

U

CSIT881

Programming and Data Structures

O

Input & Output (2)

W



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Objectives

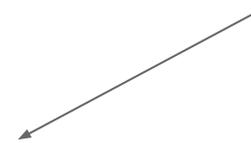
- Multi-line code statement
- Escape sequence
- String format
- Numerical operations

Multi-line code statement

To end a statement in Python, you simply press Enter.
Therefore, this code will generate a syntax error:

```
subject_code = "CSCI111"  
subject_mark = 80  
subject_grade = "D"  
  
result = "Subject result: "  
    + subject_code  
    + " mark " + str(subject_mark)  
    + " grade " + subject_grade  
  
print(result)
```

Python thinks that this is
the end of the statement



Multi-line code statement

Use the backslash `\` to indicate that a statement is continued on the next line.

```
subject_code = "CSCI111"  
subject_mark = 80  
subject_grade = "D"  
  
result = "Subject result: " \  
    + subject_code \  
    + " mark " + str(subject_mark) \  
    + " grade " + subject_grade  
  
print(result)
```

we can break a long line of code into multi-line

Multi-line code statement

Line continuation is automatic when the split comes while a statement is inside parenthesis (, brackets [or braces {

Therefore, this code is fine:

```
subject_code = "CSCI111"  
subject_mark = 80  
subject_grade = "D"  
  
print(  
    "Subject result: "  
    + subject_code  
    + " mark " + str(subject_mark)  
    + " grade " + subject_grade  
)
```

Sometimes, we should break a long line of code into multi-line to make it clearer

Escape sequence

```
print("Welcome to Unimovies!")  
print("Thursday July 30 at 7.15pm: Inside Out")
```

Program output:

```
Welcome to Unimovies!  
Thursday July 30 at 7.15pm: Inside Out
```

Escape sequence

```
print("Welcome to Unimovies!")  
print("Thursday July 30 at 7.15pm: Inside Out")
```

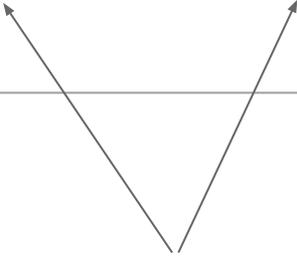
How do we write program for this output:

```
Welcome to Unimovies!  
Thursday July 30 at 7.15pm: "Inside Out"
```

Escape sequence

How about this program?

```
print("Welcome to Unimovies!")  
print("Thursday July 30 at 7.15pm: "Inside Out")
```



what is wrong with this code?

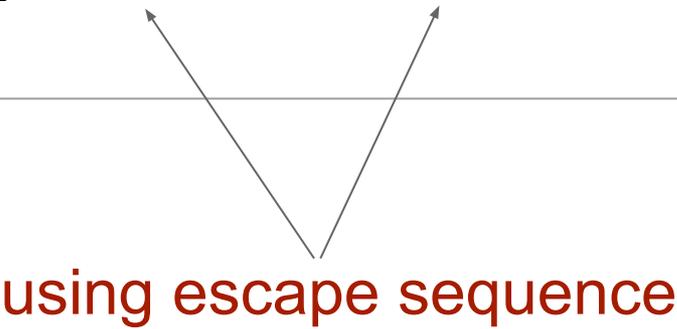
We want to write a program for this output:

```
Welcome to Unimovies!  
Thursday July 30 at 7.15pm: "Inside Out"
```

Escape sequence

The correct program

```
print("Welcome to Unimovies!")  
print("Thursday July 30 at 7.15pm: \"Inside Out\"")
```



using escape sequence

Program output:

```
Welcome to Unimovies!  
Thursday July 30 at 7.15pm: "Inside Out"
```

Escape sequence

Escape Sequence	Meaning
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\n	New line
\t	Tab

Escape sequence

```
print("Your details:\n")
print("\tName: \"John Smith\"")
print("\tSN:    \"2012345\"")
print("\nEnrolment record:\n")
print("\tMATH101")
print("\tCSCI201")
```

Program output:

```
Your details:

    Name: "John Smith"
    SN:   "2012345"

Enrolment record:

    MATH101
    CSCI201
```

Escape sequence

```
print("Escape sequence:")
print("\\n : Insert a newline.")
print("\\t : Insert a tab.")
print("\\\"" : Insert a double quote character.")
print("\\'" : Insert a single quote character.")
print("\\\\" : Insert a backslash character.")
```

What is the output of this program?

