Intro to Programming CSE101

TUT-1

TA: DHANANJAY SHARMA

dhananjay23033@iiitd.ac.in

Why Programming?

Python Language

- General Purpose
- High Level
- Dynamically typed
- Object Oriented
- Interpreted

Print()

Printing Strings:

```
text = "Hello, world!"
print(text)  # Output: Hello, world!
```

Print()

Printing Variables and Expressions:

```
x = 10

y = 20

print("The sum of", x, "and", y, "is", x + y)  # Output: The sum of 10 and 20 is 30
```

Print()

Printing Multiple Items with Separator:

```
name = "Alice"
age = 30
print(name, age, sep=', ')  # Output: Alice, 30
```

Print()

Formatted String (f-string):

```
name = "Bob"
age = 25
print("Name: {name}, Age: {age}")  # Output: Name: Bob, Age: 25
```

Print()

Using Format Specifiers:

value = 3.14159

print("Formatted value: {:.2f}".format(value)) # Output: Formatted value: 3.14

Print()

Concatenating Strings:

```
name = "Carol"

age = 28

print("Name: " + name + ", Age: " + str(age))  # Output: Name: Carol, Age: 28
```

Print()

Printing on the Same Line:

```
print("Hello", end=' ')
print("World!")  # Output: Hello World!
```

Print()

Printing on the Same Line:

```
print("Hello", end=' ')
print("World!")  # Output: Hello World!
```

Print()

Printing with Escape Characters:

```
print("Newline\nTab\tBackslash\\")    # Output: Newline
                                      #         Tab    Backslash\
```

Print()

Printing Without Line Break:

```
print("Line 1", end=' ')
print("Line 2")  # Output: Line 1 Line 2
```

Algorithm

⇒ A well-defined and systematic approach to solving a particular problem or performing a specific task.

