PYTHON

LOOPS

‘For’ loop: it is used to iterate over a sequence(eg. A list, tuple, string or range

```
# Syntax of a for loop
numbers = [1, 2, 3]
for item_name in numbers
      print(item_name)



# for loop – List – example 1
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
      print(fruit)
# Output:
# apple
# banana
# cherry

# example 2
my_list = [1, 2, 3, 4]
for jelly in my_list:
       print('hello’)
# Output:
hello
hello
hello
Hello

# example 3
mylist = [1, 2, 3, 4]
for num in mylist:
# Check for even
if num % 2 == 0:
       print(f'{num} is even’)
else:
      print(f'{num} is odd’)
```

```
# output:
1 is odd
2 is even
3 is odd
4 is even



# for loop – strings
for char in "Hello":
       print(char)
# Output:
# H
# e
# l
# l
# o

# for loop – tuples
#example 1
mylist = [(1, 2), (3, 4), (5, 6), (7, 8)]
for item in mylist:
        print(item)
#output
(1, 2)
(3, 4)
(5, 6)
(7, 8)

#example 2
mylist = [(1, 2), (3, 4), (5, 6), (7, 8)]
for a, b in mylist:
       print(a)
# output
1
3
5
7
```

PYTHON

```python
# for loop – dictionary
# example 1
d = {'K1': 1, 'K2': 2, 'K3': 3}
for item in d:
    print(item)
# output:
K1
K2
K3

# using .values() method
d = {'K1': 1, 'K2': 2, 'K3': 3}
for value in d.values():
    print(value)

#output
1
2
3

# using .items() method – to get both keys and values
d = {'K1': 1, 'K2': 2, 'K3': 3}
for key, value in d.items():
      print(key, value)

#output
K1 1
K2 2
K3 3
```

The ‘while’ loop in python continues to execute a block of code as long as a given condition remains ‘true’

```python
# syntax
while some_boolean_condition:
     # do something while the condition is True

else:
     # do something different after the loop ends

# simple while loop
x = 0
while x < 5:
    print(f'The current value of x is {x}’)
    x += 1

#output
The current value of x is 0
The current value of x is 1
The current value of x is 2
The current value of x is 3
The current value of x is 4

#  while loop with an else block
x = 0
while x < 5:
    print(f'The current value of x is {x}’)
    x += 1
else:
    print("x is not less than 5")

#output
The current value of x is 0
The current value of x is 1
The current value of x is 2
The current value of x is 3
The current value of x is 4
x is not less than 5
```

PYTHON

break, continue, pass
we can use break, continue and pass statements in our loops to add additional functionality for various cases

the three statements are defined by:

- break : breakout of the current enclosing loop
- continue : goes to the top of the closest enclosing loop
- pass : does nothing at all

```
# using ‘pass’ statement

x = [1, 2, 3]
for item in x:
    # comment
    pass
print('End of my script’)
#output:
End of my script

# using ‘continue’ statement
my_string = 'sammy’
for letter in my_string:
    if letter == 'a’:
        continue
    print(letter)
# output
s
m
m
Y

# using ‘break’ statement
my_string = 'sammy’
for letter in my_string:
      if letter == 'a’:
           break
      print(letter)
# output
s
```

```
# using while loop
x = 0
while x < 5:
    print(x)
    x += 1
# output:
0
1
2
3
4

# using while loop with break statement
x = 0
while x < 5:
    if x == 2:
        break
    print(x)
    x += 1
# output:
0
1
```

PYTHON

The **range()** function in Python is used to generate a sequence of numbers.
It creates a range object representing a sequence of integers.
The function can take up to three arguments: **start**, **stop**, and **step_size**.
The syntax for the **range()** function is:

```
range(start, stop, step_size)


# Using the range function
for num in range(10):
    print(num)
# output
0
1
2
3
4
5
6
7
8
9



# Using the range function with start and end
for num in range(3, 10):
     print(num)
# output
3
4
5
6
7
8
9
```

```
# Using the range function with start, end, and step
for num in range(0, 10, 2):
    print(num)
# Output:
0
2
4
6
8

# Using list() with range to create a list from a range object
mylist = list(range(0, 11, 2))
     print(mylist)
# Output:
[0, 2, 4, 6, 8, 10]

# Using enumerate to get the index and letter from a string
index_count = 0
for letter in 'abcde’:
     print('At index {}, the letter is {}'.format(index_count, letter))
index_count += 1
# output:
At index 0, the letter is a
At index 1, the letter is b
At index 2, the letter is c
At index 3, the letter is d
At index 4, the letter is e

# Using enumerate directly with a string
word = 'abcde’
for item in enumerate(word):
    print(item)
# Output:
(0, 'a’)
(1, 'b’)
(2, 'c’)
(3, 'd’)
(4, 'e’)
```

