

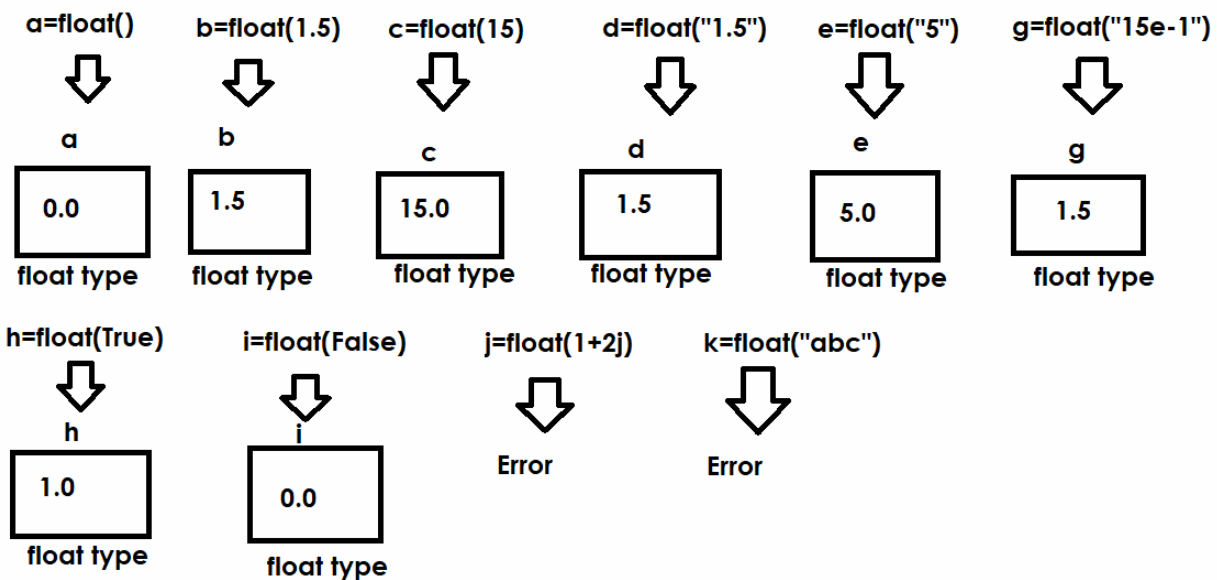
float() function

float() is a predefined function in python

This function is used to perform the following conversions

1. Float to float
2. Int to float
3. String to float
4. Bool to float

Syntax: float(value=0.0)



Example:

```
a=float()
b=float(1.5)
c=float(15)
d=float("1.5")
e=float("1.5e1")
f=float("15e-1")
g=float("15")
h=float(True)
i=float(False)
print(a,b,c,d,e,f,g,h,i,sep="\n")
#j=float("ab") Error
```

Output

0.0
1.5
15.0
1.5
15.0
1.5
15.0
1.0
0.0

Example

Write a program to find area of triangle
$\text{area} = 0.5 * \text{base} * \text{height}$

```
#input
base=float(input("Input Base of Triangle :"))
height=float(input("Input Height of Triangle :"))
```

```
#Process
area=0.5*base*height
```

```
#Output
print("Area of triangle is ",area)
```

