```

```
True
True
True
```

In [68]:

```
print(list_1 is list_2)
print(list_1 is list_3)
print(list_1 is list_4)
```

```
True
False
False
```

2.4.1 Break

In [69]:

```
import time

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

for num in nums:
    print(num)
    time.sleep(1) # Only for illustration
    if num == 3:
        break
```

```
1
2
3
```

2.4.2 Continue

In [70]:

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

for num in nums:
    if num == 3:
        continue
    print(num)
    time.sleep(1) # Only for illustration
```

```
1
2
4
5
```

2.4.3 Pass

In [71]:

```
nums = [1, 2, 3, 4, 5]
for num in nums:
    if num == 3:
        pass     # If i need to get back to this line and put codes
    print(num)
```

```
1
2
3
4
5
```

In [72]:

```
# H.W: Password guess
# Required: Enter a password and test it if true or fales, max Tries =4
# Hint: use input, while, if, break
```

```
Enter password (0 to 9), you have only 4 tries
Input password:6
------------------------------
Congrats --- Your password is right
```

3 Comprehension (next) [...]

4 Functions: Reusable block of codes [...]

5 Extra libraries (Preparing stage) [...]