Python : Decision Making and Looping

Tushar B. Kute,
http://tusharkute.com

Decision Making Statement

if-elif-else

1

```
if condition:
    statements
```

2

```
if condition:
    Statement-1
else:
    Statement-2
```

3

```
if condition:
    statement-1
elif condition:
    Statement-2
else:
    Statement-3
```

if-else

```
num = 23
if num > 0:
    print('Positive Number')
else:
    print('Negative Number')
```

Previous line of indentation

Indentation

if-elif-else

```
num = -85
if num == 0:      # Compulsory
    print('It is ZERO')
elif num > 0:     # Optional, Repeatable
    print('Positive Number')
else:             # Optional
    print('Negative Number')
```

Condition in if-else

- The condition in if-statement is parenthesis free.
- The condition can have ‘and’ and ‘or’ operator.
- It should end with a colon.
- It must return either True or False.
- The zero value is considered as False otherwise True.
  - Example:

    if True:

Exercise

- Write a program to read a number from user and find whether number is odd or even.

Nesting of if-else

- Example:
  - Read three numbers from keyboard and find largest of them. You are not allowed to use ‘and’ and ‘or’ operators.

Example:

```
num1 = int(input("Enter first:"))
num2 = int(input("Enter second:"))
num3 = int(input("Enter third:"))
if num1 > num2:
    if num1 > num3:
        print ("Largest is", num1)
    else:
        print ("Largest is", num3)
else
    if num2 > num3:
        print ("Largest is", num2)
    else:
        print ("Largest is", num3)
```

Example:

- Read an year and find whether that year is leap or not?
- Conditions for being leap year:
  - It should be divisible by four.
  - If its a century year, it should be divisible by 400 also.

Solution:

```
year = int(input("Enter the year: "))
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            print ('Leap')
        else:
            print ('NOT Leap')
    else:
        print ('Leap')
else:
    print ('NOT Leap')
```

Using ‘and’ and ‘or’

```
year = int(input("Enter the year: "))
if year%4==0 and year%100!=0 or year%400 ==0:
    print('LEAP')
else:
    print('NOT Leap')
```

Repeatable elif

- Read marks in percentages from student and find the grade of it as per below condition:
  - Above 75% : ‘Distinction’
  - 60% to below 75%: ‘First Class’
  - 50% to below 60%: ‘Second Class’
  - 40% to below 50%: ‘Pass Class’
  - Below 40%: ‘Failed’

Solution

```
marks = float(input('Enter the marks: '))

if marks >= 75:
    print('Distinction')
elif marks >= 60:
    print('First Class')
elif marks >= 50:
    print('Second Class')
elif marks >= 40:
    print('Pass Class')
else:
    print('FAILED')
```

Another way

```
marks = float(input('Enter the marks: '))

if marks < 40:
    print('FAILED')
elif marks < 50:
    print('Pass Class')
elif marks < 60:
    print('Second Class')
elif marks < 75:
    print('First Class')
else:
    print('Distinction')
```

Creating menu based programs

- Python does not have switch-case statement. So menu based programs are created simply using if-elif-else.
- Example:
  - Read a number from user and do the following by showing menu:
    - Find its square
    - Find cube
    - Check odd / even
    - Check positive / negative

Solution:

```
num = int(input("Enter the number: "))
print ("* Menu *\n1.Square\n2.Cube\n3.Odd\n4.+ve")
ch = int(input("Enter the choice: "))
if ch == 1:
    print ("Square:", num * num)
elif ch == 2:
    print ("Cube:", num * num * num)
elif ch == 3:
    if num % 2 == 0:
        print ("Even")
    else:
        print ("Odd")
elif ch == 4:
    if num > 0:
        print ("+ve")
    else:
        print ("-ve")
else:
    print ("Wrong Choice")
```

Exercises

- Write a Python program to calculate profit or loss. Input is selling cost and actual cost.
- Write a Python program to check whether a character is uppercase or lowercase alphabet.
- Write a Python program to input electricity unit charges and calculate total electricity bill according to the given condition:
  - For first 50 units Rs. 0.50/unit
  - For next 100 units Rs. 0.75/unit
  - For next 100 units Rs. 1.25/unit
  - For unit above 250 Rs. 1.50/unit
  - An additional surcharge of 17% is added to the bill

Exercises

- **Input -> basic salary**
- **Output -> gross salary**

  DA = 75% of basic

  HRA = 20% of basic
- **Conditions:**

  < 10000 : gross = da + basic

  >= 10000 and < 20000 : gross = da + basic + 50% of hra

  >= 20000 : gross = basic + da + hra
- **Sample: Input and Output**

  5000 : 3750 + 5000 = 8750

  12000 : 9000 + 12000 + 1200 = 22200

  24000 : 18000 + 24000 + 4800 = 46800

Loops


Python
Loops
while loops
for loops
Nested Loops
Loop Control Statements

The while loop

```
while condition:          #compulsory
   statement1
   statement2
else:                     #optional
   statements
```

Example-1

- Print your name for 10 number of times.

```
count = 0                # initialize
while count < 10:        # condition
    print('Tushar')
    count += 1           # change counter
```

Exercise

- Find addition of first ten natural numbers. That is, find addition of numbers from 1 to 10.

