Type Casting in Python

Type casting refers to converting the value of one data type to another. It is useful when working with operations involving multiple data types, or when you need to manipulate a value in a specific format (like converting a number to a string for display).

```
a=10  #int
b=29.5  #float
c=5j   #complex
print(a,b,c)
print(type(a),type(b),type(c))
```

Converting:

```
d=float(a)
e=complex(a)
print(d,e)
print(type(d),type(e))
```

```
d=int(b)
e=complex(b)
print(d,e)
print(type(d),type(e))
```

```
e=str(c)
print(e)
print(type(e))
```

```
x=587
s=str(x)
print(type(s))
```

Operators in Python

1. ***Arithmetic***

# arithmetic operators(+)

```
a=10
b=20
c=a+b
print(c)
```

# arithmetic operators(-)

```
a=10
b=20
```

```
c=a-b

print(c)
```

```
# # arithmetic operators(*)

a=10

b=20

c=a*b

print(c)
```

```
# # arithmetic operators(/)

a=10

b=20

c=a/b

print(c)
```

```
# # arithmetic operators(//)

a=10

b=20

c=a//b

print(c)
```

```
# # arithmetic operators()

a=10

b=20

c=a**b

print(c)
```

```
# # # arithmetic operators(remainder)

a=10

b=2

c=a%b

print(c)
```

2. ***Compression***

```
# # comparison operators(==)

a=10

b=20

c=a==b

print(c)
```

```
# # comparison operators(!=)

a=10

b=20

c=a!=b

print(c)
```

```
# # comparison operators(>)

a=10

b=20

c=a>b
```

```
print(c)
```

```
# # comparison operators(<)
a=10
b=20
c=a<b
print(c)
```

```
# # comparison operators(>=)
a=10
b=20
c=a>=b
print(c)
```

```
# # comparison operators(<=)
a=10
b=20
c=a<=b
print(c)
```

```
# # comparison operators(==)
a=10
b=20
c=a==b
print(c)
```

3. ***Assignment***

```
# # assignment operators(=)
a=10
a=2
print(a)
```

```
# # assignment operators(+=)
a=10
a+=2
print(a)
```

```
# # assignment operators(-=)
a=10
a-=2
print(a)
```

```
# # assignment operators(*=)
a=10
a*=2
print(a)
```

```
# # assignment operators(/=)
a=10
a/=2
print(a)
```

```
# # assignment operators(//=)
a=10
a//=2
print(a)
```

4. Membership

```
# membership operators(in)
a=[1,2,3,4,5,6,7]
b=5
if b in a:
    print("exist")
else:
    print("sorry")

# # membership operators(not in)
a=[1,2,3,4,5,6,7]
b=50
if b not in a:
    print("right")
else:
    print("wrong")

# # identity operators(is)
a=10
b=10
c= a is b
print(c)

# # identity operators(is not)
a=10
b=10
c= a is not b
print(c)
```

Data Structure:

List Methods in Python

A **list** is a collection of ordered, mutable (changeable) elements in Python. It can store values of any data type and provides several built-in methods to manipulate its contents.

```
# list methods
# # # # # # # # # # # # # # # append
a=[12,67,34,90,5]
a.append(33)
print(a)
```

```
# # # # # # # # # # # # # extend
a=[12,67,34,90,5]
b=[7,8,9]
a.extend(b)
print(a)
# # # # # # # # # # # insert
a=[12,67,34,90,5]
a.insert(2,77)
print(a)


# # # # # # # # # # # # # pop
a=[12,67,34,90,5]
a.pop(1)
print(a)


# # # # # # # # # # # # remove.
a=[12,67,34,90,5]
a.remove(90)
print(a)


# # # # # # # # # # # # # # # # #clear
a=[12,67,34,90,5]
a.clear()
print(a)


# # # # # # # # # # # sort
a=[12,67,34,90,5,8,12,93,1,2,77,3,1]
a.sort()
print(a)
# # # # # # # # # # # replace indexing
a=["z","a","t","y","d","q"]
a.sort()
print(a)
# # # # # # # # # # # # # # #reverse
a=[12,67,34,90,5,8,12,93,1,2,77,3,1]
a.reverse()
print(a)


# # # # # # # #copy
a=[12,67,34,90,5,8,12,93,1,2,77,3,1]
b=a.copy()
print(b)


# # # # # # #count
a=[1,2,8,9,1,1,1,1,1,1,1,1,1,1]
print(a.count(1))
# # # # # # # # # #index
a=[12,67,34,90,90,5]
print(a.index(90))
# # # # # max
a=[1,89,67,56,4,2,67890,98,34]
print(max(a))
# # # # # min
a=[1,89,67,56,4,2,67890,98,34]
print(min(a))
```