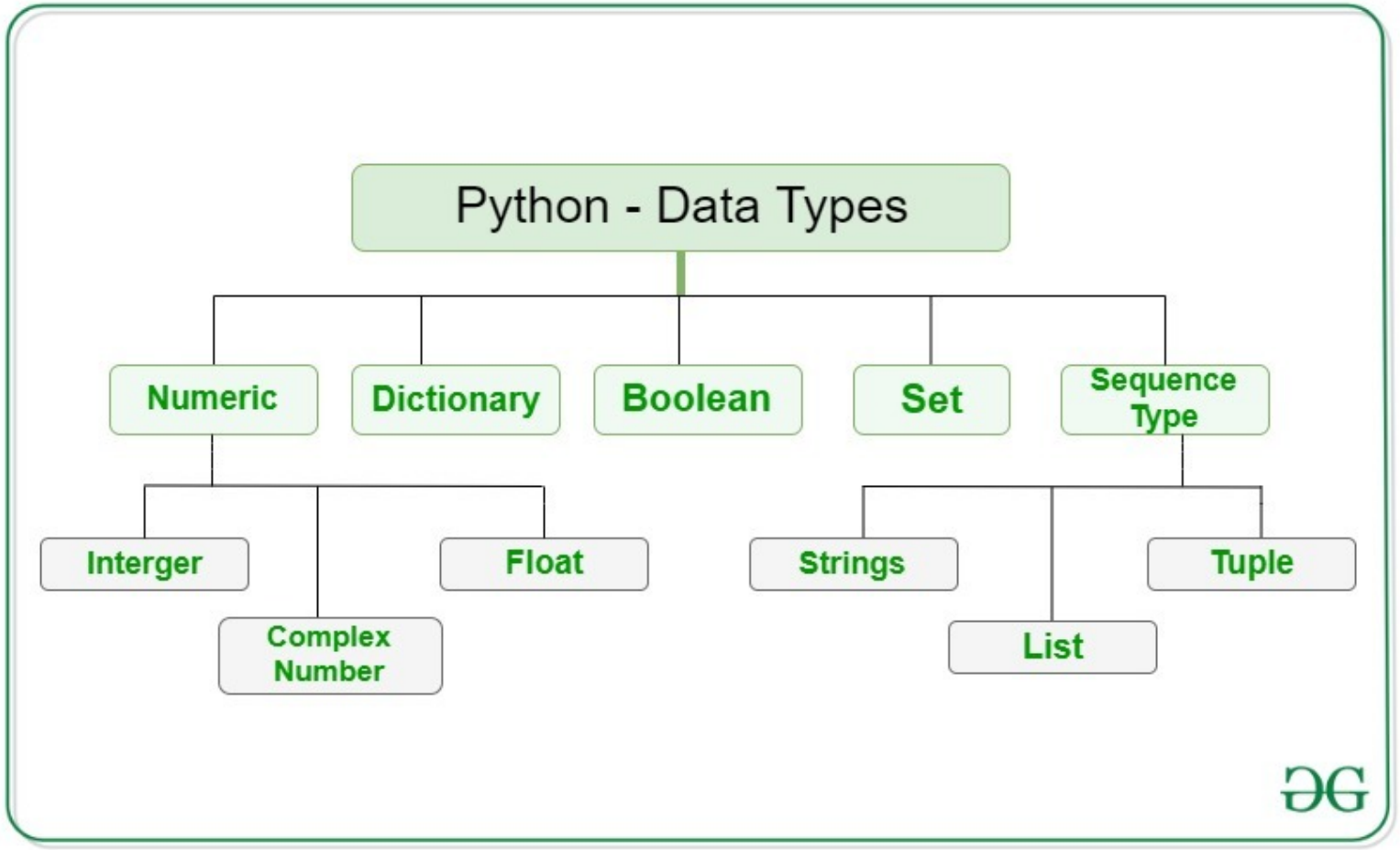
Objects

- ⇒ Objects are core things that python manipulate
- ⇒ Objects can be either scalar or non scalar
- ⇒ Scalar objects are indivisible or atomic: int, float, bool, none
- ⇒ Non scalar for example strings have internal structure
- ⇒ ‘a = 10’
  - ⇒ a is a variable that serves as a reference to an object.
  - ⇒ The object it references is an instance of the int class with the value 10.

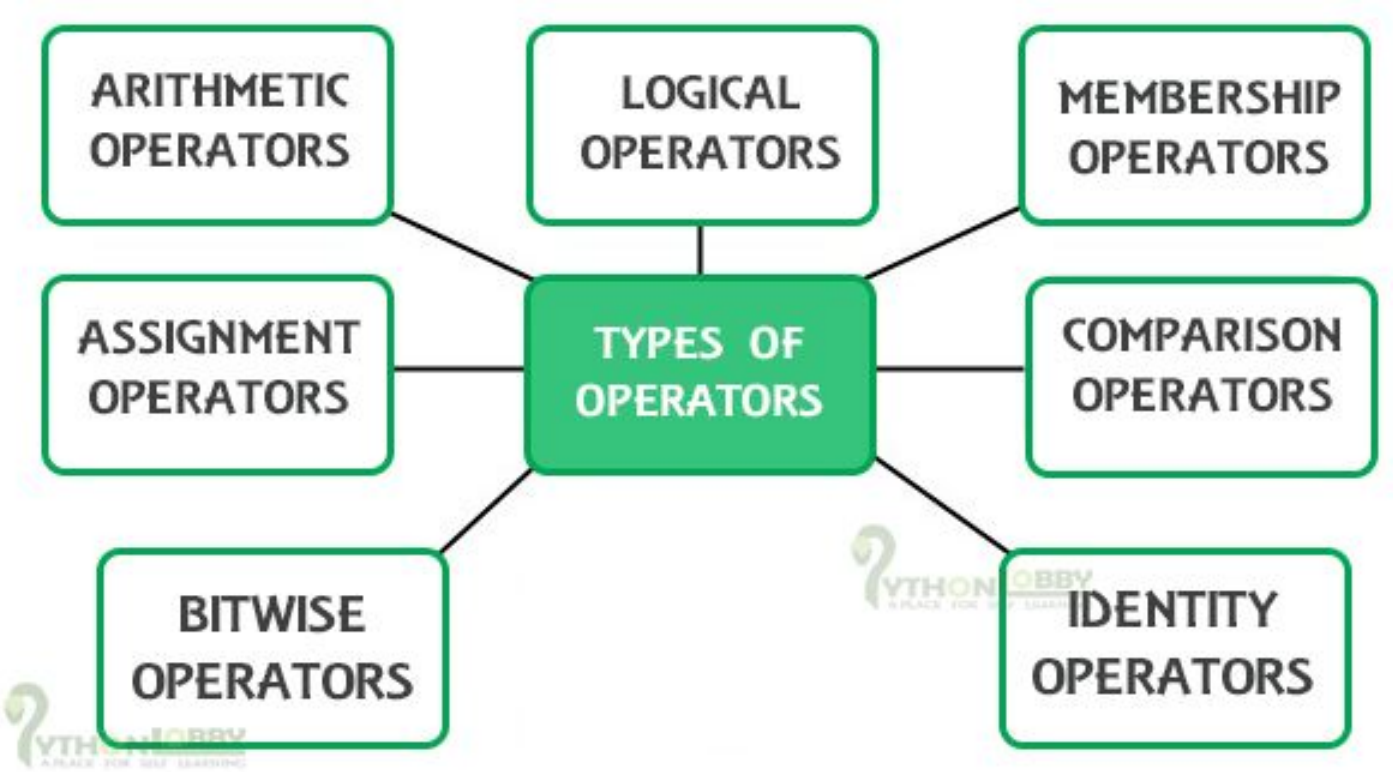
Variables & Data Type

- ⇒ Variable are just a name given to a memory block
- ⇒ Variables are used to store data values.
- ⇒ In Python, you don't need to declare the data type explicitly; Python infers it based on the assigned value.
- ⇒ Data types defines the type of values that variables can hold.

