Language Basics

Agenda

1. Variable
2. Keywords
3. Data types
4. Operators
5. Casting
6. Comments

Variable

Variable

- A **variable** is a memory location where we can store a value.
- A variable's value may change over time.
- An **identifier** is used to refer to this variable (memory location).
- An identifier is a label that names a variable.

```
num = 100
```

- An integer variable whose value is 100, is created with the name (identifier) num.

Rules for naming variables

- Python is **case sensitive** language.
- *Num* and *num* are considered as two different identifiers.
- Variable name can start with uppercase or lowercase alphabets and underscore.
- It cannot start with a digit but it can contain digits in it.
- It cannot contain any special characters.

Valid Identifiers	Invalid Identifiers
NAME, age, Gender, z, _address, last_name, Address2	1data, first@name

Keywords

Keywords

- Keywords are reserved words that has a predefined meaning.
- They perform special functions and cannot be used for naming any variables, methods or classes.

Some important python keywords					
if	else	elif	for	raise	from
while	True	False	None	try	return
break	continue	and	or	except	import
pass	class	in	global	finally	del

Keywords continued..

- Keywords are case sensitive.
- Although most of the keywords are in lowercase, few of them will start with uppercase.
- Example : True, False, None
- In IDE keywords are highlighted in different color.
- Some Python IDEs which are widely used : IDLE, Spyder, PyCharm.

```
Predict the output:

and = 200
print(and)
```

Keywords continued..

```
Predict the output:

num1 = 10
num2 = 5
num1 = (num1 + num2)
print(num1,num2)
```

- How many variables are created here?
- Have we used any keyword?

Data types

Data types

- Data type defines what type of data is stored in a variable.
- In Python we don’t mention the data type explicitly, instead it is declared during the execution time by the interpreter.

Built-in data types	
Mutable	Immutable
➢ list ➢ set ➢ dict (Dictionary)	➢ Number: int, float and complex ➢ str (String) ➢ tuple ➢ bool (Boolean)

Data types continued..

- **Mutable:** Values can be modified in the original location.
- **Immutable:** Values cannot be modified in the original location, any changes done will be stored in a new memory location.
- **type( )** function can be used to get the data type details.
- Everything in Python is an object, we will learn more about class and object later.

```
Program:

num1 = 10
num2 = 5.2
result = False
print(type(num1))
print(type(num2))
print(type(result))
```

```
Output:

<class ‘int’>
<class ‘float’>
<class ‘bool’>
```

Number data types

- They represent immutable numeric values.
- When modifications are done it will be stored in a new memory location.

- **int:** positive or negative integer number (whole number).

  Examples: a = 56, b = -77

- **float:** positive or negative number with decimal places.

  Examples: c = - 5.6, d = 7.890

- **complex:** contains real part and imaginary part (suffixed with letter j).

  Example: num = 2 + 3j

Immutable concept explained:

- **id( )** function returns an unique and constant integer for an object.
- Conceptually it corresponds to the location of an object in the memory.

```
Execute the program and observe the output:

num1 = 10 #int object 10 is created in the memory
print(id(num1),num1) #prints id of 10, 10
num1 = (num1 + 20) #int object 30 is created in the memory
print(id(num1),num1) #prints id of 30, 30
```

Bool

- It represents either True or False.
- Comparing two values result in either True or False.
- When used with mathematical operators such as +, -, * **True represents 1 and False represents 0**.

```
Program:

num1 = 15
num2 = 2
print(num1 > num2)
print(num2 == 0)
print(True + True)
print(True * False)
```

```
Output:

True
False
2
0
```

String (str)

- **str** is the built-in string class.
- str represents one or more sequence of characters.
- It is enclosed in either single or double quotes.
- It can contain alphabets, numbers, special characters and white spaces.

```
Program:

s1 = ‘Wipro’
s2 = “Welcome to Wipro”
s3 = ‘@’
print(type(s1))
print(type(s2))
print(type(s3))
```

```
Output:

class <‘str’>
class <‘str’>
class <‘str’>
```

Tuple

- Tuple is a collection of **immutable** elements.
- It maintains the insertion order and supports index based access.
- It can contain elements of different types.
- Elements are enclosed in parentheses ( ).
- If there is only one element it must be terminated with a comma.

```
Examples:

t_1 = (1, '@', 'wipro', 5.5)
t_2 = (100, 200, 300)
t_3 = ('john',)
```

List

- List is almost similar to array in C/C++/Java.
- It maintains the insertion order and supports index based access.
- It's **mutable**, elements can be added or removed from the original list.
- It can contain elements of different types.
- Elements are enclosed in square brackets [ ].

```
Examples:

li_1 = [1, 2]
li_2 = ['abc', 'xyz']
li_3 = [100, '@', 200, '$']
```

Set

- Set is **unordered** collection of unique elements.
- It doesn’t take duplicates.
- It is **mutable**, elements can be added or removed from the original set.
- The order of the elements are unpredictable.
- It can contain elements of different types.
- Elements are enclosed in curly braces { }.

```
Execute the code and observe the output:

set_1 = {'admin', 500, 'user', 500, 800}
print(set_1)
```

Index based access of elements:

- String, Tuple, List supports positive and negative index based access of elements.
- The way in which elements are accessed is same for all the three data types.

```
s1 = 'Welcome'
```

0	1	2	3	4	5	6
W	e	l	c	o	m	e
-7	-6	-5	-4	-3	-2	-1

```
print(s1[0])  #prints W
print(s1[-4]) #prints c
```

```
li = [1, 45, 77, 99]
```

0	1	2	3
1	45	77	99
-4	-3	-2	-1

```
print(li[-1]) #prints 99
print(li[1])  #prints 45
```

Dictionary (dict)

- Dict is collection of key-value pairs enclosed in curly braces { }.
- **From Python 3.6 dict maintains the insertion order.**
- It is mutable, elements can be added or removed from the original dict.
- **Keys are unique.**
- Each element is written as **key : value** combination.
- Key and value can be of same or different types for every element.

```
Examples:

dict_1 = {100:'arun', 200:'chandu', 'user':'admin'}
dict_2 = {1:'A', 2:'B', 3:'C'}
```

We will be discussing each data type in detail in the later sessions.