a=[1,89,67,56,4,2,67890,98,34]

a[7]=23

print(a)

<u>Dictionary Methods in Python</u>

A **dictionary** in Python is a collection of key-value pairs that is unordered, mutable, and indexed. It allows you to quickly access, update, and manipulate data using keys.

A={key_name:”value”}

<u># # # # # # # # # # clear</u>

a={1:"abc",2:"xyz"}

a.clear()

print(a)

<u># # # # # # # # # # # pop</u>

a={1:"xyz",2:"abc"}

a.pop(1)

print(a)

<u># # # # # # # # # # # popitem</u>

a={1:"xyz",2:"abc",3:"ankit"}

a.popitem()

print(a)

<u># # # # # # # # # # update</u>

a={1:"abc",2:"xyz"}

a.update({3:"jatin"})

print(a)

<u># # # # # # # # # # copy</u>

a={1:"xyz",2:"abc"}

b=a.copy()

print(b)

<u># # # # # # # # # # # get</u>

a={1:"xyz",2:"abc"}

b=a.get(1)

print(b)

<u># # # # # # # # # items</u>

a={1:"abc",2:"xyz"}

b=a.items()

print(b)

<u># # # # # # keys</u>

a={1:"abc",2:"xyz"}

b=a.keys()

print(b)

<u># # # # # values</u>

```
a={1:"abc",2:"xyz"}
b=a.values()
print(b)
# # # # # # # fromkeys
k="a","b","c"
v=1,2,3,4
c=dict.fromkeys(k,v)
print(c)
```

String Methods in Python

Strings in Python are **immutable** sequences of characters. Python provides many **built-in string methods** to manipulate and process string data efficiently. These methods do not change the original string but return a **new string** instead.

```
# # # # capitalize
a="hello world"
print(a.capitalize())

# # # # # # # # # # # title
a="hello world xyz"
print(a.title())


# # # # # # # # # #  center
a="hello world"
print(a.center(97))


# # # # # # # # # upper
a="heloo world"
print(a.upper())


# # # # # # # # lower
a="HELOO WORLD"
print(a.lower())
```

```
# # # # # # # # # # # # join
a="hello"
b="world"
print(" ".join([a,b]))
# # # # # # # # count
a="hello world"
print(a.count('l'))


# # # # # # # # index
a="hello world"
print(a.index('w'))
# # # # # # # replace
a="hello world"
print(a.replace('l','a'))
# # # # # # # len
a="hello world"
print(len(a))
# # # # # # # split
a="hello world my self aditya     "
b=a.split()
```

```
print(b)

print(type(b))

# 0   1   2   3   4   5   6

# a   b   c   d   e   f   g
```

```
# -7 -6  -5  -4  -3  -2  -1

# # # # # # # # # # # slicing

a="animesh"

print(a[::-1])
```

<u>Tuple In Python</u>

A **tuple** is an immutable sequence type in Python used to store multiple items in a single variable. Tuples are defined by listing values separated by commas, often enclosed in parentheses for clarity.

```
# # count

a=(36,78,22,11,3,4,7,7,7,7)

print(a.count(7))


# index

a=(36,78,22,11,3,4,7)

print(a.index(11))
```

```
# WE WANT O/P THIS [5,4,3,2,1,6,7,8,9,10]

a=[1,2,3,4,5,6,7,8,9,10]


# # REMOVE ZEROS FROM IT

a = 10023400607210054321000521621002
```