Output

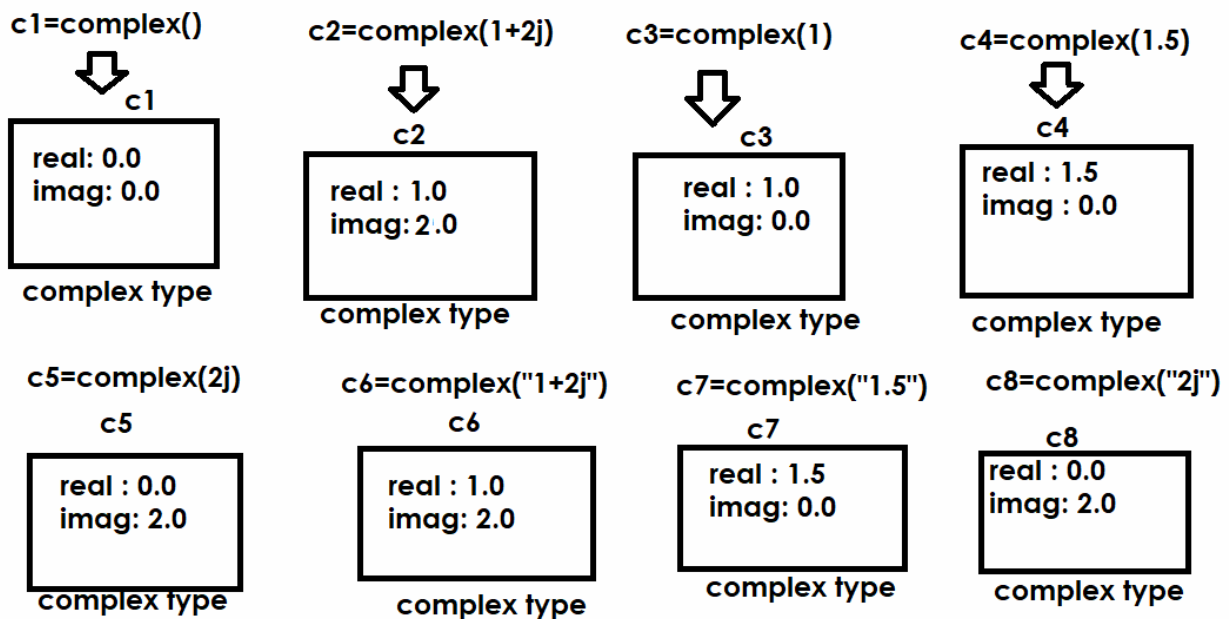
Input Base of Triangle :1.2
Input Height of Triangle :1.3
Area of triangle is 0.78

complex() function

`complex()` is a predefined function in python. This function is used to perform the following conversions

1. Complex to complex
2. Int to complex
3. Float to complex
4. Boolean to complex
5. String to complex

Syntax: <variable-name>=complex(value)



Example:

```
c1=complex()
c2=complex(1+2j)
c3=complex("1+2j")
c4=complex("2j")
c5=complex("1")
c6=complex(1)
c7=complex(1.5)
c8=complex(True)
c9=complex(False)
print(c1,c2,c3,c4,c5,c6,c7,c8,c9,sep="\n")
#c10=complex("ab")
```

Output

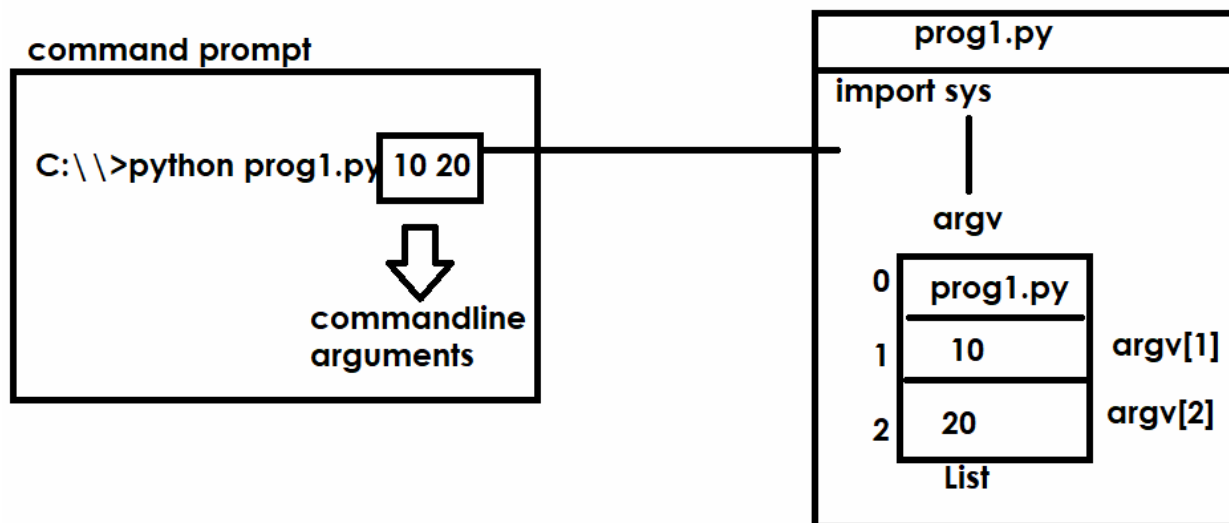
0j
(1+2j)
(1+2j)
2j
(1+0j)
(1+0j)
(1.5+0j)
(1+0j)
0j

Command line arguments

What are command line arguments?

