Conditions and Loops

If Statement

- condition operators can be used in several ways, most commonly in "if statements" and loops.

```
if condition:
    action
```

```
1  x=4
2  y=2
3
4  if x>y:
5      x+=1
6      result="x has a new value is : {}"
7      print(result.format(x))
```

- Remember!
  Indentations define the scope of code
  If statement, without indentation (will raise an error).

Short Hand If

- Used when you have only one statement to execute.

```
1  x=4
2  y=2
3
4  if x>y: print (" x is greater than y")
5
```

Elif

- if the previous conditions were not true, then try this condition.

```
1  x=2
2  y=2
3
4  if x>y:
5      print (" x is greater than y")
6
7  elif x==y:
8      print("x and y are equal")
```

Else

- The else keyword catches anything which isn't caught by the preceding conditions.

```
1  x=2
2  y=4
3
4  if x>y:
5      print (" x is greater than y")
6
7  elif x==y:
8      print("x and y are equal")
9
10 else:
11     print("y is greater than x")
12
```

Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line.

```
a = 2
b = 330
print("A") if a > b else print("B")
```

pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
if b > a:
  pass
```

Loops

- python has two type of loop:
    1. while loop
    2. for loop

While Loop

```
1  i = 1
2  while i < 6:
3    print(i)
4    i += 1
5
```

break Statement

With the break statement we can stop the loop even if the while condition is true.

```
1  i = 1
2  while i < 6:
3    print(i)
4    if i == 3:
5      break
6    i += 1
```

continue Statement

With the continue statement we can stop the current iteration, and continue with the next

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)

# Note that number 3 is missing in the result
```

```
1
2
4
5
6
```

else Statement

With the else statement we can run a block of code once when the condition is false.

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

For loops

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
1  fruits = ["apple", "banana", "cherry"]
2  for x in fruits:
3    print(x)
```

Nested Loops

- A nested loop is a loop inside a loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop"

```
color = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in color:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Range() Function

- To loop through a set of code a specified number of times, we can use the range() function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

- NOTE : in for loop we also use break , continue , else , and pass.

- it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6).

```
for x in range(2, 6):
  print(x)
```

- it is possible to specify the increment value by adding a third parameter.

```
for x in range(2, 30, 3):
  print(x)
```

Problems

- Create a program that asks the user for input 2 different information for 2 person “name & age ” ,compare it and display who is the oldest in an appropriate sentence.

  (*hint: you should indicate if they are the same age if it happen*)

- Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. *Hint: how does an even / odd number react differently when divided by 2?*
- Extras:

  - If the number is a multiple of 4, print out a different message.

  - Ask the user for two numbers: one number to check (call it num) and one number to divide by (check). If check divides evenly into num, tell that to the user. If not, print a different appropriate message.

- Take a list, say for example this one:

  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

  and write a program that prints out all the elements of the list that are less than 5.

- Create a program that asks the user for a number and then prints out a list of all the divisors of that number. (If you don't know what a *divisor* is, it is a number that divides evenly into another number. For example, 13 is a divisor of 26 because 26 / 13 have no remainder.)

Any Question…?