



# COMPUTING FUNDAMENTALS USING PYTHON

---

**Ms. Archana A**  
**Assistant Professor**

Department of Computer Applications

# COMPUTING FUNDAMENTALS USING PYTHON

---

## Python Basics Conversion

**Archana A**

Computer Applications

# COMPUTING FUNDAMENTALS USING PYTHON

## Type Conversion

---

- In **Implicit type conversion**, Python automatically converts one data type to another data type.
- Python promotes conversion of **lower datatype (integer) to higher data type (float) to avoid data loss.**

### Example 1: Converting integer to float

```
num_int = 123
num_flo = 1.23

num_new = num_int + num_flo

print("datatype of num_int:", type(num_int))
print("datatype of num_flo:", type(num_flo))

print("Value of num_new:", num_new)
print("datatype of num_new:", type(num_new))
```

When we run the above program, the output will be:

```
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>

Value of num_new: 124.23
datatype of num_new: <class 'float'>
```

In the output, we can see the data type of num\_int is an integer while the data type of num\_float is a float.

Also, we can see the num\_new has a float data type **because Python always converts smaller data types to larger data types** to **avoid the loss of data**

- Now, let's try adding a string and an integer, and see how Python deals with it
- Example: addition of string(higher) data type with integer (lower) data type

```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str:",type(num_str))

print(num_int+num_str)
```

- When we run the program the output will be:

```
Data type of num_int: <class 'int'>
Data type of num_str: <class 'str'>

Traceback (most recent call last):
  File "python", line 7, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- As we can see from the output, we got **TypeError**. Python is **not able to use Implicit Conversion** in such conditions.
- However, Python **has a solution** for these types of situations which is known as **Explicit Conversion**

- In **Explicit Type Conversion**, users convert the data type of an object to required data type.
- We use the predefined functions like `int()`, `float()`, `str()`, etc to perform explicit type conversion.
- This type of conversion is also called **typecasting** because the user casts (changes) the data type of the objects.



- Now, again we will try the same example addition of string and integer using explicit type casting.

### Example 3: Addition of string and integer using explicit conversion

```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))

num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str

print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

- When we run the program the output will be:

```
Data type of num_int: <class 'int'>
Data type of num_str before Type Casting: <class 'str'>

Data type of num_str after Type Casting: <class 'int'>

Sum of num_int and num_str: 579
Data type of the sum: <class 'int'>
```

- Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.
- Explicit casting in python is therefore done using **constructor functions**:
  - `int()`
  - `float()`
  - `str()`.

- **int()**
  - Constructs an integer number from
    - An integer literal
    - A float literal (by rounding down to the previous whole number) literal
    - String literal (providing the string represents a whole number)

```
>>> a=int(10)
```

```
>>> a
```

```
10
```

```
>>> a=int(19.99)
```

```
>>> a
```

```
19
```

```
>>> a=int('123')
```

```
>>> a
```

```
123
```

```
>>> a=int('A')
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#21>", line 1, in <module>
```

```
    a=int('A')
```

```
ValueError: invalid literal for int() with base 10: 'A'
```

### int() – other type to int

Try the following and record the output

- `int(10.56)`-float to int
- `int(10+5j)??` – complex to int
- `int(True)` = bool to int
- `int('15')`, `int('15.5')`, `int("0b111")` – only base10

- **float()**
  - Constructs a float number from
    - An integer literal
    - A float literal
    - A string literal (providing the string represents a float or an integer)

# COMPUTING FUNDAMENTALS USING PYTHON

## Type Conversion –float() example



```
>>> a=float(10)
>>> a
10.0
>>> a=float(14.56)
>>> a
14.56
>>> a=float('10')
>>> a
10.0
>>> a=float('12.34')
>>> a
12.34
>>> a=float('A')
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    a=float('A')
ValueError: could not convert string to float: 'A'
```



### float() – other type to float

#### Convert the following to float type:

- float(5), float(0b111), float(0xbe)
- float(5+2j)
- float(True)
- float("10"), float("10.6"), float("0xbe") – only base 10
- float("hi")

### complex() – other type to complex

- `complex(10)`
- `complex(0b11)`
- `complex(10.5)`
- `complex(True)`, `complex(False)`
- `complex('10')`, `complex('10.5')`-only base 10
- `complex(10,5)`, `complex(10.5,6)`, `complex(5.7,6.3)`
- `complex("5")`, `complex("1", "5")`, `complex(2,"3")`

- **str()**
  - Constructs a string from a wide variety of data types  
  
including
    - Strings
    - Integer literals
    - Float literals

```
>>> a=str(1)
```

```
>>> a
'1'
```

```
>>> a=str(12.34)
```

```
>>> a
'12.34'
```

```
>>> a=str('12')
```

```
>>> a
'12'
```

```
>>> a=str('12.34')
```

```
>>> a
'12.34'
```

```
>>> a=str('A')
```

```
>>> a
'A'
```

```
>>> a=str('Hello, Hi')
```

```
>>> a
'Hello, Hi'
```

## Type Conversion or Casting– Key Points

---

### Key Points to Remember

1. Type Conversion is the conversion of object from one data type to another data type.
2. Implicit Type Conversion is automatically performed by the Python interpreter.
3. Python avoids the loss of data in Implicit Type Conversion.
4. Explicit Type Conversion is also called Type Casting, the data types of objects are converted using predefined functions by the user.
5. In Type Casting, loss of data may occur as we enforce the object to a specific data type



**THANK YOU**

---

**Ms. Archana A**  
**Assistant Professor**

Department of Computer Science

**archana@pes.edu**

+91 80 6666 3333 Extn 392