

CSIT110

Fundamental Programming with Python

Loop Statements (2)

Goh X. Y.



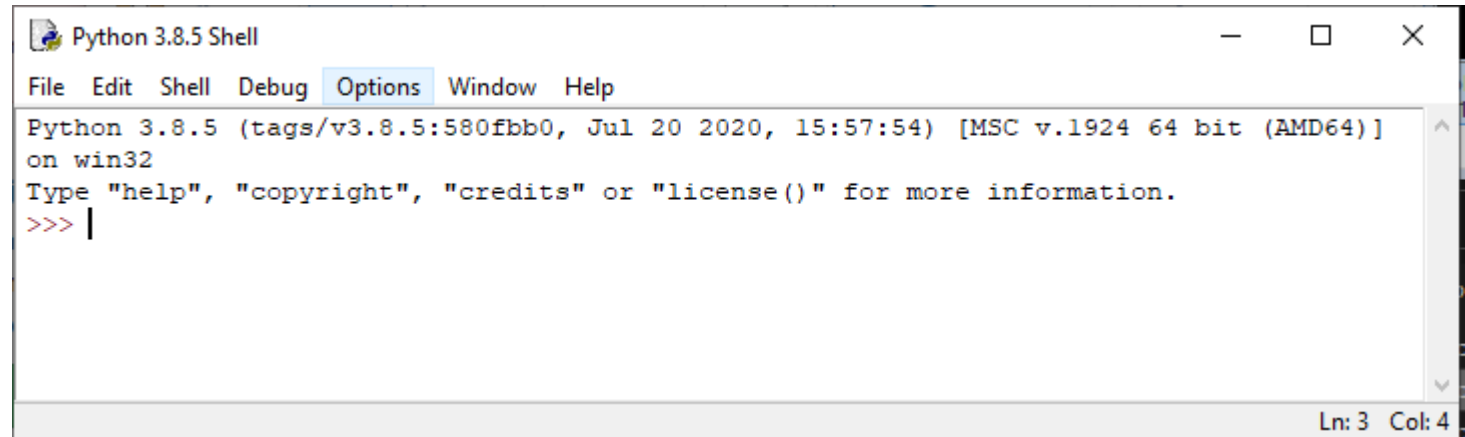
In this lecture

- Recap
- While loop
- Reading documentation

Recap – IDLE

use the python interpreter + run a python script

Window

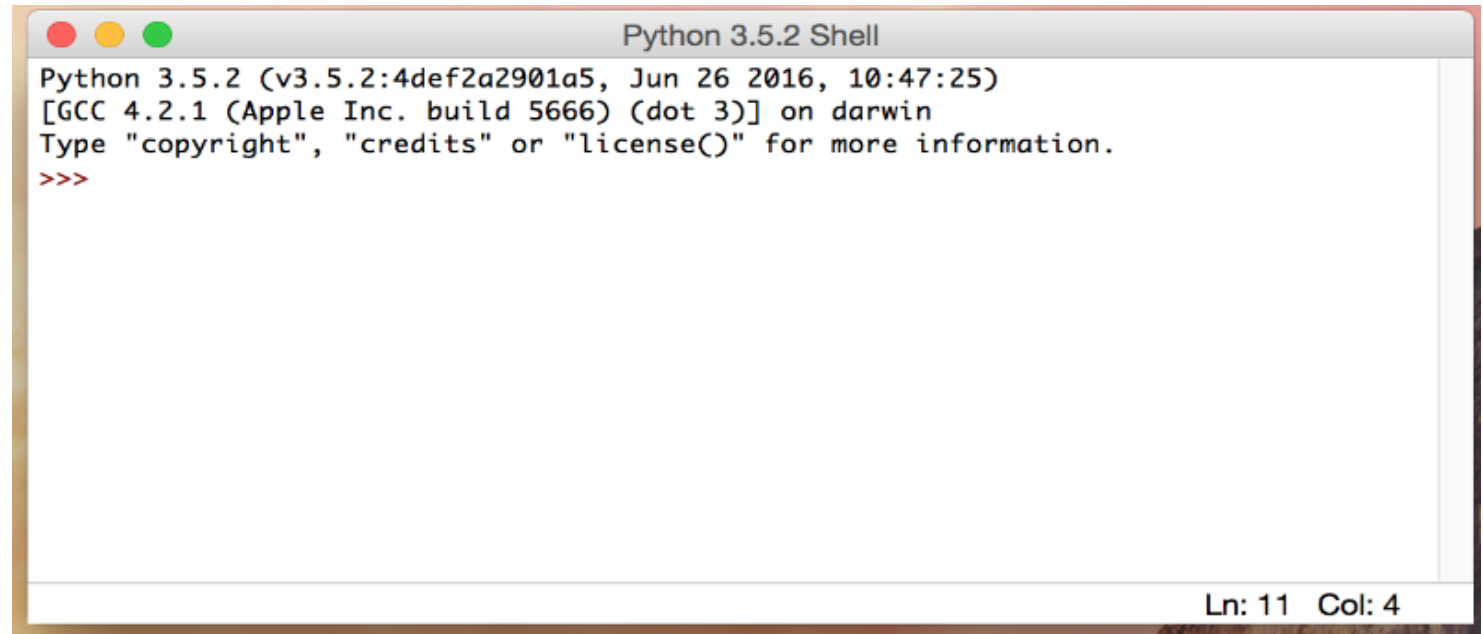


A screenshot of the Python 3.8.5 Shell window on a Windows operating system. The window has a standard Windows title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options (highlighted), Window, and Help. The main text area displays the following text: "Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32" followed by "Type 'help', 'copyright', 'credits' or 'license()' for more information." and a prompt ">>> |". The status bar at the bottom right shows "Ln: 3 Col: 4".

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

Mac/
Unix



A screenshot of the Python 3.5.2 Shell window on a Mac/Unix operating system. The window has a standard Mac/Unix title bar with red, yellow, and green window control buttons. Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options (highlighted), Window, and Help. The main text area displays the following text: "Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25) [GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin" followed by "Type 'copyright', 'credits' or 'license()' for more information." and a prompt ">>>". The status bar at the bottom right shows "Ln: 11 Col: 4".