Solution

```
count, add = 1, 0
while count <= 10:
    add = add + count
    count += 1
else:
    print('Addition is', add)
```

```
Addition is 55
```

Exercise

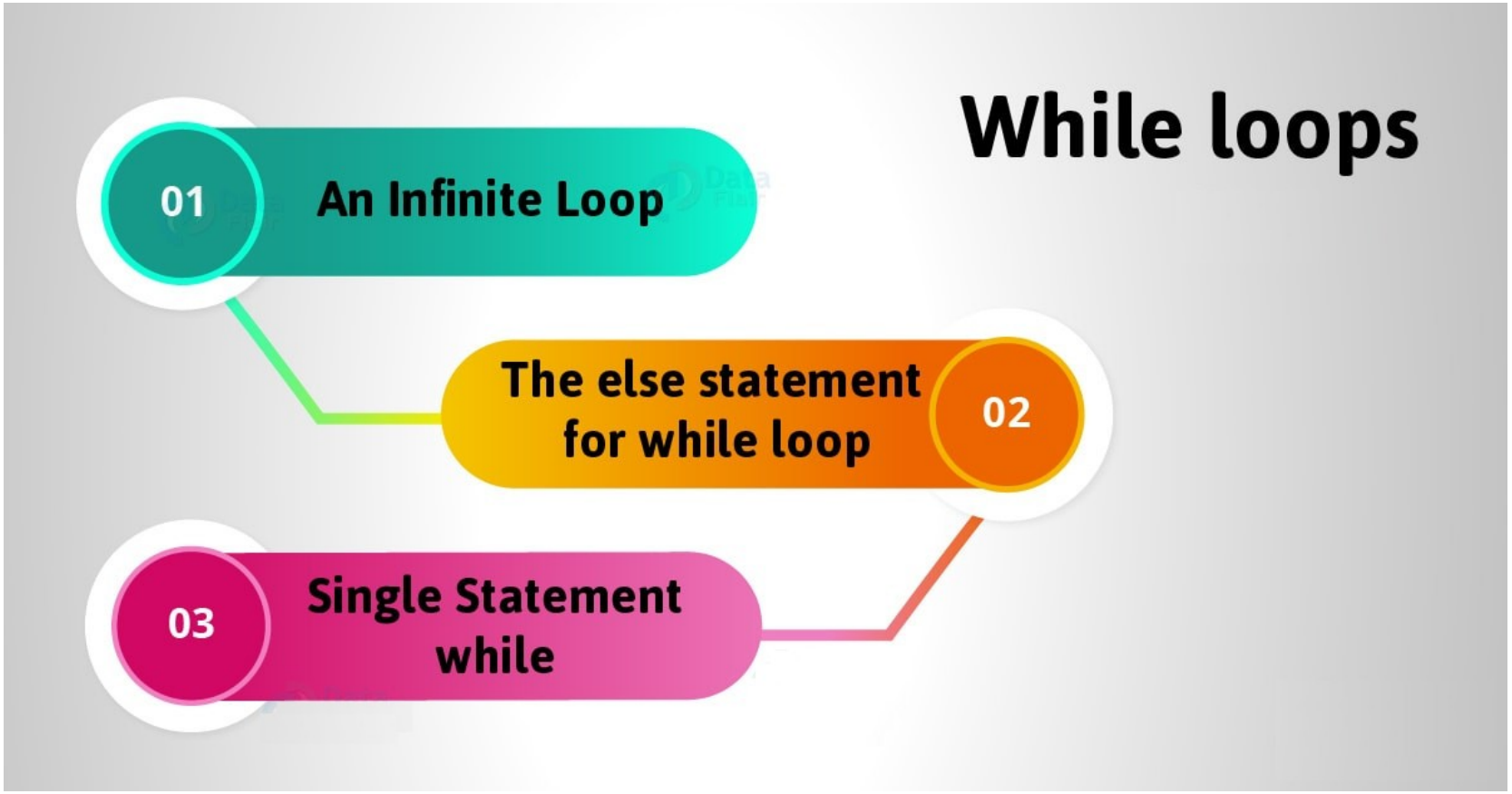
- Find addition of all the odd numbers from 1 to 20. That is, find addition of numbers 1, 3, 5 … 19.

