Basic Programming in Python

2. Chapter: Basics and If Statement

Lecturer: Daniel Weinhardt

Summer Term 2024

Slides were created by Nohayr Muhammad Abdelmoneim
Thank you very much for sharing!

Overview

- Simple Arithmetic
- Comments
- Variables: Usage, types, naming
- Assignment
- Printing
- Other Operations and their precedence
- If statement

Simple Arithmetic

Operators	Meaning	Example	Result
+	Addition	4 + 2	6
–	Subtraction	4 – 2	2
*	Multiplication	4 * 2	8
/	Division	4 / 2	2
%	Modulus operator to get remainder in integer division	5 % 2	1
**	Exponent	5**2 = 5^2	25
//	Integer Division/ Floor Division	5//2 -5//2	2 -3

Printing the output

- Executing an arithmetic operation, doesn't necessarily mean to have an output.
- The print() function is used to display outputs.
- You can combine both messages and values/variables in print function.

```
25 + 30
25 + 30 / 6
3**2
print(25 + 30)
print("What is 25 + 30? ", 25 + 30)
print(25 + 30 / 6)
print("25 + 30 / 6 = ", 25 + 30 / 6)
```

```
55
What is 25 + 30?  55
30.0
25 + 30 / 6 =  30.0
```

Comments

Comments are parts of your code that are not executed. They are used to make your code clearer to you or your team members.

```
#This is a comment
print("Hello, world!")
#print("This is another comment")
```

```
Hello, world!
```

```
# a line after a # (hash) is called a single-line comment

'''
lines between 3 ' (3 single-quotes)
are called
multi-line comment
'''

# or you can put a # (hash) in front of each line

# This is another
# way to write
# a multi-line comment
```

Introducing Variables

- Variables are used to store data.
- Unlike other programming languages, you don't need to declare a variable or specify its type.
- Variable assignment: assigning a value to a variable

```
x=2
print("x= ",x)
y=4.56
print("y= ",y)
z="John"
print("z is ",z)
a=b=c=27
print("a= ",a, " b= ",b, ",c= ",c)
l,m,n=4, 7.5, "Jack"
print("l= ",l, ", m= ",m, "and n= ",n)
```

```
x=  2
y=  4.56
z is  John
a=  27  b=  27 ,c=  27
l=  4 , m=  7.5 and n=  Jack
```

Introducing Variables

- All operations can be performed on variables.
- Some operations might not be suitable for certain variable types.

```
x=3
y=5
print(x+y)
m="Hello"
n="World"
print(m+n)
print(x+m)
```

```
8
HelloWorld
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most rece
Cell In[23], line 7
      5 n="World"
      6 print(m+n)
----> 7 print(x+m)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Naming variables

- Rules for variable names:
  - A variable name must start with a letter or the underscore character.
  - A variable name can’t start with a number.
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
  - Variable names are case-sensitive (age, Age and AGE are three different variables).
  - Variable names can’t be Python keywords like: if, print, for, etc.

Naming variables convention

- It’s better to have a descriptive name of your variables. The more variables you have, the harder it is to track its role.
- For multi word variable names, there are common conventions:
  - CamelCase: variableName
  - Pascal Case: VariableName
  - Snake Case: variable_name

Variables types

- A variable type refers to the type of data stored in it.
- Some of the built-in datatypes in Python:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

Examples on variable types

- In Python, you don’t have to explicitly mention datatype.
- Variable type is set automatically when assigning a value.

Type conversion/Casting

- You can get the type of any variable using the method type()
- int(): constructs int value from int, float (by removing decimal points), string (a number without points), etc.
- float(): constructs float value from int, float, string (a number with or without point), etc.
- str(): constructs a string from int, float, string, etc.

```
x=2
print(type(x))
x=str(2)
print(type(x))
x=float("2.5")
print(type(x))
x=int("2")
print(type(x))
x=int(2.5)
print(x, type(x))
x=int("2.5")
print(type(x))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'int'>
2 <class 'int'>
```

```
---------------------------------------
ValueError
Cell In[7], line 11
      9 x=int(2.5)
     10 print(x, type(x))
---> 11 x=int("2.5")
     12 print(type(x))

ValueError: invalid literal for int()
```

Example

Calculate the net salary of an employee given that:

- 10% of the gross salary is deducted for taxes.
- 20% of the gross salary is deducted for health insurance.
- An employee gets 100€ for each child they have.

What are the inputs in this case?

What are the outputs?

Can this be solved using what we have learnt so far?

Example

- Inputs:
  - Gross salary
  - Number of children
- Output:
  - Net Salary
- Procedure:

```
GrossSalary=2000
NumberOfChildren=3

NetSalary=0.7*GrossSalary+100*NumberOfChildren
print("Net salary is: ", NetSalary)
```

