```
#this is hello world print code.
print("Hello, World!")
```

```
Hello, World!
```

```
#python indentation.
if 5 > 2:
  print("Five is greater than two!")
```

```
Five is greater than two!
```

```
#python indentation.
if 5 > 2:
print("Five is greater than two!")
```

```
  File "<ipython-input-8-acf99c81b6b6>", line 3
    print("Five is greater than two!")
    ^
IndentationError: expected an indented block after 'if' statement on line 2
```

SEARCH STACK OVERFLOW

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

```
Hello, World!
```

```
#variables in python.
x=5
y="Hello"

print(x)
print(y)
```

```
5
Hello
```

```
x=4 #x is int type
x="Hello" #x is now str type
print(x)
```

```
Hello
```

```
#specify datatype with casting.
x = str(3)
y = int(3)
z = float(3)
print(x)
print(y)
print(z)
```

```
3
3
3.0
```

```
#find datatype
x = 5
y = "John"
print(type(x))
print(type(y))
```

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

```
#single and double quotes are same.
x = "Sally"
print(x)
```

```
y = 'John'
print(y)
```

```
     Sally
     John
```

```
#case sensitive.
a = 4
A = "Sally"
#A will not overwrite a
print(a)
```

```
     4
```

```
#multiple variable in one line.
x,y,z="Orange","Banana","Cherry"
print(x)
print(y)
print(z)
```

```
     Orange
     Banana
     Cherry
```

```
#same value to multiple variables
x=y=z="Orange"
print(x)
print(y)
print(z)
```

```
     Orange
     Orange
     Orange
```

```
#unpack a list.
fruits = ["apple","banana","cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

```
     apple
     banana
     cherry
```

```
#output variables.
x = "Python is awesome"
print(x)
```

```
     Python iss awesome
```

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

```
     Python is awesome
```

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

```
     Python is awesome
```

```
x = 5
y = 10
print(x+y)
```

```
     15
```

```
"""In the print() function, when you try to combine a string
and a number with the + operator, Python will give you an error
```

```
"""
x = 5
y = "John"
print(x+y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-431a49d41c67> in <cell line: 6>()
      4 x = 5
      5 y = "John"
----> 6 print(x+y)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

SEARCH STACK OVERFLOW

```
x = 5
y = "John"
print(x,y)
```

```
5 John
```

```
#global & local variable
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()
print("Python is " + x)
```

```
Python is fantastic
Python is awesome
```

```
#global keyword
def myfunc():
  global x
  x = "fantastic"
myfunc()
print("Python is " + x)
```

```
Python is fantastic
```

```
x = "awesome"
def myfunc():
  global x
  x = "fantastic"
  print("Python is " + x)
myfunc()
print("Python is " + x)
```

```
Python is fantastic
Python is fantastic
```

```
#convert from one type to another
x = 1 #int
y = 2.5 #float
z = 1j #complex

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```

```
#specify a variable type
x = 1
y = 2.2
z = "3"
print(int(x))
print(int(y))
print(int(z))

print(float(x))
print(float(y))
print(float(z))

x = "s1"
print(str(x))
print(str(y))
print(str(z))
```

```
1
2
3
1.0
2.2
3.0
s1
2.2
3
```

```
a = "Hello"
print(a[1])
```

```
e
```

```
#loop through a string
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

```
#string length
x = "banana"
print(len(x))
```

```
6
```

```
#check string
txt = "The best things in life are free"
print("life" in txt)
print("cost" in txt)
```

```
True
False
```

```
txt = "The best things in life are free"
if "free" in txt:
  print("Yes, 'free' is present.")
else:
  print("No, 'free' is not present")
```

```
No, 'cost' is not present
```

```python
txt = "The best things in life are free"
if "cost" in txt:
  print("Yes, 'cost' is present.")
else:
  print("No, 'cost' is not present")
```

```
No, 'cost' is not present
```

```python
#string slicing
a = "Hello World"
print(a[2:5])
print(a[:5])
print(a[2:])
print(a[-5:-2])
```

```
llo
Hello
llo World
Wor
```

```python
#upper case
a = "Hello, World!"
print(a.upper())
```

```
HELLO, WORLD!
```

```python
#lower case
a = "HELLO, WORLD"
print(a.lower())
```

```
hello, world
```

```python
#remove whitespace
a = " Hello, World! "
print(a.strip())
```

```
Hello, World!
```

```python
#replace
a = "Jello, World"
print(a)
print(a.replace("J","H"))
```

```
Jello, World
Hello, World
```

```python
#split string
a = "Hello, World"
b = a.split(",")
print(b)
```

```
['Hello', ' World']
```

```python
#string concatenation
a = "Hello"
b = "World"
c = a + b
print(c)
c = a + " " + b
print(c)
```

```
HelloWorld
Hello World
```

```python
#string format
age = 24
txt = "My name Sajib, I am " + age
print(txt)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-66-276f34d6d543> in <cell line: 3>()
      1 #string format
      2 age = 24
----> 3 txt = "My name Sajib, I am " + age
      4 print(txt)

TypeError: can only concatenate str (not "int") to str
```

