Information Technology

Year 2020/2021

Introduction to Programming

Programming in Python

What we are going to learn

Values and Variables:

- Integers
- Float
- Strings
- Boolean

Lists[], Tuples(), Sets{}

Conditional Structures:

- IF
- IF / ELSE
- IF / ELIF / ELSE

Cycles:

- FOR using:
  - lists
  - sets
  - tuples
  - range
- While:
  - using “Break”

Functions → “*def*”

Variables

- A Variable is a container that will hold a value
- Each container will have:
  - NAME → how you refer to it
  - TYPE → what type of data values it will contain
- Primitive types:
  - Integer (e.g. 123)
  - Floating Point (e.g. 123.456)
  - String (e.g. **“**This is a text**”** or **‘**this is also a text**’**)
  - Boolean (**True** or **False**)

Creating a Variable

- In Python we create a variable with the assignment operator " **=** "
- The simple command:

  a = 10

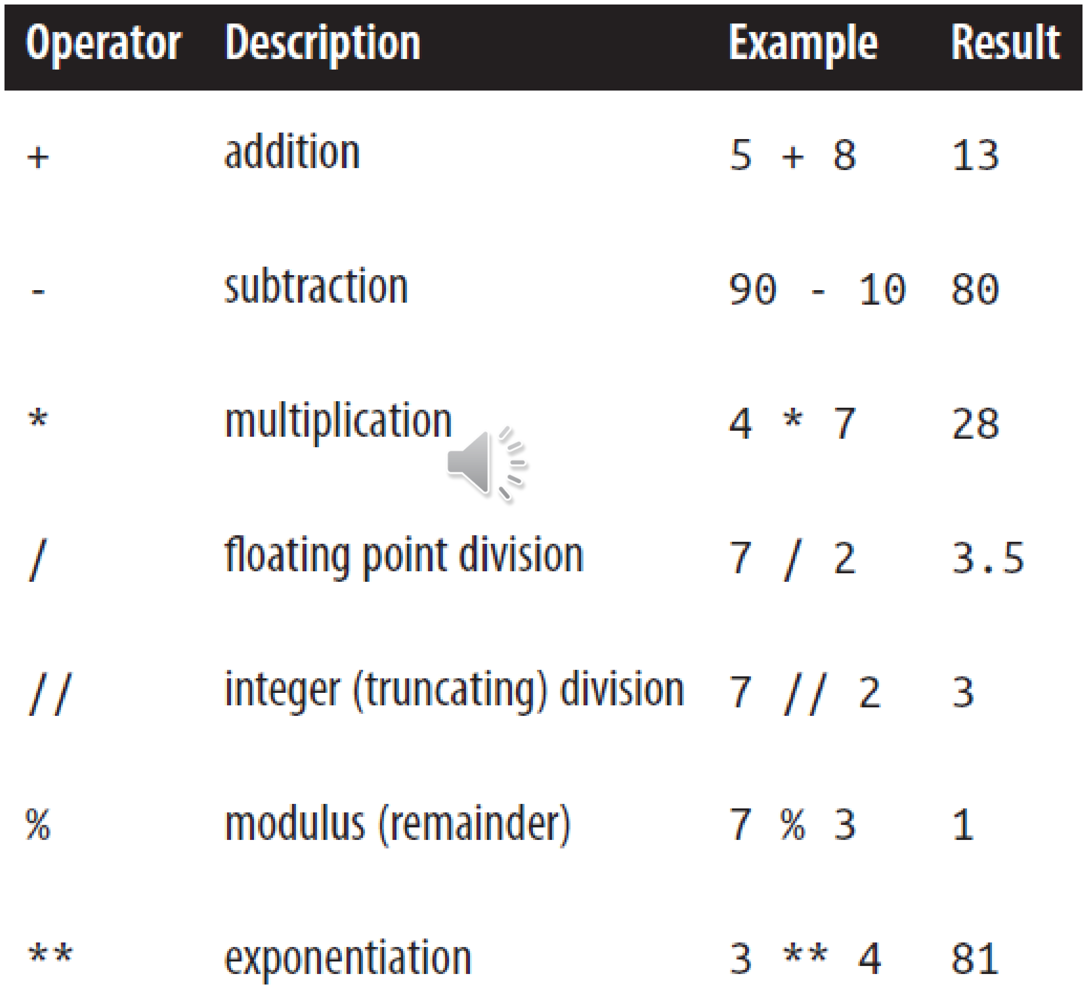
  – Will do the following:
    - Create a container (variable)
    - Label the container with "**a**"
    - The container will have the type "Integer"
    - Put the integer number 10 into the container

