SHU CS CUBE Python Workshop 1.0

Let's Print Hello World

1. Single Quotation
2. Double Quotation

With Double Quotation Marks

In [ ]:

```
print ("Hello World")
```

```
Hello World
```

Using single quotation marks

In [ ]:

```
print ('Hello World')
```

```
Hello World
```

Comments

1. Single Line #
2. Multiline """ """

Single line comment

In [ ]:

```
#Hey, this is a comment
print ("I just added a comment")
```

```
I just added a comment
```

Multi-line comment

In [ ]:

```
"""
This is comment
This is a multi line commment
"""
print ("I just added multi line comment")
```

```
I just added multi line comment
```

Variables

1. Varibale assignment
2. Valid variable names
3. Single quotation and double quotation strings
4. Case sensitive
5. Assigning Values

    - One Value to multiple variables
    - Multiple Values to multiple variable

In [ ]:

```
name= "Uzma"
print (name)
name= 23
print (name)
```

```
Uzma
23
```

In [ ]:

```
x=y=z = "CS Cube"
print (x,y,z)
```

```
CS Cube CS Cube CS Cube
```

In [ ]:

```
x, y, z= "SHU", "CS Cube", "Workshop1"
print( x, "'s",y,"has organized", z)
```

```
SHU 's CS Cube has organized Workshop1
```

Casting- specifying datatype

In [ ]:

```
x= str(3)  #'3'
y = int(3) #3
z= float(3) #3.0
print (x," ",y," ",z)
```

```
3   3   3.0
```

In [ ]:

```
x = int(1)   # x will be 1
y = int(2.3) # y will be 2.3
z = int("90") # z will be 90
print(x,y,z)
```

```
1 2 90
```

In [ ]:

```
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
print (z,w)
```

```
3.0 4.2
```

In [ ]:

```
var1 = str("SHU") # var1 will be 'SHU'
var2 = str(25)    # var2 will be '25'
var3 = str(3.0)  # var3 will be '3.0'
print (var1,var2,var3)
```

```
SHU 25 3.0
```

Type() function

In [ ]:

```
x="Uzma"
y=18
print(type(x))
print(type(y))
```

```
<class 'str'>
<class 'int'>
```

Unpacking- Extracting Values from list or tuples into variables

In [ ]:

```
Names= ["SHU", "CS Cube", "Workshop1"]
x,y,z = Names
print(x,y,z)
```

```
SHU CS Cube Workshop1
```

Print Statement

1. Separated with Comma
2. using + operator
3. Mathematical Operation inside print()

In [ ]:

```
var1= "CS"
var2= "Cube"
var3= "Python"
var4= "Workshop"
num1= 1
num2= 2
print(var1,var2)
print(var3+var4)
print(var4,var5)
print(num1+num2)
print(num2/num1)
print(num1-num2)
# You can't use + between a string and an integer type variable, you have to use comma
```

Local and Global Variables & the *global* keyword

In [ ]:

```
x= "Outside the function"
def func():
  x= "Inside the function" #Local Scope
  print(x)

func()
print(x)
```

```
Inside the function
Outside the function
```

In [ ]:

```
y="Outside the function"
def func():
  global y
  y= "Inside the function" #global Scope
  print(y)

func()
print(y)
```

```
Inside the function
Inside the function
```

Taking Input using input()

1. String type
2. Numeric type

In [ ]:

```
user_input = input("Enter something: ")
print("You entered:", user_input)
```

```
Enter something: Uzma
You entered: Uzma
```

In [ ]:

```
num_str = input("Enter a number: ")
num = int(num_str)  # Convert the input to an integer

result = num * 2
print("The result of doubling the number is:", result)
```

```
Enter a number: 2
The result of doubling the number is: 4
```

Strings

1. Multiline string
2. len()
3. str.upper() and str.lower()
4. str.strip()
5. str.find(substring)
6. str.replace(old, new)
7. str.startswith(prefix) and str.endswith(suffix)
8. str.isdigit() and str.isalpha()
9. Slicing
    - Slice From the Start
    - Slice To the End
    - Negative Indexing

In [ ]:

```
# You can also use multiline strings within your code.
message = """
This is a multiline string assigned to a variable.
It can be useful for creating formatted text.
"""

print(message)

a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

```
This is a multiline string assigned to a variable.
It can be useful for creating formatted text.

Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

In [ ]:

```
text = "Hello, World!"
length = len(text)
print("Length of the string:", length)
```

In [ ]:

```
text = "Hello, World!"
upper_case = text.upper()
lower_case = text.lower()
print(upper_case)
print(lower_case)
```

In [ ]:

```
text = "   Hello, World!   "
stripped_text = text.strip()
print(stripped_text)
```

In [ ]:

```
text = "Hello, World!"
index = text.find("World") #returns index 7
index2= text.find("Bye") #when it doesn't find, then returns -1
print(index,index2)
```

```
7 -1
```

In [ ]:

```
text = "Hello, World!"
starts_with_hello = text.startswith("Hello") # returns True or False
ends_with_exclamation = text.endswith("!")
print(starts_with_hello, ends_with_exclamation)
```

```
True True
```

In [ ]:

```
number = "12345"
is_digit = number.isdigit() #returns boolean value

word = "Python"
is_alpha = word.isalpha() #returns boolean value
print (is_digit,is_alpha)
```

```
True True
```

In [ ]:

```
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

In [ ]:

```
b = "Hello, World!"
print(b[:5])
```

```
Hello
```

In [ ]:

```
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

In [ ]:

```
b = "Hello, World!"
print(b[-5:-2])
```

```
orl
```

Formatting Strings

In [ ]:

```
name = "Charlie"
age = 40

formatted_string = "Hello, {}! You are {} years old.".format(name, age)
print(formatted_string)
```

```
Hello, Charlie! You are 40 years old.
```

In [ ]:

```
name = "Charlie"
age = 40

formatted_string = "Hello, {0}! You are {1} years old.".format(name, age)
print(formatted_string)
```

```
Hello, Charlie! You are 40 years old.
```

In [ ]:

```
name = "Charlie"
age = 40

formatted_string = "Hello, {namex}! You are {agex} years old.".format(namex=name, agex=age)
print(formatted_string)
```

```
Hello, Charlie! You are 40 years old.
```

If-Else

In [ ]:

```
a = 10
b = 20
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In [ ]:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

In [ ]:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

Shorthand If-Else

In [ ]:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

```
B
```

In [ ]:

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

```
Above ten,
and also above 20!
```

In [ ]:

```
a = 33
b = 200

if b > a:
  pass
```

While Loops

In [ ]:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

In [ ]:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

In [ ]:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)

# Note that number 3 is missing in the result
```

```
1
2
4
5
6
```

For Loops

In [ ]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

In [ ]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

In [ ]:

```
for x in range(6):
  print(x)
```

```
0
1
2
3
4
5
```

In [ ]:

```
for x in range(2, 6):
  print(x)
```

```
2
3
4
5
```

In [ ]:

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

In [ ]:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
3
4
5
Finally finished!
```

In [ ]:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Let's check if a string contains any vowel

In [ ]:

```
input_string = "hello, world"
vowel_count = 0

# Use a loop to count the vowels
for char in input_string:
    if char.lower() in "aeiou":
        vowel_count += 1

print("Input string:", input_string)
print("Vowel count:", vowel_count)
```

```
Input string: hello, world
Vowel count: 3
```

In [ ]:

```
for i in range(1, 6):
    print('*' * i)
```

```
*
**
***
****
*****
```

List

Ordered collection of mutable elements.

In [ ]:

```
a = []
a.append("a")
a.append("b")
a.append("c")
a.insert(1, 6)
a.remove('c')
a
```

Function

A reusable block of code that performs a specific task.

In [ ]:

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

print(sum(1,2))
```

Classes and Objects

Classes:

- Blueprint or template.
- Define the structure and behavior of objects.
- Act as a template for creating multiple objects.
- Contain attributes and methods.

Objects:

- Specific instances created from a class.
- Have unique attributes and behaviors.
- Represent real-world entities or data.
- Can be created using the class as a template.

In [ ]:

```
class Person:
    def __init__(self, name,sem):
      self.name = name
      self.semester = sem

    def introduce(self):
      return f"My name is {self.name} and I am from {self.semester} semester"

obj1 = Person("Fatima", 7)
obj1.introduce()
```

TO-DO List

In [6]:

```
class Todo:
  def __init__(self):
    self.tasks = []

  def add_task(self,data):
    self.tasks.append(data)
    print (data,"successfully added")

  def remove_task(self,data):
    self.tasks.remove(data)
    print (data,"successfully removed")

  def print_list(self):
    for task in self.tasks:
      print(task)


fatima = Todo()
fatima.add_task("abc")
fatima.add_task("def")
fatima.remove_task("abc")
# fatima.print_list()

uzma = Todo()
uzma.add_task(123)
uzma.print_list()
```

```
abc successfully added
def successfully added
abc successfully removed
123 successfully added
123
```