String format

```
fname = "John"
lname = "Smith"
age = 20
gpa_score = 3.2

print("Hi {0} {1}!".format(fname, lname))
print("{1} {2} is {0} years old".format(age, fname, lname))
print("His GPA score is {0:.2f}".format(gpa_score))
```

```
print("Hi {0} {1}!".format(fname, lname))
```



```
print("{1} {2} is {0} years old".format(age, fname, lname))
```



```
print("His GPA score is {0:.2f}".format(gpa_score))
```



String format with alignment

```
print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")
```

Program output

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.94
Sodium	Na	11	22.99
Potassium	K	19	39.098
Rubidium	Rb	37	85.468
Caesium	Cs	55	132.905
Francium	Fr	87	223

12345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

left alignment, using 15 spaces

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.94
Sodium	Na	11	22.99
Potassium	K	19	39.098
Rubidium	Rb	37	85.468
Caesium	Cs	55	132.905
Francium	Fr	87	223

123456789012345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

left alignment, using 10 spaces

```

Alkali metals:

Element          Symbol          Atomic number    Atomic weight
Lithium          Li              3                6.94
Sodium           Na              11               22.99
Potassium        K               19               39.098
Rubidium         Rb              37               85.468
Caesium          Cs              55               132.905
Francium         Fr              87                223

```

123456789012345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

center alignment, using 25 spaces

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.94
Sodium	Na	11	22.99
Potassium	K	19	39.098
Rubidium	Rb	37	85.468
Caesium	Cs	55	132.905
Francium	Fr	87	223

123456789012345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

right alignment, using 15 spaces

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.94
Sodium	Na	11	22.99
Potassium	K	19	39.098
Rubidium	Rb	37	85.468
Caesium	Cs	55	132.905
Francium	Fr	87	223

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345

<	left
>	right
^	center

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15.3f}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

we would like to display the Atomic Weight
as having exactly **3 digits** after the decimal mark

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.940
Sodium	Na	11	22.990
Potassium	K	19	39.098
Rubidium	Rb	37	85.468
Caesium	Cs	55	132.905
Francium	Fr	87	223.000

123456789012345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15.4f}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	6.9400
Sodium	Na	11	22.9900
Potassium	K	19	39.0980
Rubidium	Rb	37	85.4680
Caesium	Cs	55	132.9050
Francium	Fr	87	223.0000

123456789012345678901234567890123456789012345678901234567890123456789012345

```

print("Alkali metals:")
print()
print("{0:<15}{1:<10}{2:^25}{3:>15}".format("Element", "Symbol", "Atomic number", "Atomic weight"))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Lithium", "Li", 3, 6.94))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Sodium", "Na", 11, 22.990))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Potassium", "K", 19, 39.098))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Rubidium", "Rb", 37, 85.468))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Caesium", "Cs", 55, 132.905))
print("{0:<15}{1:<10}{2:^25}{3:>15.0f}".format("Francium", "Fr", 87, 223))
print()
print("123456789012345678901234567890123456789012345678901234567890123456789012345")

```

Alkali metals:

Element	Symbol	Atomic number	Atomic weight
Lithium	Li	3	7
Sodium	Na	11	23
Potassium	K	19	39
Rubidium	Rb	37	85
Caesium	Cs	55	133
Francium	Fr	87	223

123456789012345678901234567890123456789012345678901234567890123456789012345

String format with **alignment**

```
print("{0} x {1} = {2}".format(1, 5, 1*5))
print("{0} x {1} = {2}".format(2, 5, 2*5))
print("{0} x {1} = {2}".format(3, 5, 3*5))
print("{0} x {1} = {2}".format(4, 5, 4*5))
print("{0} x {1} = {2}".format(5, 5, 5*5))
print("{0} x {1} = {2}".format(6, 5, 6*5))
print("{0} x {1} = {2}".format(7, 5, 7*5))
print("{0} x {1} = {2}".format(8, 5, 8*5))
print("{0} x {1} = {2}".format(9, 5, 9*5))
print("{0} x {1} = {2}".format(10, 5, 10*5))
```