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

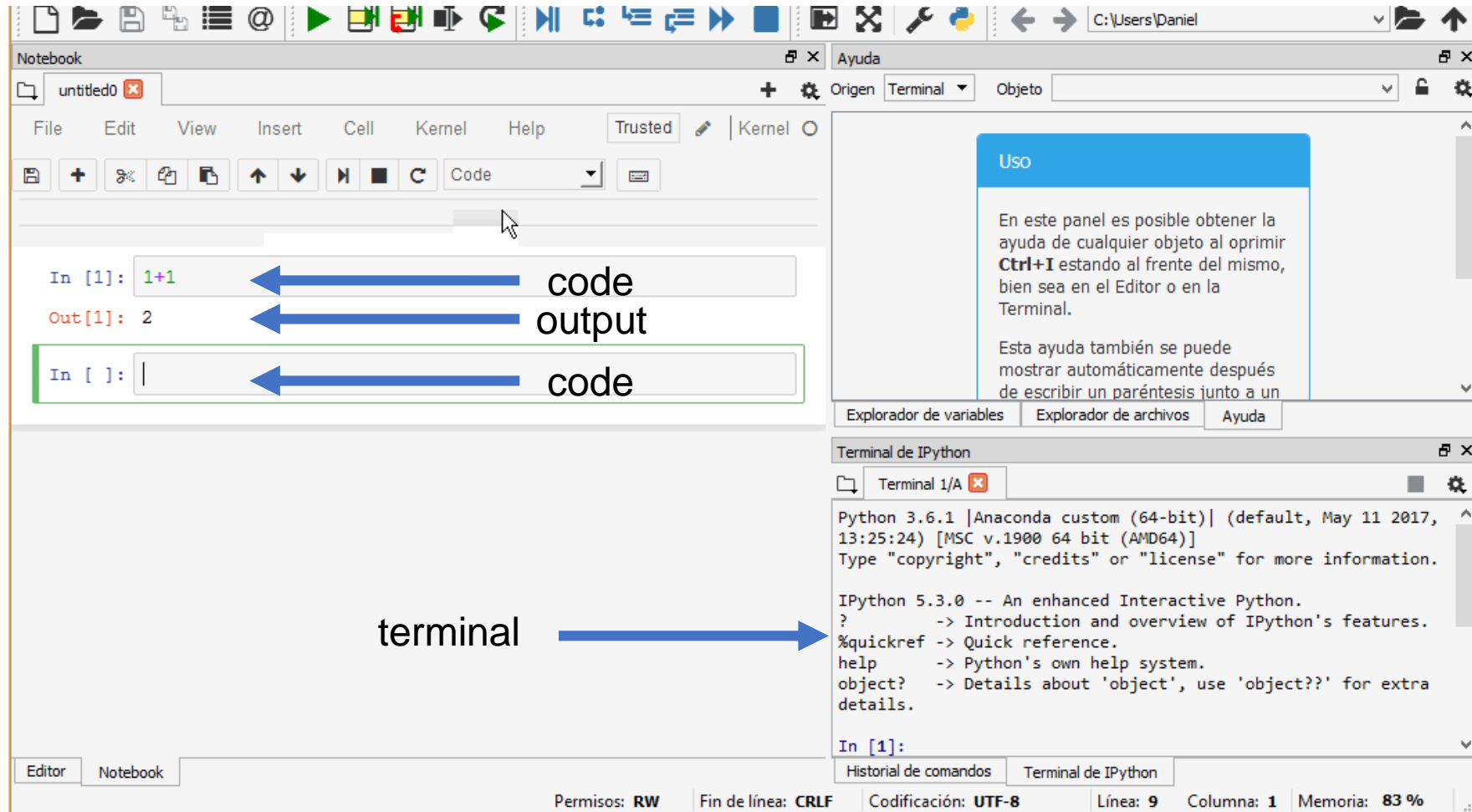
Ln: 11 Col: 4

Using Python Interpreter VS Running a python script

- Hit 'Enter' key to execute a line
- Interprets one line at a time
- A line with only variable name will print out the variable object
- If the line of code returns a value, it will also be printed out.
 - E.g. `>>>type(1232)` gives `<class 'int'>`
- Runs all the command in the script
- Usually triggered by a run button
- Or by typing ``python script.py`` in the console
- A line with only variable name will not print out anything
- Nothing will be printed out if the line of code returns a value unless the `print()` function is used