SEARCH STACK OVERFLOW

```
#use format() function
age = 24
txt = "My name is Sajib, I am {}"
print(txt.format(age))
```

```
My name is Sajib, I am 24
```

```
#format() method takes unlimited number of arguments
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity,itemno,price))
```

```
I want 3 pieces of item 567 for 49.95 dollars.
```

```
#format() with index numbers
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity,itemno,price))
```

```
I want to pay 49.95 dollars for 3 pieces of item 567.
```

```
#list
list1 = ["apple","banana","cherry"]
print(list1)
list1[1] = "mango"
print(list1)
```

```
['apple', 'banana', 'cherry']
['apple', 'mango', 'cherry']
```

```
#append items
list1 = ["apple","banana","cherry"]
list1.append("orange")
print(list1)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
#insert
list1 = ["apple","banana","cherry"]
list1.insert(1,"orange")
print(list1)
```

```
['apple', 'orange', 'banana', 'cherry']
```

```
#extend list
list1 = ["apple","banana","cherry"]
list2 = ["orange","mango","papaya"]
list1.extend(list2)
print(list1)
```

```
['apple', 'banana', 'cherry', 'orange', 'mango', 'papaya']
```

```
#add tuple
list1 = ["apple","banana","cherry"]
tuple = ("orange","mango")
list1.extend(tuple)
print(list1)
```

```
['apple', 'banana', 'cherry', 'orange', 'mango']
```

```
#remove item
list1 = ["apple","banana","cherry"]
list1.remove("banana")
print(list1)
```

```
['apple', 'cherry']
```

```
#remove specified index
list1 = ["apple","banana","cherry"]
list1.pop(1)
print(list1)
```

```
['apple', 'cherry']
```

```
#delete entire list
list1 = ["apple","banana","cherry"]
del list1
```

```
#clear the list contents
list1 = ["apple","banana","cherry"]
list1.clear()
print(list1)
```

```
[]
```

```
#Sort the list alphabetically:
#ascending
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

```
#Sort the list numerically:
list1 = [100, 50, 65, 82, 23]
list1.sort()
print(list1)
```

```
[23, 50, 65, 82, 100]
```

```
#descending
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse=True)
print(thislist)
```

```
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

```
list1 = [100, 50, 65, 82, 23]
list1.sort(reverse=True)
print(list1)
```

```
[100, 82, 65, 50, 23]
```

```
#reverse
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.reverse()
print(thislist)
```

```
['banana', 'pineapple', 'kiwi', 'mango', 'orange']
```

```
#copy
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
mylist=thislist.copy()
print(mylist)
```

```
['orange', 'mango', 'kiwi', 'pineapple', 'banana']
```

```
#Make a copy of a list with the list() method:
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
mylist=list(thislist)
print(mylist)
```

```
['orange', 'mango', 'kiwi', 'pineapple', 'banana']
```

```
#tuple
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

```
#set
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'cherry', 'apple', 'banana'}
```

```
#dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

```
#adding items
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

```
#Nested Dictionaries
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
print(myfamily)
```

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', 'year': 2011}}
```

```
import numpy as np
print(np.__version__)
```

```
1.22.4
```

```
arr = np.array([1,2,3,4])
print(arr)
print(type(arr))
print(arr.ndim)
```

```
[1 2 3 4]
<class 'numpy.ndarray'>
1
```

✓ 0s completed at 21:01