Quiz time:

- What is the data type of items?

  items = [1, 23, ‘hello’, 1]

- Which of the following is an invalid variable name?

  a) my_string b) _str c) 1_string d) myData

- What value will be stored in result?

  result = True + False

Answers:

1. list
2. c
3. 1

Operators

Operators

Operators are used to perform operations on one or more variables.

They are classified into:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic operators

- *x = 10*
- *y = 3*

Name	Symbol	Example
Addition	+	x + y → 13
Subtraction	-	x – y → 7
Multiplication	*	x * y → 30
Division	/	x / y → 3.333…
Modulus	%	x % y → 1
Exponentiation	**	x ** y → 1000
Integer division	//	x // y → 3

Assignment operators

Symbol	Example	Similar to	Final result in a
=	a = 100	-	100
+=	a += 5	a = a + 5	105
-=	a -= 10	a = a – 10	90
/=	a /= 3	a = a / 3	33.333..
%=	a %= 2	a = a % 2	0
//=	a //=4	a = a // 4	25
*=	a *= 5	a = a * 5	500
**=	a **= 2	a = a ** 2	10000

Comparison operators

- Comparison operators compares two values and yields True or False.
- *x = 5*
- *y = 7*

Name	Symbol	Example
Equal to	==	x == y → False
Not Equal to	!=	x != y → True
Greater than	>	x > y → False
Lesser than	<	x < y → True
Greater than or equal to	>=	x >= y → False
Lesser than or equal to	<=	x <= y → True

Logical operators

- Logical operators are used to combine two or more conditions.
- They yield True or False.
- *x = 200*

Symbol (keyword)	Explanation	Example
and	Returns True only if all the given conditions are satisfied.	(x<500) **and** (x==0) → False
or	Returns True if any one of the given condition is satisfied.	(x<500) **or** (x==0) → True
not	Reverses the result.	**not** (x==0) → True

Identity operators

- Identity operators checks whether two variables are of the same object i.e referring the same memory location.
- *li_1 = ['@', '$']*
- *li_2 = li_1*
- *li_3 = ['1', '2']*

Symbol (keyword)	Explanation	Example
is	Returns True if both variables are the same object.	li_1 **is** li_2 → True li_1 **is** li_3 → False
is not	Returns True if both variables are not the same object.	li_1 **is not** li_3 → True

Read more about Python's memory management : http://foobarnbaz.com/2012/07/08/understanding-python-variables/

Membership operators

- Membership operators yields True or False based on whether the given object is present or not in the collection of objects.
- *li_1 = ['@', '$']*

Symbol (keyword)	Explanation	Example
in	Returns True if the given object is present in the collection of objects.	'@' **in** li_1 → True '#' **in** li_1 → False
not in	Returns True if the given object is not present in the collection of objects.	'&' **not in** li_1 → True

Bitwise operators

- Bitwise operators are used with binary numbers.
- They take the binary representation of a number and performs bitwise operations.

Name	Symbol	Explanation	Example
AND	&	Result is 0 if any one of the bits is 0. Result is 1 only if both bits are 1.	4 & 7 → 4
OR	|	Result is 1 if any one of the bits is 1. Result is 0 if both bits are 0.	4 | 7 → 7
XOR	^	Result is 1 only when the bits are different. Result is 0 when bits are same.	4 ^ 7 → 3
NOT	~	Inverts all the bits. 0 to 1 and 1 to 0. ~x is equivalent to –x–1	~3 → –4

Bitwise operator working explained:

- Example: 4 & 7 → 4

8 4 2 1

0 1 0 0 → binary representation of 4.

0 1 1 1 → binary representation of 7.

---

0 1 0 0 → binary representation of 4 which is the result.

Quiz

- What is the answer to this expression 22 % 3 is?

  a) 7 b) 1 c) 0 d) 5

- What does ~4 evaluate to?

  a) -5 b) -4 c) -3 d) +3

- What does 3 ^ 4 evaluate to?

  a) 81 b) 12 c) 0.75 d) 7

Answers:

1. b
2. a
3. d

Casting

Casting

- Casting is converting a value from one data type to another data type.
- int( ), float( ) and str( ) constructors are used to perform casting.

```
Examples:

print(int(4.5))  #prints 4, float is converted to int
print(int("65")) #prints 65 as int, string is converted to int

print(float(5))  #prints 5.0, int is converted to float

a = 123
s = str(a)
print(s)   #prints 123 as string, int is converted to string
```

Comments

Comments

- Comments are statements which are not executed.
- They help in understanding the code better.
- Any important note for self or others can be included here.

Type of comment	Example
Single line	#This is comment-1 print(‘welcome’) #This is comment-2
Multi line	‘ ‘ ‘ Hi This is multi line comment-1 Welcome to Wipro ’ ’ ’ “ “ “ Hi This is multi line comment-2 Happy Learning..!! ” ” ”

Thank you