Conditional Statements

# if

```
a = int(input("enter "))

if a == 56:

   print("right")
```

# if-else

```
a = int(input("enter "))

if a == 56:

   print("right")

else:

   print("wrong")
```

# odd or even

```
a = int(input("enter"))

if a % 2 == 0:

   print("even no.")

else:

   print("odd no.")
```

```
a = int(input("enter"))

if a % 2 != 0:

   print("odd no.")

else:

   print("even no.")
```

# elif

```
a = int(input("enter:"))
if a == 1:
    print("monday")
elif a == 2:
    print("tuesday")
elif a == 3:
    print("wednesday")
elif a == 4:
    print("thursday")
elif a == 5:
    print("friday")
elif a == 6:
    print("saturday")
elif a == 7:
    print("sunday")
else:
    print("enter weekdays only")
```

'''***<u>Practice</u>***

<33 fail

33-50 e garde

50-65 d grade

65-75 c grade

75-85 b grade

85-95 a grade

95-100 a+ grade

'''

```
a = int(input("enter:"))

if a <= 33:

    print("fail")

elif a > 33 and a <= 50:

    print("e grade")

elif a > 50 and a <= 65:

    print("d garde")

elif a > 65 and a <= 75:

    print("c garde")

elif a > 75 and a <= 85:

    print("b garde")

elif a > 85 and a <= 95:

    print("a grade")

elif a > 95 and a <= 100:

    print("a+ grade")

else:

    print("wrong no.")
```

# nested-if

```
a = int(input("enter:"))

if a < 33:

    print(f"{a} + 4 = {a + 4} ")

    a += 4

    if a < 33:

        print("fail")

    else:

        print("passed by grace marks")

else:
    print("pass")
```

<u>#</u> <u>Youngest</u>

```
a = int(input("enter the age of 1st person:"))

b = int(input("enter the age of 2nd person:"))

c = int(input("enter the age of 3rd person:"))

d = int(input("enter the age of 4th person:"))


if a < b:
    if a < c:
        if a < d:
            print("a is youngest")


if b < a:
    if b < c:
        if b < d:
            print("b is youngest")


if c < a:
    if c < b:
        if c < d:
            print("c is youngest")


if d < a:
    if d < b:
        if d < c:
            print("d is youngest")
```

<u>LOOP</u>

<u># for loop</u>

```
for i in range(1, 14, 1):
    print(i)


for i in range(-10, 0):
    print(i)


a = int(input("enter:"))
for i in range(1, 11):
    result = a * i
    print(f"{a} x {i} = {result}")
```

<u># sum of numbers</u>

```
a = int(input("enter:"))
sum = 0
for i in range(1, a + 1):
    sum += i
print(sum)
```

<u># factorial</u>

```
a = int(input("enter:"))
fact = 1
for j in range(1, a + 1):
    fact *= j
print(fact)
```

Programming Problems (WAP = Write a Program)

1. WAP to check whether a person is eligible for voting or not. (Accept age from user)
2. WAP to check whether a number entered by the user is odd or even.
3. WAP to check whether a number is divisible by 7 or not.
4. WAP to check whether the last digit of a number (entered by user) is divisible by 3 or not.
5. WAP to check whether a year is a leap year or not.
6. WAP to accept percentage from the user and display the grade according to the following criteria:
    - Marks > 90 → Grade A
    - Marks > 80 and ≤ 90 → Grade B
    - Marks ≥ 60 and ≤ 80 → Grade C
    - Marks < 60 → Grade D
7. WAP to accept a number from 1 to 7 and display the name of the day (1 for Sunday, 2 for Monday, etc.)
8. WAP to accept two numbers and a mathematical operator, and perform the operation accordingly.
   Example:

Enter First Number: 7

Enter Second Number: 8

Enter Operator: +

Result: 15

9. WAP to check whether a number entered by the user is a three-digit number or not.
10. WAP to find the lowest number out of two numbers entered by the user.
11. WAP to find the largest number out of two numbers entered by the user.
12. WAP to check whether a number (entered by user) is divisible by both 2 and 3.
13. WAP to accept the age of 4 people and display the youngest one.
14. WAP to accept the age of 4 people and display the oldest one.
15. WAP to check if a character is a vowel or not.