

NAME: DANIYAL SAEED

SAP ID: 53937

ASSIGNMENT NO 1

Keywords Exploration:

- Write a Python program to display all reserved keywords using keyword.kwlist.

```
main.py + Run Share $ Command Line Arguments
```

```
1 import keyword
2 print(keyword.kwlist)
3 print("\nDANIYAL SAEED")
```

```
[ 'False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class',
  'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'glob',
  'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
  'return', 'try', 'while', 'with', 'yield']

DANIYAL SAEED

** Process exited - Return Code: 0 **
```

- Try to use one keyword as a variable name and observe what happens. Explain the error.

The screenshot shows a Python code editor interface. On the left, there is a file browser with a single file named "main.py". The code editor window contains the following Python code:

```
1 def = 10
2 print(def)
3
```

On the right, the output window displays the results of running the code. It shows the code being executed, followed by an error message:

```
File "...main.py"... line 1
def = 10
^
SyntaxError: invalid syntax
>-
** Process exited - Return Code: 1 **
```

- **Keywords** in Python are reserved words that have special meaning in the language (like `def`, `if`, `else`, `for`, `while`, etc.).
- You **cannot** use these keywords as variable names because Python uses them to understand the structure and flow of the code.
- Using a keyword as a variable name confuses the Python interpreter, so it raises a syntax error.

2. Working with Data Types:

- Create three variables: an integer, a float, and a string.
- Print their values and use the `type()` function to show their data types.
- Convert the float into an integer and print the result.

```

main.py + Run Share $ Command Line Arguments
1 integer = 30
2 floats = 35.3
3 string = 'Rawalpindi'
4 print("\t\t CREATING 3 VARIABLE")
5
6
7 print (integer)
8 print (floats)
9 print (string)
10
11 print("\n")
12 print("\t\t SHOWING VARIABLE TYPE")
13
14 print (type(integer))
15 print (type(floats))
16 print (type(string))
17
18 print("\n")
19 print("\t\t SHOWING INTO INTEGER VAL")
20
21
22 a = int(floats)
23 print(a)
24 print("\n")
25 print("DANIYAL SAEED")

```

CREATING 3 VARIABLE
30
35.3
Rawalpindi
SHOWING VARIABLE TYPE
<class 'int'>
<class 'float'>
<class 'str'>
SHOWING INTO INTEGER VAL
35
DANIYAL SAEED
** Process exited - Return Code: 0 **

3. Escape Sequences in Strings

- Write a program that demonstrates at least four escape sequences:
- Newline (\n)
- Tab (\t)
- Double quotes inside a string (\")
- Backslash (\\\)
- Print the results and explain the formatting changes.

```

main.py + Run Share $ Command Line Arguments
1 integer = 30
2 floats = 35.3
3 string = 'Rawalpindi'
4 print("\t\t CREATING 3 VARIABLE")
5
6
7 print (integer)
8 print (floats)
9 print (string)
10
11 print("\n")
12 print("\t\t SHOWING VARIABLE TYPE")
13
14 print (type(integer))
15 print (type(floats))
16 print (type(string))
17
18 print("\n")
19 print("\t\t SHOWING INTO INTEGER VAL")
20
21
22 a = int(floats)
23 print(a)
24 print("\n")
25 print("DANIYAL SAEED")

```

CREATING 3 VARIABLE
30
35.3
Rawalpindi
SHOWING VARIABLE TYPE
<class 'int'>
<class 'float'>
<class 'str'>
SHOWING INTO INTEGER VAL
35
DANIYAL SAEED
** Process exited - Return Code: 0 **

4. Triple-Quoted Strings & Docstrings

- Write a function with a docstring that explains what the function does.
- Inside the function, use a triple-quoted string to print a paragraph spanning multiple lines.
- Call the function and display both its output and its `__doc__`.

The screenshot shows a code editor window with a Python file named `main.py`. The code defines a function `describe_earth` with a multi-line docstring. It then calls this function and prints its docstring. The output pane shows the function's docstring and the name of the user who ran the code.

```
main.py + Run Share $ Command Line Arguments
1- def describe_earth():
2     """
3         This function prints a paragraph describing Earth.
4         It demonstrates the use of triple-quoted strings inside a function.
5     """
6     print("""
7         Earth is the third planet from the Sun and the only astronomical object known to harbor life.
8         About 71% of Earth's surface is covered with water, mostly by oceans, with the remainder consisting of continents and islands.
9         The atmosphere protects life on Earth by absorbing ultraviolet solar radiation and reducing temperature extremes between day and night.
10    """)

11 # Call the function
12 describe_earth()

13 # Display the function's docstring
14 print(describe_earth.__doc__)

15
16
17
18
19 print("DANIYAL SAEED")
20
21
22
```

Output:

```
Earth is the third planet from the Sun and the only astronomical object known to harbor life.
About 71% of Earth's surface is covered with water, mostly by oceans, with the remainder consisting of continents and islands.
The atmosphere protects life on Earth by absorbing ultraviolet solar radiation and reducing temperature extremes between day and night.

This function prints a paragraph describing Earth.
It demonstrates the use of triple-quoted strings inside a function.

DANIYAL SAEED

** Process exited - Return Code: 0 **
```

5. Perform DMAS Operations

- Create two integer variables, `a = 15` and `b = 4`
- Perform and print the results of:
 - D (Division) → a / b (floating-point division) and $a // b$ (integer division)
 - M (Multiplication) $a * b$
 - A (Addition) $-a + b$
 - S (Subtraction) → $a - b$

The screenshot shows a code editor interface with a Python file named `main.py`. The code performs several arithmetic operations on variables `a` and `b`, and then prints the results and the author's name.

```
1 a = 15
2 b = 4
3
4 # Division
5 float_division = a / b      # floating-point division
6 int_division = a // b       # integer division
7
8 # Multiplication
9 multiplication = a * b
10
11 # Addition
12 addition = a + b
13
14 # Subtraction
15 subtraction = a - b
16
17 # Print results
18 print("Division (float):", float_division)
19 print("Division (int):", int_division)
20 print("Multiplication:", multiplication)
21 print("Addition:", addition)
22 print("Subtraction:", subtraction)
23
24
25
26 print("DANIYAL SAEED")
```

The output window shows the results of the operations and the final printed message:

- Division (float): 3.75
- Division (int): 3
- Multiplication: 60
- Addition: 19
- Subtraction: 11
- DANIYAL SAEED

** Process exited - Return Code: 0 **