SIMPLE COMMAND:

```
# Simple command
print("Hello, World!")
```

OUTPUT: Hello, World!

COMMENTS IN PYTHON:

```
# Comments in Python
x = 1
#The initial value of x is 1.
if x>0:
    print("These are two comments")
```

OUTPUT: These are two comments

MULTIPLE STATEMENTS ON SINGLE LINE:

```
# Multiple Statements on a single line
print ("Statement1")
print ("Statement2")

#You can write above two statements in following way
print("Statement1"); print("Statement2")
```

OUTPUT: Statement1
Statement2
Statement1
Statement2

INDENTATION:

```
# No Indentation
x=1
if x>0 :
    print("This statement has no indentation")
        print("This statement has no indentation")
```

OUTPUT: print("This statement has no indentation")
IndentationError: unexpected indent

INDENTATION WITH SINGLE TAB:

```
# Indentation with single TAB
x=1
if x>0 :
    print("This statement has single tab indentation")
    print("This statement has single tab indentation")
```

OUTPUT: This statement has single tab indentation
This statement has single tab indentation

INDENTATION WITH SINGLE TAB + SPACE:

```
# Indentation has single TAB + SPACE
x=1
if x>0 :
     print("This statement has single tab + space indentation")
     print("This statement has single tab + space indentation")
```

OUTPUT: This statement has single tab + space indentation
This statement has single tab + space indentation

DATA TYPES AND TYPE CASTINGS:

```
# Data types and type castings
a = 1234
print(type(a))  # <class 'int'>

b = -1234
print(type(b))  # <class 'int'>

c = 0
print(type(c))  # <class 'int'>

g = 1.04
print(type(g))  # <class 'float'>

h = -11.23
print(type(h))  # <class 'float'>

i = 0.34
print(type(i))  # <class 'float'>

j = 2.21e-10
print(type(j))  # <class 'float'>

k = 5E220
print(type(k))  # <class 'float'>
```

OUTPUT: <class 'int'>
<class 'int'>
<class 'int'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>

DATA TYPES AND TYPE CASTINGS- COMPLEX NUMBERS:

```
# Data types and type castings - Complex Numbers
x = complex(1, 2)
print(type(x))  # <class 'complex'>
print(x)        # (1+2j)

z = 1 + 2j
print(type(z))  # <class 'complex'>
print(z)        # (1+2j)
```

OUTPUT: <class 'complex'>
(1+2j)

<class 'complex'>
(1+2j)

DATA TYPES AND TYPE CASTINGS – BOOLEAN:

```
# Data types and type castings - Boolean
x = True
print(type(x))  # <class 'bool'>

y = False
print(type(y))  # <class 'bool'>
```

OUTPUT: <class 'bool'>
<class 'bool'>

STRINGS:

```
#Strings
str1="String"  # Strings start and end with double quotes
print(str1)

str2='String'  # Strings start and end with single quotes
print(str2)

str3="String'  # Strings start with double quote and end with single quote
               # SyntaxError

str4='String"  # Strings start with single quote and end with double quote
               # SyntaxError

str5="Day's"  # Single quote within double quotes
print(str5)

str6='Days"s'  # Double quote within single quotes
```

SPECIAL CHARACTERS:

```
#Special Characters
print("This is a backslash(\\) mark.")
# This is a backslash (\) mark.

print("This is tab \t key")
# This is tab   key.

print("These are \'single quotes\'")
# These are 'single quotes'

print("These are \"double quotes\"")
# These are "double quotes"

print("This is a new line\nNew line")
```

```
# This is a new line.
# New line
```

OUTPUT: This is a backslash(\) mark.
This is tab	key
These are 'single quotes'
These are "double quotes"
This is a new line
New line

STRING INDICES AND ACCESSING STRING ELEMENTS:

```
# Strings indices and accessing string elements
string1 = "PYTHON TUTORIAL"

print(string1[0])    # Print first character (P)
print(string1[-15])  # Print first character (P)
print(string1[-1])   # Print last character (L)
print(string1[14])   # Print last character (L)
print(string1[4])    # Print 4th character (O)
print(string1[-11])  # Print 4th character (O)

# Check index before accessing
if len(string1) > 16:
    print(string1[16])
else:
    print("Index 16 is out of range")  # Safe handling
```

OUTPUT: P
P
L
L
O
O
Index 16 is out of range

LISTS:

```
# Lists
mylist1 = [5, 12, 13, 14]    # The list contains all integer values
print(mylist1)

mylist2 = ['red', 'blue', 'black']    # The list contains all string values
print(mylist2)

mylist3 = ['red', 12, 121.21]    # The list contains a string, an integer, and a float value
print(mylist3)
```

OUTPUT: [5, 12, 13, 14]
['red', 'blue', 'black']
['red', 12, 121.21]

LISTS INDICES:

```
# Lists Indices
mylist = []
print(mylist)
```

OUTPUT: []

COLOR LISTS INDICES:

```
# Color Lists Indices
color_list = ["red", "blue", "green", "black"]  # The list contains four elements starting at
index 0 and ending at 3

# Accessing elements
print(color_list[0], color_list[3])  # Print first and last element: red black
print(color_list[-1])  # Returns and prints last element: black
```

OUTPUT: red black
Black

COLOR LIST INDICES:

```
# Color Lists Indices
color_list = ["red", "blue", "green", "black"]  # The list contains four elements starting at
index 0 and ending at 3

# Accessing elements
print(color_list[0], color_list[3])  # Print first and last element: red black
print(color_list[-1])  # Returns and prints last element: black
print(color_list)    # Creates error as the indices is out of range
```

LISTS SLICE:

```
# Lists Slice
color_list = ["red", "blue", "green", "black"]  # The list contains four elements starting at
index 0 and ending at 3

print(color_list[0:2])  # Cut first two items: ['red', 'blue']
print(color_list[1:2])  # Cut 2nd item: ['blue']
print(color_list[1:-2])  # Cut 2nd item: ['blue']
print(color_list[:3])  # Cut first three items: ['red', 'blue', 'green']
print(color_list[:])  # Creates a copy of the original list: ['red', 'blue', 'green',
'black']
```

OUTPUT: ['red', 'blue']
['blue']
['blue']
['red', 'blue', 'green']
['red', 'blue', 'green', 'black']