Solution

```
count, add = 1, 0
while count <= 20:
    add = add + count
    count += 2
else:
    print('Addition is', add)
```

```
Addition is 100
```

More about ‘while’



Infinite loop

- Be careful while using a while loop. Because if you forget to increment the counter variable in python, or write flawed logic, the condition may never become false.
- In such a case, the loop will run infinitely, and the conditions after the loop will starve.
- To stop execution, press Ctrl+C. However, an infinite loop may actually be useful.
- While is the only loop where we can make while loop intentionally infinite.

Examples:

```
num = 0
while num < 10:
    print(num)

# Counter change missing
print('Ended')
```

1

2

```
while True:
    print('Tushar')
```

The else statement

- A while loop may have an else statement after it. When the condition becomes false, the block under the else statement is executed.
- However, it doesn't execute if you break out of the loop or if an exception is raised.

```
num = 0
while num < 10:
    print(num)
    num += 1
else:
    print('Ended')
```

Single statement while

- We can write single statement in while loop. This statements will be separated by semicolon.

```
num = 0
while num < 10: print(num); num += 1
```

The ‘for’ loop

- Python for loop can iterate over a sequence or collection of items.
- The structure of a for loop in Python is different than that in C++ or Java.
- That is, for(int i=0;i<n;i++) won’t work here. In Python, we use the ‘in’ keyword.
- Syntax:

```
for n in sequence:
    statements
```

n : variable
sequence: collection of elements

while vs for loop

while loop	for loop
Can have condition for iteration	Don’t have any loop condition
The counter can be incremented or decremented by any number	The loop can iterate through sequence by one increment only
The loop can be infinite	This loop can’t be infinite

Examples:

```
>>> for n in 1,6,7,8,4:
...     print(n, end=' ')
...
1 6 7 8 4
>>> for n in 1,6,7,8,4:
...     print(n*n, end=' ')
...
1 36 49 64 16
>>> for n in 'h',23,3.12,12:
...     print(n, end=' ')
...
h 23 3.12 12
```

Same sequence

Operation on data

Mixed Elements

More on ‘for’ loop

The range( ) function

- This function yields a sequence of numbers. When called with one argument, say n, it creates a sequence of numbers from 0 to n-1.
- Example:

```
>>> arr = range(10)
>>> list(arr)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> arr = range(1,6)
>>> list(arr)
[1, 2, 3, 4, 5]
>>> arr = range(1,12,2)
```

Using range( ) function

```
>>> for x in range(10):
        print(x, end=' ')

0 1 2 3 4 5 6 7 8 9
>>> for x in range(1,10,2):
        print(x, end=' ')

1 3 5 7 9
```

The range( ) decrement operation

```
>>> arr = range(12,2,-2)
>>> list(arr)
[12, 10, 8, 6, 4]
```

Iterating through sequence

```
X = [4,7,9,1,5]          # list
>>> for num in x:
      print(num, end=' ')

4 7 9 1 5
>>> s = 'MITU Skillologies'    # string
>>> for data in s:
      print(data, end=' ')

M I T U   S k i l l o l o g i e s
```

Collection of strings

```
>>> col = ['Mar','Hin','San','Nep']
>>> for name in col:
        print(name)
Mar
Hin
San
Nep
```

The else statement

- Like a while loop, a for-loop may also have an else statement after it. When the loop is exhausted, the block under the else statement executes.

```
>>> for x in range(10):
        print(x, end=' ')
    else:
        print('Loop Ended')

0 1 2 3 4 5 6 7 8 9 Loop Ended
```

Example:

- Print your name 10 times using for loop.

```
for n in range(10):
    print('Tushar')
```

Example:

- Find addition of first 10 natural numbers using for loop.

```
add = 0
for n in range(1,11):
    add += n
else:
    print('Sum:',add)
```

Example:

- Find addition of all odd numbers from 1 to 20 using for loop.

```
add = 0
for n in range(1,21,2):
    add += n
else:
    print('Sum:',add)
```

