IOTA ACADEMY
Become Job Ready...

INTRODUCTION: PART - 2

Data Types

- Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data.
- Since everything is an object in Python Programming, data types are actually **classes** and variables are **instance (object)** of these classes.

Table 1-2: Common Data Types

Data type	Examples
Integers	-2, -1, 0, 1, 2, 3, 4, 5
Floating-point numbers	-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25
Strings	'a', 'aa', 'aaa', 'Hello!', '11 cats'

Numbers in Python

integers (int)

- In Python, integers are zero, positive or negative whole numbers without a fractional part or decimal part, e.g. 0, 100, -10

In [1]:
```
# int example

x = 10
print(type(x))      # use the type function. To determine the type of value stored in any variable.
```
```
<class 'int'>
```

float

- Float is used to represent real numbers and is written with a decimal point that divides the integer and fractional portions. For example, 97.98, 32.3+e18, -32.54e100 all are floating point numbers.

In [2]:
```
# float example

x = 10.2
y = 10/1

print(type(x))
print(type(y))
```
```
<class 'float'>
<class 'float'>
```

String (str)

- String is a sequence of characters.
- It can be created using single quotes(' '), double quotes(" ") or triple quotes(' ' ' ' ' ' or """ """).
- Whatever goes inside quotes it becomes string whether it be a number, text etc.

In [3]:
```
# Example

name = "IOTA Academy"
msg = 'for learning Data Analytics'
print(name,msg)
```
```
IOTA Academy for learning Data Analytics
```

Standard Input and Output

Input method

- In Python, we use input() function to take input from the user.
- Whatever you enter as input, the input function will read as string means the datatype of variable will be 'str'.

In [4]:
```
# input example 1

x = input("Enter your name: ")

print(type(x))          # str means string
print("Hello",x)

# The input() function waits for the user to type some text on the keyboard and press enter.
```
```
Enter your name: iota
<class 'str'>
Hello iota
```

In [5]:
```
# input example 2 -- enter a number as input and check datatype

x = input("Enter a number: ")

print(type(x))          # str means string
print("Your number is: ",x)

# The input() function waits for the user to type some text on the keyboard and press enter.
```
```
Enter a number: 345675577
<class 'str'>
Your number is:  345675577
```

Output method

In Python, we use **print()** function to display output.

In [6]:
```
a = 10
b = 20

# method 1: using f
print(f'a is {a} and b is {b}')

# method 2: positional arguments
print('a is {0} and b is {1}'.format(a,b))

# method3:  if position is not mentioned
print('a is {} and b is {}'.format(b,a))

# method 4: Keyword arguments
print('a is {v1} and b is {v2}'.format(v1=a, v2=b))

# combination of positional and keyword arguments
print('a is {0}, b is {1} and other is {other}'.format(a,b,other='OTHER'))
```
```
a is 10 and b is 20
a is 10 and b is 20
a is 20 and b is 10
a is 10 and b is 20
a is 10, b is 20 and other is OTHER
```

In [7]:
```
# example

name = input("Enter your name: ")

print(f"Hi {name}, you have successfully logged in.")
```
```
Enter your name: IOTA
Hi IOTA, you have successfully logged in.
```

In [8]:
```
# input and output example

# function multiply two numbers which are taken as user input

# when we take user input. it is always a string and

# we can't apply arithmetic operations to strings so we have to change it to int or float.

x1 = input("Enter a number: ")
print("x1", type(x1))

x1 = int(x1)      #this is called: Tyepcasting
print("x1 type after type casting",type(x1))


x2 = input("Another number: ")
x2= int(x2)
print("x2 after type casting",type(x2))

multiply = x1*x2
print("Multiply is: ",multiply)
```
```
Enter a number: 30
x1 <class 'str'>
x1 type after type casting <class 'int'>
Another number: 20
x2 after type casting <class 'int'>
Multiply is:  600
```

Print function

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.

In [9]:
```
# example

print("iota","academy","indore")  # default separator is single space
```
```
iota academy indore
```

In [10]:
```
# example

print("iota","academy","indore", sep="---")
```
```
iota---academy---indore
```

In [11]:
```
print("iota","academy",sep="--> ")
print("I am next line","value2")  # default end is next line("\n")
```
```
iota--> academy
I am next line value2
```

In [12]:
```
# example

print("iota","academy",sep="--> ",end='\n')
print("I am next line","value2")
```
```
iota--> academy
I am next line value2
```

In [13]:
```
# example

print("IOTA","Academy 3*10",3*10,sep="-->",end=" *** ")
print("I am learning Python")
```
```
IOTA-->Academy 3*10-->30 *** I am learning Python
```

In [14]:
```
for i in range(100):
    print(i,end=' ')
```
```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 4
0 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 7
7 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

Operators

- Operators are special symbols in Python that carry out arithmetic or logical computation.
- The value that the operator operates on is called the operand.

Operator Types:

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Special Operators

Arithmetic Operators

- Used to perform mathematical operations like addition, substraction, multiplication etc.

Table 1-1: Math Operators from Highest to Lowest Precedence

Operator	Operation	Example	Evaluates to...
**	Exponent	2 ** 3	8
%	Modulus/remainder	22 % 8	6
//	Integer division/floored quotient	22 // 8	2
/	Division	22 / 8	2.75
*	Multiplication	3 * 5	15
-	Subtraction	5 - 2	3
+	Addition	2 + 2	4

In [15]:
```
# floor division: returns an integer closest and less than the result after division.

print(7//2)   # answer will be 3 (<3.5 and close to 3.5)
```
```
3
```

In [16]:
```
# five to the power 3

print(5**3)
```
```
125
```

In [17]:
```
# modulus operator returns remainder

print(5%2)
```
```
1
```

BODMAS Rule

- To avoid unexpected results, it is advised that you always use brackets in your calculations.