The arguments or values send from command prompt to program are called command line arguments

Command line arguments are of type string



What is argv?

“argv” is a predefined variable exist in “sys” module

Command line arguments are stored inside argv variable.

“argv” is variable of type list

Inside this list all values are stored in “string” type.

Example:

Write a program to print sum of two integers

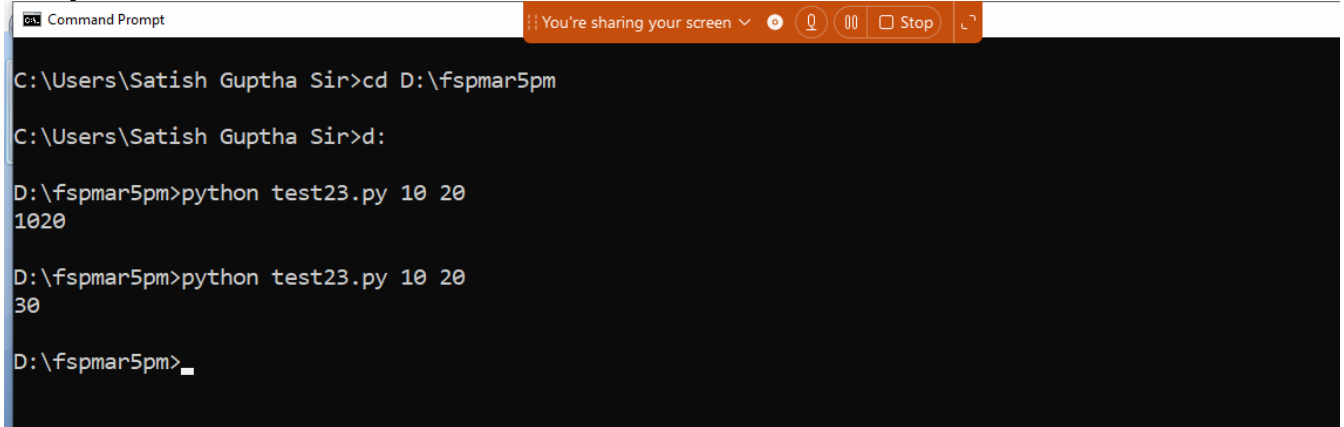
input two integers from command prompt as command

line arguments

```
import sys
```

```
res=int(sys.argv[1])+int(sys.argv[2])  
print(res)
```

Output



The screenshot shows a Windows Command Prompt window with a dark background. The title bar reads "Command Prompt". In the top right corner, there is a status bar with the text "You're sharing your screen" and several icons. The command history shows the user navigating to the directory "D:\fspmar5pm" and running the command "python test23.py 10 20" twice. The first run outputs "1020" and the second run outputs "30".

```
C:\Users\Satish Guptha Sir>cd D:\fspmar5pm  
C:\Users\Satish Guptha Sir>d:  
D:\fspmar5pm>python test23.py 10 20  
1020  
D:\fspmar5pm>python test23.py 10 20  
30  
D:\fspmar5pm>_
```

Example:

Write a program to add two complex numbers
input complex number from command prompt as command line arguments

```
import sys
```

```
c1=complex(sys.argv[1])  
c2=complex(sys.argv[2])  
c3=c1+c2  
print(c1,c2,c3)
```