Exercises

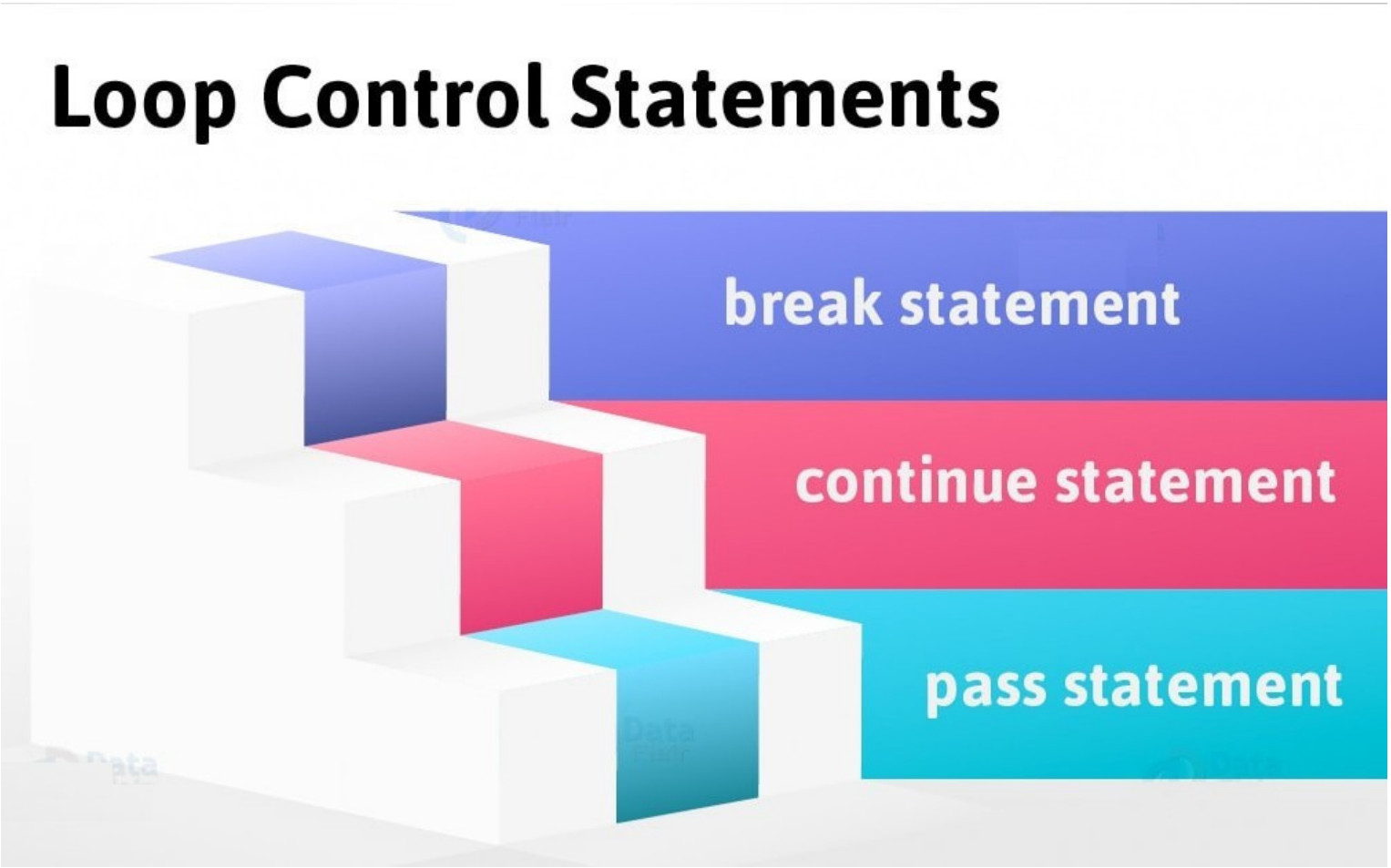
- Read a number from user and find the factorial of the number.
- Take a number user input and find sum of digits of this number.
- Write a program to print all the odd numbers from 10 to 30.
- Read a number from keyboard and print it in reverse.
- Write a program to print Fibonacci series up to n terms.

Nesting of loops

- You can also nest a loop inside another.
- You can put a for loop inside a while, or a while inside a for, or a for inside a for, or a while inside a while.
- Or you can put a loop inside a loop inside a loop. You can go as far as you want.

```
for i in range(1,6):
    for j in range(i):
        print("*",end=' ')
    print()
```

Nesting of loops



The ‘break’ statement

- When you put a break statement in the body of a loop, the loop stops executing, and control shifts to the first statement outside it. You can put it in a for or while loop.

```
for i in range(10):
    if i == 5:
        break
```

The ‘continue’ statement

- When the program control reaches the continue statement, it skips the statements after ‘continue’.
- It then shifts to the next item in the sequence and executes the block of code for it. You can use it with both for and while loops.

```
for i in range(10):
    if i % 2 == 0:
        continue
    print(i, end=' ')
```

```
1 3 5 7 9
```

The ‘pass’ statement

- We use the pass statement to implement stubs.
- When we need a particular loop, class, or function in our program, but don’t know what goes in it, we place the pass statement in it.
- It is a null statement.
- The interpreter does not ignore it, but it performs a no-operation (NOP).
- It is used to fill indented space

```
while True:
    pass
```

Exercises

- Write a program to check whether entered number if prime or not.
- Write a program to print all the prime numbers from 5 to 50.
- Print following pattern:

  1

  2 2

  3 3 3

  4 4 4 4

  5 5 5 5 5
- Find power of x raised to y from user input.

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

c/MITUSkillologies

Web Resources
http://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com