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
```

String format with **alignment**

```
print("{0:>2} x {1:>1} = {2:>2}".format(1, 5, 1*5))
print("{0:>2} x {1:>1} = {2:>2}".format(2, 5, 2*5))
print("{0:>2} x {1:>1} = {2:>2}".format(3, 5, 3*5))
print("{0:>2} x {1:>1} = {2:>2}".format(4, 5, 4*5))
print("{0:>2} x {1:>1} = {2:>2}".format(5, 5, 5*5))
print("{0:>2} x {1:>1} = {2:>2}".format(6, 5, 6*5))
print("{0:>2} x {1:>1} = {2:>2}".format(7, 5, 7*5))
print("{0:>2} x {1:>1} = {2:>2}".format(8, 5, 8*5))
print("{0:>2} x {1:>1} = {2:>2}".format(9, 5, 9*5))
print("{0:>2} x {1:>1} = {2:>2}".format(10, 5, 10*5))
```

we want a better output

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
```

Arithmetic operators

+	Addition	$3 + 5 = 8$ $3 + 5.0 = 8.0$ $1.2 + 3.4 = 4.6$
-	Subtraction	$5 - 2 = 3$ $5 - 2.0 = 3.0$ $6.5 - 1.2 = 5.3$
*	Multiplication	$5 * 2 = 10$ $5 * 2.0 = 10.0$ $6.5 * 1.3 = 8.45$

Arithmetic operators

/	Division	$10/2 = 5.0$ $10/4 = 2.5$ $10/2.0 = 5.0$ $10.0/1.2 = 8.3333$
//	Floor division	$10//2 = 5$ $10//4 = 2$ $10//2.0 = 5.0$ $10.0//1.2 = 8.0$

What is the difference between **Division** and **Floor division**?

Arithmetic operators

/	Division	$10/2 = 5.0$ $10/4 = 2.5$ $10/2.0 = 5.0$ $10.0/1.2 = 8.3333$
//	Floor division	$10//2 = 5$ $10//4 = 2$ $10//2.0 = 5.0$ $10.0//1.2 = 8.0$

Note that division of two integers give a decimal number

$$10/2 = 5.0$$

So if we want integer result, we should use **Floor division**

$$10//2 = 5$$

Arithmetic operators

<code>**</code>	Exponent	$10^{**}2 = 100$ $10^{**}4 = 10000$ $1.1^{**}2 = 1.21$ $16^{**}0.5 = 4.0$ $36^{**}0.5 = 6.0$
-----------------	----------	--

`16**0.5` square root of 16

Arithmetic operators

%	Modulus	$15\%2 = 1$ $124\%10 = 4$ $28\%2 = 0$ $37\%5 = 2$ $-15\%2 = 1$
---	---------	--

when x is an odd number: $x\%2 = 1$

when x is an even number: $x\%2 = 0$

to find the last digit of positive integers:

$$124\%10 = 4$$

$$23\%10 = 3$$

<code>+=</code>	<code>x += 2</code> is the same as <code>x = x + 2</code>
<code>--</code>	<code>x -= 2</code> is the same as <code>x = x - 2</code>
<code>*=</code>	<code>x *= 2</code> is the same as <code>x = x * 2</code>
<code>/=</code>	<code>x /= 2</code> is the same as <code>x = x / 2</code>
<code>//=</code>	<code>x //= 2</code> is the same as <code>x = x // 2</code>
<code>**=</code>	<code>x **= 2</code> is the same as <code>x = x ** 2</code>
<code>%=</code>	<code>x %= 2</code> is the same as <code>x = x % 2</code>

Problem solving example

A shop sells a product item for \$10, but makes a discount that 3 items only cost \$20. Write a program to ask the user to enter the number of items they want to buy. Then the program displays the cost.

How much does it cost for 7 items?

How much does it cost for 12 items?

How much does it cost for 14 items?