PYTHON

```
# Using enumerate to get index and letter separately
word = 'abcde’
for index, letter in enumerate(word):
print(index)
print(letter)
print('\n’)

# Output:
0
a

1
b

2
c

3
d

4
e

# Using zip to combine two lists
mylist1 = [1, 2, 3]
mylist2 = ['a', 'b', 'c’]
for item in zip(mylist1, mylist2):
    Print(item)
# Output:
(1, 'a’)
(2, 'b’)
(3, 'c’)
```

```
# Using the in operator to check membership
print('x' in [1, 2, 3]) # False
print('x' in ['x', 'y', 'z']) # True
print('a' in 'a world') # True
print('mykey' in {'mykey': 345}) # True
d = {'mykey': 345}
print(345 in d.keys()) # False


# Using min and max functions
mylist = [10, 20, 30, 40, 100]
     print(min(mylist)) # 10
     print(max(mylist)) # 100

# Using shuffle from random to shuffle a list
from random import shuffle
mylist = [1, 2, 3, 4, 5, 6, 7]
shuffle(mylist)
print(mylist)
# Output:
[6, 2, 4, 3, 5, 7, 1]

# Using randint from random to generate a random integer
from random import randint
mynum = randint(0, 100)
print(mynum) # Output will be a random integer between 0 and 100

# Getting user input using input function
result = input('Enter a number here: ')
```

PYTHON

Functions allows us to create blocks of code that can be easily executed many times, without needing to constantly rewrite the entire block of code.

A function is a group of statements performing a specific task

```
# Basic function without parameters
def name_of_function():
     ''’
     This is a docstring that explains the function.
     ''’
     print("Hello")

name_of_function() # Call the function
# output
Hello

# Function with a parameter
def greet_person(name):
    print("Hello, " + name)

greet_person("Alice") # Call the function with argument "Alice“

# output
Hello, Alice


# Example 3: Function with parameters and return statement
def add_function(num1, num2):
    return num1 + num2
result = add_function(5, 3) # Call the function with arguments 5 and 3
print(result) # Print the result

# output
8
```

```
#example
def say_hello():
   print("hello")
   print("are")
   print("you")
say_hello()

#output
hello
are
You

# using an f-string
def say_hello(name):
   print(f'hello {name}’)

say_hello('jose’)

#output
hello jose


#example
def add_num(num1, num2):
       return num1 + num2

result = add_num(10, 20)
print(result) #output-30

# Function to return the result
def return_result(a, b):
     return a + b

result = return_result(10, 20)
print(result)     #output-30
```

PYTHON

```python
#  Function with print
def myfunc(a, b):
    print(a + b)
    return a + b
result = myfunc(10, 20)    #output-30
print(result)     #output-30

# Function with logic for summing numbers
def sum_numbers(num1, num2):
       return num1 + num2

print(sum_numbers(10, 20)) # Output: 30
print(sum_numbers('10', 20)) # Output: TypeError (unsupported operand type(s) for +)

# Function to check if a number is even
def even_check(number):
    result = number % 2 == 0
    return result

print(even_check(20)) # Output: True
print(even_check(21)) # Output: False

# Function to check if any number is even inside a list
def check_even_list(num_list):
      for number in num_list:
          if number % 2 == 0:
              return True
           else:
                pass

print(check_even_list([1, 3, 5])) # Output: None (No even number)
print(check_even_list([2, 4, 5])) # Output: True (Contains even number)
print(check_even_list([2, 1, 1, 1])) # Output: True (Contains even number)
```

```python
# Function to return all even numbers in a list
def check_even_list(num_list):
    even_numbers = []

    for number in num_list:
         if number % 2 == 0:
           even_numbers.append(number)
          else:
               pass
    return even_numbers

print(check_even_list([1, 3, 5])) # Output: []
print(check_even_list([2, 4, 5])) # Output: [2, 4]
print(check_even_list([2, 1, 1, 1])) # Output: [2]

# Tuple unpacking with stock prices
stock_prices = [('APPL', 200), ('GOOG', 400), ('MSFT', 100)]

for item in stock_prices:
    print(item)

# output
('APPL', 200)
('GOOG', 400)
('MSFT', 100)

# Tuple unpacking for ticker and price
for ticker, price in stock_prices:
     print(ticker)
# output
APPL
GOOG
MSFT
```

PYTHON

```python
# Tuple unpacking for calculation
for ticker, price in stock_prices:
    print(price + (0.1 * price))
# output
220.0
440.0
110.0

# Employee of the month using a function and tuple unpacking
work_hours = [('Abby', 100), ('billy', 400), ('cassie', 800)]

def employee_check(work_hours):
    current_max = 0
    employee_of_month = ''

    for employee, hours in work_hours:
        if hours > current_max:
            current_max = hours
            employee_of_month = employee
        else:
            pass

    return (employee_of_month, current_max)

result = employee_check(work_hours)
print(result)

#output
('cassie', 800)

# Assigning function result to variables using tuple unpacking
name, hours = employee_check(work_hours)
print(name)   # Output: cassie
print(hours)  # Output: 800
```