Control statements

Ch-6

contents

- if
- else
- elif
- while
- for
- break
- continue
- pass
- assert
- return

if statement

- if-else-elif.ipynb

- Notice word indentation
  - mandatory

If..else statement

- if-else-elif.ipynb

- Notice word indentation
  - mandatory

If..elif..else statement

- if-else-elif.ipynb

- Notice word indentation
  - mandatory

while loop

- Syntax:

```
while condition:
    Statements
```

- while-loop.ipynb

for loop

- Syntax:

```
for somevar in sequence/range:
    loop statement(s)
```

break statement

- Used in the while and for loops
- When ‘break’ is executed,
  - the Python interpreter jumps out of the loop to process the next statement.

- break-return-assert-pass.ipynb

continue statement

- Used in the while and for loops
- When ‘continue’ is executed,
  - the Python interpreter skips the remaining statements incurrent loop iteration.

- break-return-assert-pass.ipynb

Pass statement

- The ‘pass’ statement
  - doesn’t do anything.
  - Used as a place holder.
- Used inside ‘if’ statement or inside a loop
  - to represent no operation.
- We use it when
  - we need a statement syntactically but
    - we do not want to do any operation.

- break-return-assert-pass.ipynb

assert statement

- The ‘assert’ statement is
  - useful to check if a particular condition is fulfilled or not.
- Syntax:
  - `assert expression, message`
  - `message` is not compulsory
- E.g.
  - If we want to assure that the user must enter only a number > 0.
  - `assert x > 0, “Wrong input entered.”`
  - Python interpreter checks if x>0 is True or False.
    - If it is True, then the next statements will execute.
    - Else, it will display assertion error.

return statement

- Used in user defined functions.
- Will be discussed later.