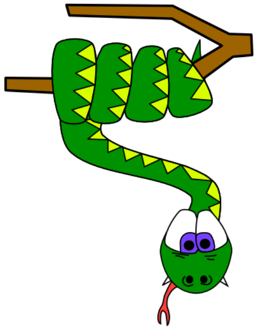

Control Statements

Session-2

Outline

- if-else
- Loop
- break
- continue
- pass

Control structures

- Decision making statements
  1. if-else
  2. if-elif-else
- **Note:**
- There is no switch case statement in Python
- We can do this easily enough

1. with a sequence of **if... elif... elif...else**"

Iterative Statements–

- **Loop statements:**
- Allows repeated execution of a statement or a set of statements multiple times based on the specified condition or range.
    1. While Loop
    2. For Loop
    3. Range
- **Loop Control Statements:**
- Are used to **change flow of execution** from its normal sequence-
- 1. Break
- 2. Continue
- 3. Pass

Indentation in Python

- Python uses offside rule notation for coding
- ***Uses indentation for blocks, instead of curly brackets***
- ***The delimiter followed in Python is a colon (:)*** and indented spaces or tabs

if-else statement

```
x= 3
if x > 5:          #Press enter after colon(:)
        print("true")
        print(x)
else:
        print ("false")
        print(“still in else block”)
print ("Out of if block")
```

if-else statement syntax

- if condition1:

  statement(s)
- elif condition2:

  statement(s)
- elif condition3:

  statement(s)
- else:

  statement(s)

while loop

- Repeats execution of a statement or a set of statements while a given condition is TRUE.
- Checks the condition each time before executing the statements in body of loop.
- x=1
- while x<=7:

```
        print(x)
        x+=1
```

for loop

- Repeats execution of a sequence of statements for a specific number of times
- **Syntax:**
- **for variable in sequence:**
  statement_1

```
        statement_2
          .
          statement_n
```

- for x in 1,2,‘hello', 7,’world',5.16:

  ```
          print(x)
  ```

Out put:

```
1
2
hello
7
world
5.16
```

```
for y in “python”: #iterating over string
        print(y)
```

Out put:

```
p
y
t
h
o
n
```

range() function

- Range is a built-in function that creates a list of integers.
- nums = range(6) *# creates a list of integers*
- print (nums) # Prints "[0, 1, 2, 3, 4, 5]"
- range(1,6) # 1, 2, 3, 4, 5
- range(0,6,2) # 0, 2, 4, increments of 2
- range(6,1,-2) # 6,4,2, decrements of 2

range() function in loops

- Used in case the need is to iterate over a specific number of times within a given range in steps/intervals mentioned. **Syntax is** :
- **range(lower limit*, upper limit, Increment_by/decrement_by*)**

  *** means optional**

Loop	Out put	Explanation

for i in **range(6)**:
        print(i)
	0,1,2,3,4,5	Prints all the values in given range from 0, **exclusive of upper limit**

for i in **range(1,6)**:
        print(i)
	1,2,3,4,5	Prints all the values in given range exclusive of upper limit

for i in **range(0,6,2)**:
        print(i)
	0,2,4	Prints values in given range in **increments of 2**

for i in **range(6,1,-2)**:
        print(i)
	6,4,2	Prints values in given range in **decrements of 2**

for ch in **"Hello World"**:
        print(ch)
		Prints all the characters in the string

break

- When an **external condition is triggered**, **Exits a loop** immediately.
- Break Statement:Terminates execution of loop statements and resumes execution at statement immediately following the loop. e.g.

```
x=1
while x<=7:
    print(x)
    x+=1
    if x==5:
        break
print("out of loop")
```

continue

- Causes the next iteration of the loop to execute and immediately retest its condition prior to reiterating.

```
x=1
while x<=7:
    if x==5:
        x=x+1
        continue
    print(x)
    x=x+1

print("out of loop")
```

pass

- Pass statement is never executed.
- Used when a **statement is syntactically required** and do not want any code to execute now.
- If there is a need to implement code in future.
- Behaves like a **placeholder for future code.**