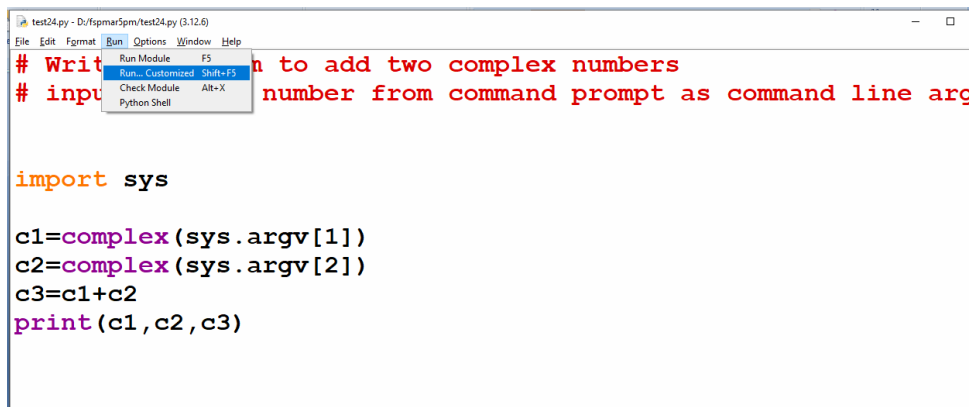


The screenshot shows a Windows Command Prompt window with a dark background. The command history shows the user running the command "python test24.py 1+2j 1+3j" which outputs "(1+2j) (1+3j) (2+5j)". The next command is "python test24.py 1 2" which outputs "(1+0j) (2+0j) (3+0j)".

```
D:\fspmar5pm>python test24.py 1+2j 1+3j  
(1+2j) (1+3j) (2+5j)  
D:\fspmar5pm>python test24.py 1 2  
(1+0j) (2+0j) (3+0j)  
D:\fspmar5pm>_
```

How to given command line arguments from IDLE (OR) how to test program with command line argument using IDLE?

- ⇒ Run
 - Run Customized

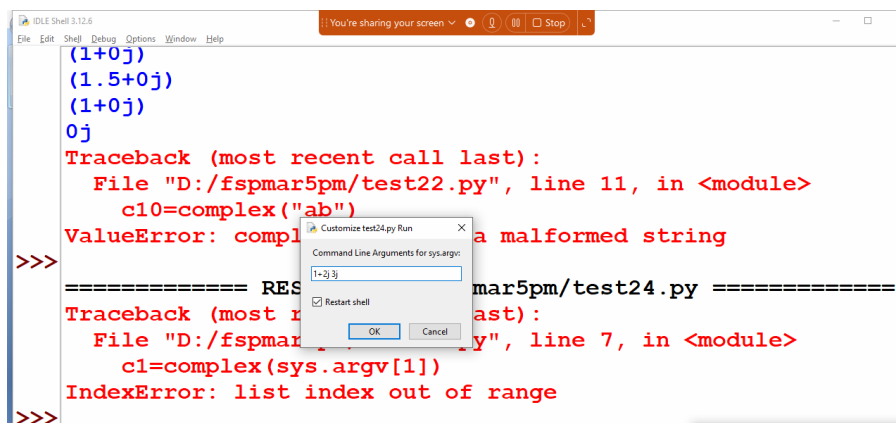


The screenshot shows the IDLE editor window with a file named 'test24.py'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The 'Run' menu is open, showing options: Run Module (F5), Run Customized (Shift+F5), Check Module (Alt+X), and Python Shell. The code in the editor is as follows:

```
# Write a program to add two complex numbers
# input number from command prompt as command line arg

import sys

c1=complex(sys.argv[1])
c2=complex(sys.argv[2])
c3=c1+c2
print(c1,c2,c3)
```



The screenshot shows the IDLE Shell window. It displays the output of the program for three different inputs: (1+0j), (1.5+0j), and (1+0j). The output for the third input is 0j. Below the output, there is a traceback showing a ValueError: complex() argument after * must be a malformed string. A dialog box titled 'Customize test24.py Run' is open, showing the command line arguments for sys.argv as '1+0j'. The dialog box has 'Restart shell' checked and 'OK' and 'Cancel' buttons. The shell also shows a traceback for a second error: IndexError: list index out of range, occurring at line 7 in the module.

bool()

It is a predefined function in python.

This function is used to perform the following conversions

1. Bool to bool
2. Int to bool
3. Float to bool
4. String to bool

Syntax: <variable-name>=bool(value)

a=bool()



a

False

bool type

b=bool(True)



b

True

bool type

c=bool(1)



c

True

bool type

d=bool(False)



d

False

bool

e=bool(0)



e

False

bool type

f=bool(500)



f

True

g=bool(1.5)

g

True

bool type

h=bool(0.0)

h

False

bool type

i=bool(1+2j)

i

True

bool type

j=bool("A")

j

True

bool type

k=bool("")

k

False

bool type

l=bool("False")

l

True

bool type