  – Likewise, "**b = 12.345**" will create another container, label it "**b**", assign it the type "Floating Point" and put the value 12.345 into it

Types of Variables

- Python recognizes the value assigned to a variable and gives it the correct type
  - a = 10 → a will be type “Int” (Integer)
  - b = 1.123 → b will be type “Float” (Floating Point)
  - c = “This is a text” → c will be type “Str” (String)
  - d = True → d will be type “Bool” (Boolean)

Numeric Operators



Operator	Description	Example	Result
+	addition	5 + 8	13
-	subtraction	90 - 10	80
*	multiplication	4 * 7	28
/	floating point division	7 / 2	3.5
//	integer (truncating) division	7 // 2	3
%	modulus (remainder)	7 % 3	1
**	exponentiation	3 ** 4	81

(Lubanovic, 2014, p. 21)

Comparison Operators

Equality	==
inequality	!=
less than	<
less than or equal	<=
greater than	>
greater than or equal	>=
membership	in

Other Operators

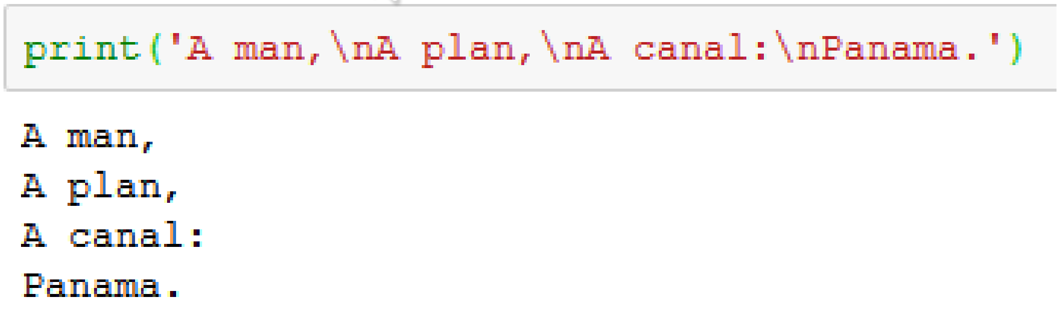
- Logical:
  - and
  - or
  - Not
- Assignment
  - = a = 5
  - += a += 5 ⇔ a = a + 5
  - -= a -=5 ⇔ a = a - 5
  - *= a *= 5 ⇔ a = a * 5
  - /= a /= 5 ⇔ a = a / 5
  - %= a %= 5 ⇔ a = a % 5
  - **= a **= 5 ⇔ a = a ** 5

String

- What is a string?
  - Sequence of characters:
    - “this is a string”
    - ‘this is also a string’
    - “can contain any char like 123 ! * etc.”
  - Printing a string:
    - print(“can contain any char like 123 ! * etc.”)
  - Printing several strings:
    - print("several", 'strings', 'in', "a", "sequence")

Special characters

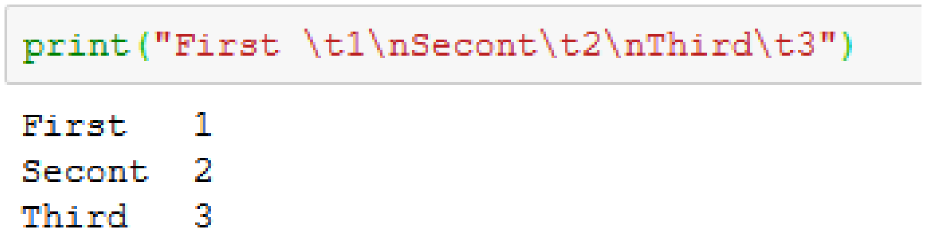
- Escape Character: “\”
  - Means the next character has special meaning
- Some examples:
  - “\n” means “New Line”



```
print('A man,\nA plan,\nA canal:\nPanama.')
```

```
A man,
A plan,
A canal:
Panama.
```