Variables & Data Type



Operators


TYPES OF OPERATORS
ARITHMETIC OPERATORS
LOGICAL OPERATORS
MEMBERSHIP OPERATORS
ASSIGNMENT OPERATORS
COMPARISON OPERATORS
BITWISE OPERATORS
IDENTITY OPERATORS

Operators

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Memory Optimisation

```
a = 10

b = 10

if a is b:

    print("a and b are referring to same memory")

else :

 print("a and b are referring to different memory")
```

Consider the problem of finding the smallest of 4 numbers, viz. minT = min(T1, T2, T3, T4).

```
def MinTime(T1, T2, T3, T4):
      minT= T1;
    if (T2 < minT):  minT = T2;
    if (T3 < minT): minT = T3;
    if (T4 < minT): minT = T4;
     return(minT)

  nextEventTime = MinTime(65.0, 87.1, 26, 75.0)
  output(nextEventTime)
```

What if there are 6 numbers?

```
def MinTime(T1, T2, T3, T4, T5, T6):
      minT= T1;
    if (T2 < minT):  minT = T2;
     if (T3 < minT): minT = T3;
     if (T4 < minT): minT = T4;
              if (T5 < minT): minT = T5;
              if (T6 < minT): minT = T6;
    return(minT)
  nextEventTime = MinTime(65.0, 87.1, 26, 75.0)
  output(nextEventTime)
```

Can we Generalize for N numbers?

```
def MinTime(*args):
      minT= INF;
 for num in args:
  if num<minT:
    minT=num
 return(minT)
   nextEventTime = MinTime(65.0, 87.1, 26, 75.0)
   output(nextEventTime)
```

Finding Second Minimum

Finding Second Minimum

```
def second_minimum(numbers):
  if len(numbers) < 2:
        return None  # Not enough elements to find second minimum
    min1 = float('inf')  # Initialize first minimum to positive infinity
    min2 = float('inf')  # Initialize second minimum to positive infinity
for num in numbers:
        if num < min1:
            min2 = min1
            min1 = num
        elif num < min2 and num != min1:
            min2 = num
return min2
```

Square Root of x

1. Linear Search

Square Root of x

1. Linear Search

```
def sqrt(num: int, prec: float) -> float:
    guess = 0
    while abs(guess**2 - num) > prec:
        guess += prec
    return guess

print(sqrt(10, 0.01))
```

Is this correct?

Square Root of x

1. Linear Search

```
def sqrt(num: int, prec: float) -> float:
    guess = 0;
    while abs(guess**2 - num) > prec:
        guess += (prec/10);
    return guess;

print(sqrt(8,0.01))
```

Square Root of x

1. Newton Raphson Method

```
def sqrt(num: int, prec: float) -> float:
    guess = num / 2.0  # Starting guess (num/2)
    while abs(guess**2 - num) > prec:
        guess = (guess + num / guess) / 2.0  # Newton-Raphson method
    return guess


print(sqrt(8,0.01))
```

Time Complexity

1. Linear Search:

2. Newton Method:

Binary Search for floor(sqrt(n))

Binary Search for floor(sqrt(n))

```
def sqrtFloor(num: int) -> int:
    left = 0
    right = num
    while left <= right :
        guess = (left + right)//2
        if guess**2 == num: break;
        elif guess**2 < num: left = guess+1;
        else right = guess-1;
    return guess
```

Income Tax Calculation

As per the old tax regime (applicable in FY 2022-23). The INCOME TAX rate applicable to a particular income is as follows :

Tax slabs for AY 2022-23

AMOUNT	***INCOME TAX RATE***
Up to ₹2,50,000	*0%*
₹2,50,001 – ₹5,00,000	*5% above ₹2,50,000*
₹5,00,001 – ₹7,50,000	*10% above ₹5,00,000 + ₹12,500*
₹7,50,001 – ₹10,00,000	*15% above ₹7,50,000 + ₹37,500*
₹10,00,001 – ₹12,50,000	*20% above ₹10,00,000 + ₹75,000*
₹12,50,001 – ₹15,00,000	*25% above ₹12,50,000 + ₹1,25,000*
Above ₹15,00,001	*30% above ₹15,00,000 + ₹1,87,500*

WAP to calculate the total payable income tax.

Income Tax Calculation

```
def calculate(amount, percent):
    return (amount * percent) / 100
def calculate_income_tax(total_income: float) -> float:
  if total_income <= 250000:
      return 0
   elif total_income <= 500000:
      return calculate(total_income -  250000, 5)
   elif total_income <= 750000:
      return calculate(total_income - 500000, 10) + 12500
   elif total_income <= 1000000:
      return calculate(total_income - 750000, 15) + 37500
   elif total_income <= 1250000:
      return calculate(total_income - 1000000, 20) + 75000
```

Income Tax Calculation

```
    elif total_income <= 1500000:
        return calculate(total_income - 1250000, 25) + 125000
    else:
        return calculate(total_income - 1500000, 30) + 187500
if __name__ == '__main__':
    total_income = float(input("What's your \
                    annual income?\n>>> "))
    tax = calculate_income_tax(total_income)
    print(f"Total tax applicable at \
            ₹{total_income} is ₹{tax}")
```