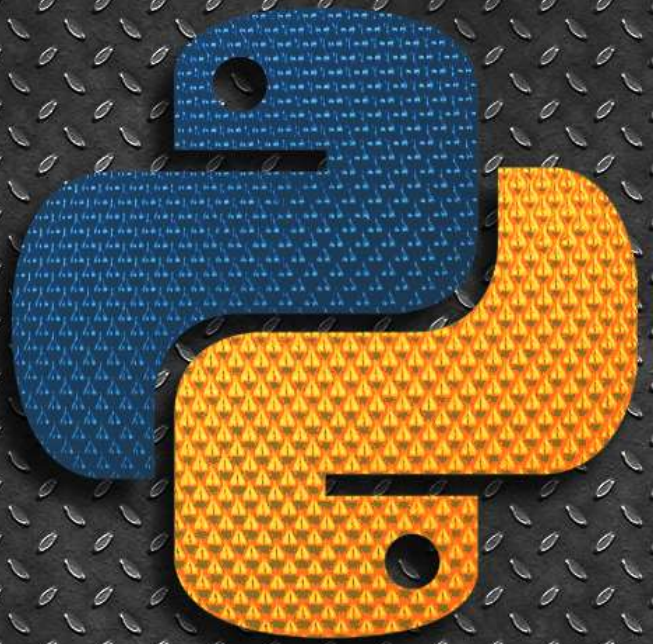
Python Fundamentals



BREAK & CONTINUE

Break Keyword

you may want to exit the entire loop when a certain condition is met.

To do that, we use the break keyword.

```
j = 0
for i in range(5):
    j = j + 2
    print ('i = ', i, ', j = ', j)
    if j == 6:
        break
```

```
You should get the following output.
i = 0 , j = 2
i = 1 , j = 4
i = 2 , j = 6
```

Continue Keyword

The rest of the loop after the keyword is skipped for that iteration.

```
j = 0
for i in range(5):
    j = j + 2
    print ('i = ', i, ', j = ', j)
    if j == 6:
        continue
    print('j value is `,j)
```

```
i =  0 , j =  2
j value is  2
i =  1 , j =  4
j value is  4
i =  2 , j =  6
i =  3 , j =  8
j value is  8
i =  4 , j =  10
j value is  10
```

Try, Except Statement

Controls how the program proceeds when an error occurs.

The syntax is as follows:

try:

do something

except:

do something else when an error occurs

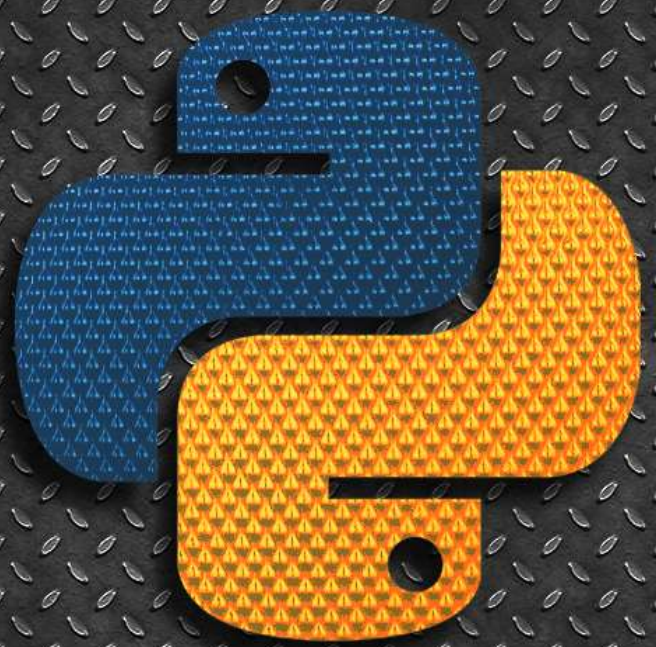
```
try:
        answer =12/0
        print (answer)
except:
        print ("An error occurred")
```

Loop Assignment

Print the list of numbers which are divisible by 5 and multiple of 8 between 2000 and 2500 (both included)

Write a Python program to create the multiplication table (from 1 to 10) of a number getting input from the user

Python Fundamentals



Functions

Functions

Pre-written code to perform a certain task.

There are built in functions like print() , upper() etc.

We can also define our own functions using the keywords def and return.

```
def functionName(parameters):
    code what the function should do
    return [expression]
```

Functions - Example

Pre-written code to perform a certain task.

There are built in functions like print() , upper() etc.

We can also define our own functions using the keywords def and return.

```
def findSum(a,b):
    sum = a+b
    return sum


print(findSum(2,3))
```

Functions – Example Exercise

A **prime number is** a natural number greater than 1, which **is** only divisible by 1 and itself.

First few **prime numbers** are : 2 3 5 7 11 13 17 19 23

```
def checkIfPrime (numberToCheck):
    for x in range(2,numberToCheck):
        if (numberToCheck%x == 0):
            return False
        return True
```

Functions – Variable Scope

- Any variable declared inside a function is only accessible within the function.
- These are known as local variables.

- Any variable declared outside a function is known as a global variable
- It is accessible anywhere in the program.

Functions – Variable Scope

```
message1 = "I am Global Variable"

def myFunction():
    print("\n Inside the function")
    #Global variables are accessible inside a function
    print (message1)
    #Declaring a local variable
    message2 = "I am Local Variable"
    print (message2)

#Calling the function
myFunction()
print("\nOUTSIDE THE FUNCTION")

#Global variables are accessible outside function
print (message1)

#Local variables are NOT accessible outside function.
print (message2)
```

Functions – Passing Arbitrary List as Argument

Python allows a function to collect an arbitrary (random) number of arguments from the calling statement.

```
def make_pizza(size, *toppings):
    print(f"\nMaking a {size} -inch pizza with toppings:")
    for topping in toppings:
        print(f"- {topping}")

make_pizza(16, 'pepperoni')
make_pizza(12, 'mushrooms', 'green peppers’)
```

```
Making a 16 -inch pizza with toppings:
- pepperoni
Making a 12 -inch pizza with toppings:
- mushrooms
- green peppers
```

Functions – Required and Keyword Args

Required arguments are the arguments passed to a function in correct positional order.

Here, the number of arguments in the function call should match exactly with the function

```
def printme( str ):
    "This prints a passed string into this function"
    print (str)
    return

# Now call printme function
printme(str="test")
```

Functions – Required and Keyword Args

Keyword arguments in a function call, the caller identifies the arguments by the parameter name.

```
def printinfo( name, age ):
    "This prints a passed info into this function"
    print ("Name: ", name)
    print ("Age ", age)
    return

# call printinfo function
printinfo( age=50, name="miki" )
```