```
Net salary is:  1700.0
```

Assignment operators

- Assignment operators provide different ways to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Comparison operators

- Comparison operators compare two values and returns True or False

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

- Logical operators are used to combine two conditional statements.

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

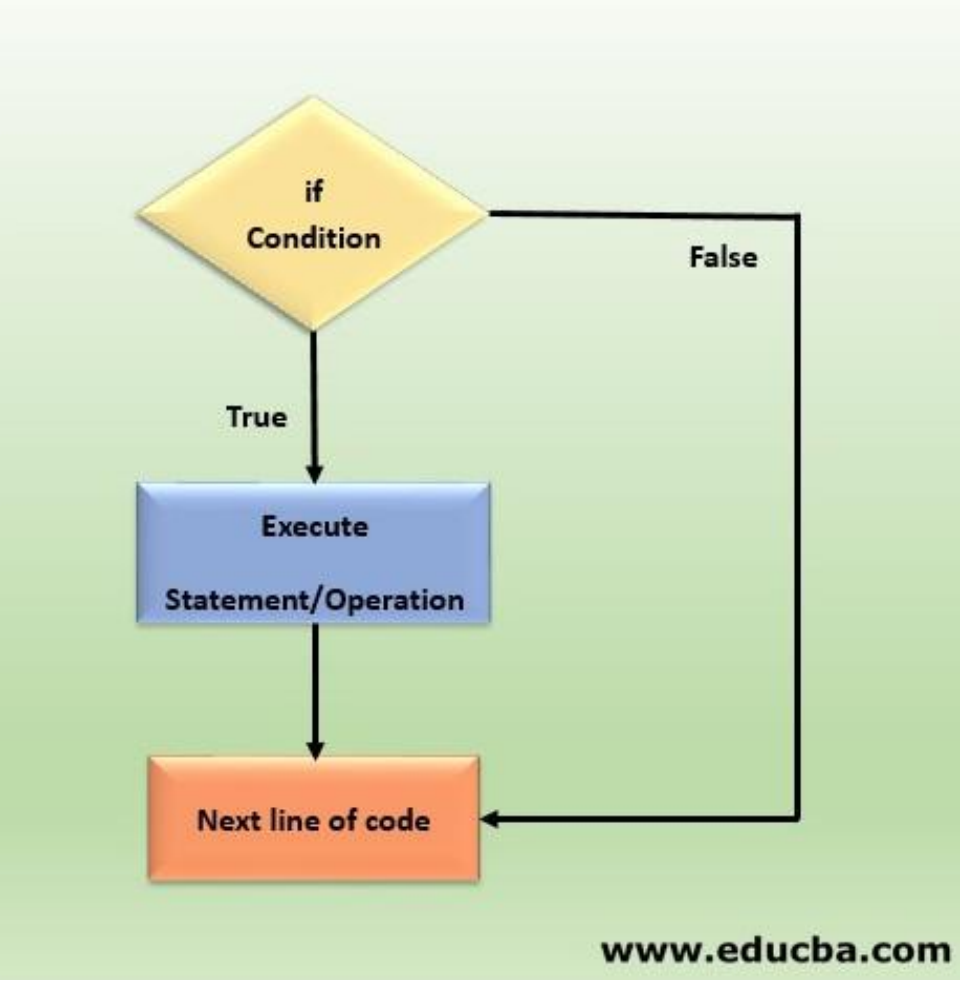
Operators precedence in Python

- Different operators have different priority in execution.

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x  -x  ~x`	Unary plus, unary minus, and bitwise NOT
`*  /  //  %`	Multiplication, division, floor division, and modulus
`+  -`	Addition and subtraction
`<<  >>`	Bitwise left and right shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==  !=  >  >=  <  <=  is  is not  in  not in`	Comparisons, identity, and membership operators
`not`	Logical NOT
`and`	AND
`or`	OR

If statement

- So far, the execution of commands has been sequential.
- All command are executed without exceptions.
- In some cases, we want the execution of command to depend on certain cases or conditions.
- If statements allow us to do so.



Syntax of if statement

```
x=10
if x>0: #This is the condition
    print("If is executed") #If value of condition is true, this is executed.
print("The rest of the code") #This is executed in both cases.
```

```
If is executed
The rest of the code
```

```
x=10
if x<0:
    print("If is executed")
print("The rest of the code")
```

```
The rest of the code
```

```
x=10
if x>0:
    print("If is executed")
    print("We can have multiple lines of code")
print("The rest of the code")
```

```
If is executed
We can have multiple lines of code
The rest of the code
```

If else

- If else is used when we have only two cases, true or false.

```
x=10
if x>0: #This is the condition
    print("If is executed") #If value of condition is true, this is executed.
else:
    print("Else is executed") #If value of condition is false, this is executed.
print("The rest of the code") #This is executed in both cases.
```

```
If is executed
The rest of the code
```

```
x=10
if x<0:
    print("If is executed")
else:
    print("Else is executed")
print("The rest of the code")
```

```
Else is executed
The rest of the code
```

If elif else

- If elif else is used when we have multiple cases
- The first true condition is executed then the rest of the code.

```
x=10
if x<0: #This is the condition
    print("If is executed") #If value of condition is true, this is executed and if terminates.
elif x>2:
    print("First elif is executed")#If the value of condition is true, this is executed and if terminates.
elif x>5:
    print("Second elif is executed") #If value of condition is false, this is executed and if terminates.
print("The rest of the code") #This is executed in any case.
```

```
First elif is executed
The rest of the code
```

```
x=10
if x<0: #This is the condition
    print("If is executed") #If value of condition is true, this is executed and if terminates.
elif x>2 and x<9:
    print("First elif is executed")#If the value of condition is true, this is executed and if terminates.
elif x==10:
    print("Second elif is executed") #If value of condition is false, this is executed and if terminates.
else:
    print("Else is executed") #If all conditions are false, this is executed.
print("The rest of the code") #This is executed in any case.
```

```
Second elif is executed
The rest of the code
```

QUESTIONS?