```
(5 - 1) * ((7 + 1) / (3 - 1))
        ↓
    4 * ((7 + 1) / (3 - 1))
        ↓
    4 * (   8   ) / (3 - 1))
        ↓
    4 * (   8   ) / (   2   )
        ↓
    4 * 4.0
        ↓
      16.0
```

Figure 1-1: Evaluating an expression reduces it to a single value.

Comparison Operators

- Used to compare values.
- **It either returns True or False**, according to the condition.
- **> , < , >= , <= , == , !=** are comparison operators.

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

In [18]:
```
x = 7!=6

print(x)      # comparison operators always returns True or False
```
```
True
```

In [19]:
```
42 == 42
```
Out[19]:
```
True
```

In [20]:
```
50 == 56
```
Out[20]:
```
False
```

In [21]:
```
3 != 4
```
Out[21]:
```
True
```

In [22]:
```
'hello' == 'hello'
```
Out[22]:
```
True
```

In [23]:
```
34 == '34'
```
Out[23]:
```
False
```

Logical Operators

- **and, or** and **not** are logical operators.

a	b	condition 1	condition 2	condition 1 output	condition 2 output	and	or
5	10	a>b	b!=12	FALSE	TRUE	FALSE	TRUE
12	12	a>b	b!=12	FALSE	FALSE	FALSE	FALSE
15	13	a>b	b!=13	TRUE	FALSE	FALSE	TRUE
10	13	a<b	a!=b	TRUE	TRUE	TRUE	TRUE

In [24]:
```
a, b = True, False  # keywords

print(a and b)       # True and False---> False
print(a or b)        # True or False ---> True
print(not a)         # not True ----> False
```
```
False
True
False
```

In [25]:
```
4<3 or 5<1
```
Out[25]:
```
False
```

In [26]:
```
# and

# when left and right both conditions are true then true else false

if 5>6 and 7>=7:
    print("Both are True")

else:
    print("Atleast one is false")
```
```
Atleast one is false
```

Bitwise Operators

Bitwise operators act on operands as if they were strings of binary digits. It operates bit by bit.

& , | , ~ , ^ , >> , << are Bitwise operators

Examples:

In [27]:
```
a, b = 10, 4

# 10 to binary is : 1010: 2^3+2^1
# 4 to binary is : 0100: 2^2

print(a&b)  # 1010 & 0100 --> 0000 --> 0
print(a|b)  # 1010 | 0100 ---> 1110 ---> 14 (8+4+2)
```
```
0
14
```

Assignment Operators

- Used to assign values to the variables.
- a = 5, here "=" is a simple assignment operation. "=" assigns the value 5 (right) to the variable a on the left.
- =, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<= are assignment operators.

In [28]:
```
a = 40
a = a + 20
print(a)
```
```
60
```

In [29]:
```
# same as above

a = 40
a += 20
print(a)
```
```
60
```

In [30]:
```
a = 11
a += 10  # same as a = a+10
print(a)
```
```
21
```

In [31]:
```
a = 11
a = a + 10
print(a)
```
```
21
```

In [32]:
```
a = 10

a /= 20    # a = a/20
print(a)
```
```
0.5
```

In [33]:
```
# same as above
a = 10
a = a/20
print(a)
```
```
0.5
```

Special Operators

- Identity Operators

- **is and is not** are the identity operators.
- Used to check if two values (or variables) are located on the same part of the memory.

In [34]:
```
a = 15
print(id(a))

b = 5
print(id(b))

print(a is b)   # True for only basic data types (numbers and strings)
```
```
1996959607536
1996959607216
False
```

In [35]:
```
#1.1
a = "hemant "  # here is a space so both a and b are different
b = "hemant"

print(id(a))
print(id(b))

print(a is not b)
```
```
1997063879024
1997063879280
True
```

In [36]:
```
print(len(a))
print(len(b))
```
```
7
6
```

In [37]:
```
a = ['hemant',5]
b = ['hemant',5]

print(id(a))
print(id(b))

print(a is b)
```
```
1997063978112
1997063878912
False
```

- Membership Operators

- **in and not in** are the membership operators in Python.
- Used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In [38]:
```
# Example:

lst = [1,2,3,4,5]

print(6 in lst)
```
```
False
```

In [39]:
```
text = 'learning P,oython'

'P,o' in text
```
Out[39]:
```
True
```

Exercise: 2

1) Print the given strings in the specified format.

```
"IOTA" "Academy" "is" "the" "best" "institute."
```

- **Expected Output:**

  IOTA-Academy-is-the-best-institute.

2) Write a program (WAP) that reads two integers, a and b, from the user. Your program should compute and display:

- The sum of a and b
- The difference when a is subtracted from b
- The product of a and b
- The quotient when a is divided by b
- The remainder when a is divided by b
- The result of a to the power b

3) WAP to calculate the area of a circle which radius is given (pi - 3.14 or 22/7).

4) WAP to convert Degree Celsius to Fahrenheit.

5) WAP to calculate the area of a triangle whose sides are given.

ERRORS ARE OKAY!

Programs will crash if they contain code the computer can't understand, which will cause Python to show an error message. An error message won't break your computer, though, so don't be afraid to make mistakes. A *crash* just means the program stopped running unexpectedly.

Great Job!