  - “\t” means “Tab”

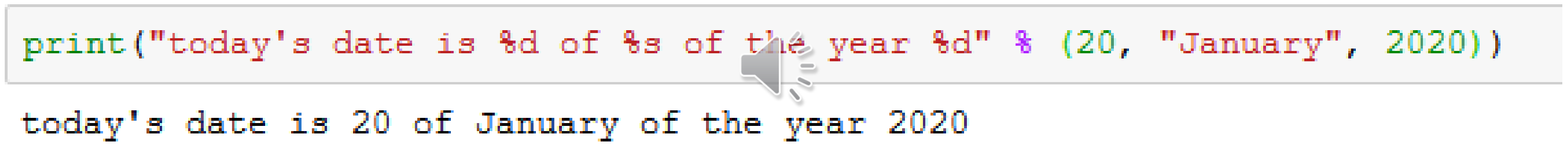


```
print("First \t1\nSecont\t2\nThird\t3")
```

```
First   1
Secont  2
Third   3
```

Special characters (cont)

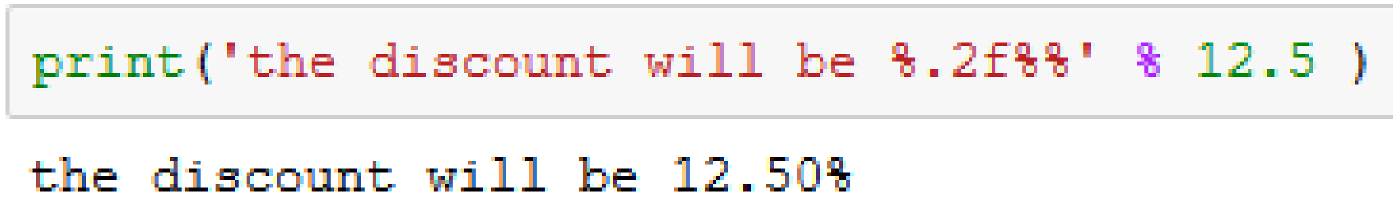
- Substitution Character: “%”
  - When printing, put something where this char is
- Example:



```
print("today's date is %d of %s of the year %d" % (20, "January", 2020))
```

```
today's date is 20 of January of the year 2020
```

- Duplicating a char will remove its special meaning:



```
print('the discount will be %.2f%%' % 12.5 )
```

```
the discount will be 12.50%
```

Special characters (cont)

- Some more examples:

In:	myStr = "Today's date is %d of %s of the year %d" print(myStr % (20, "January", 2020))
Out:	Today's date is 20 of January of the year 2020

In:	myStr = "To print a \\"Backslash\\" we can duplicate the char '\\\\'" print(myStr)
Out:	To print a "Backslash" we can duplicate the char '\\'

Some String functions

In:	myStr = " This is the string to Play With " print("[“ + myStr.lower() + "]“)
Out:	[this is the string to play with]
In:	print("["+myStr.upper()+"]")
Out:	[THIS IS THE STRING TO PLAY WITH]
In:	print("["+myStr.strip()+"]")
Out:	[This is the string to Play With]
In:	print("I've removed %d spaces" % (len(myStr)-len(myStr.strip())))
Out:	I've removed 6 spaces
In:	print("["+myStr.replace("Play", "Work")+"]")
Out:	[This is the string to Work With]
In:	print("["+myStr.replace("Play", "Work").strip()+"]")
Out:	[This is the string to Work With]

More string functions

In:	mySTR = "0123456789abcdefghi"
	print(myStr[9])

Out:	9
In:	print(myStr[10]) #the 10th char, counting from 0 (first position)
Out:	a
In:	print(myStr[3:11]) #from position 3 to position 11 excluding 11th
Out:	3456789a
In:	print(myStr[10:]) # from 10th position onward
Out:	abcdefghi
In:	print(myStr[:10]) # from the beginning til 10th position
Out:	0123456789
In:	print(myStr[-5:]) #the last 5 letters
Out:	efghi