IDE example

Spyder



IDE example

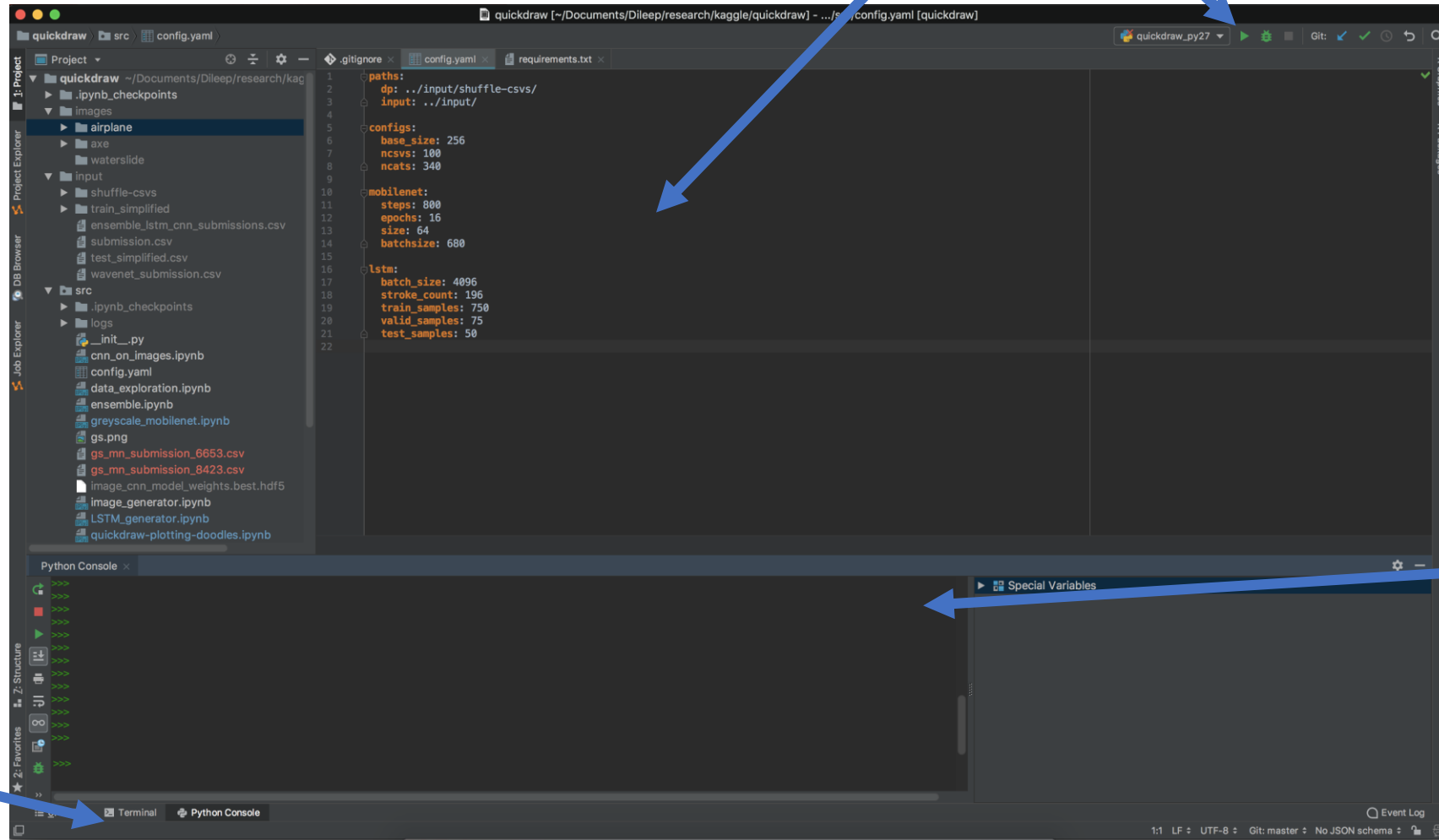
Pycharm

run button

script

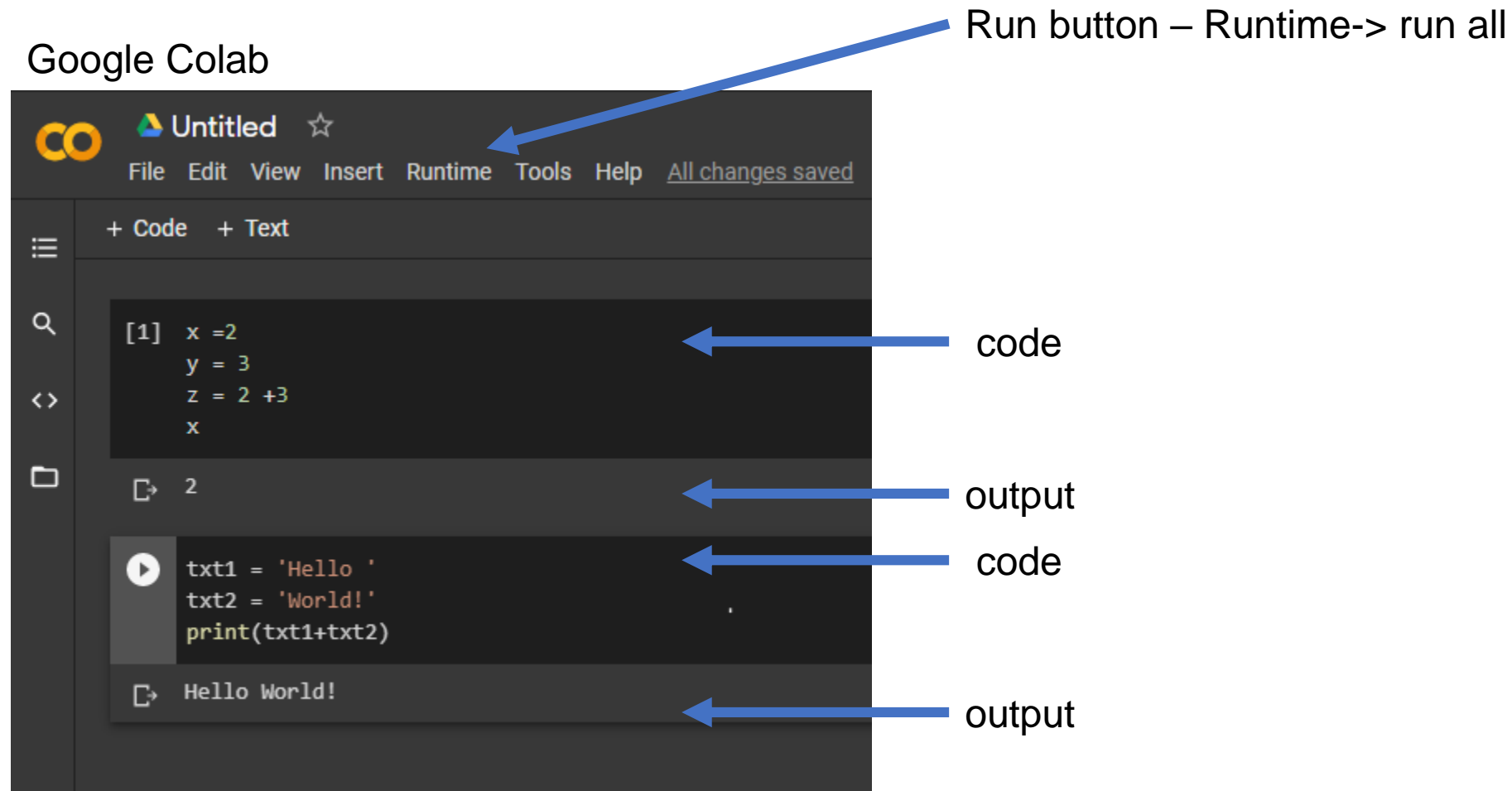
Interpreter
Terminal

Console
output



Recap – Online IDE

Google Colab



The screenshot shows the Google Colab web interface. At the top, there's a header with the Colab logo, the text 'Untitled', a star icon, and a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar, there's a toolbar with '+ Code' and '+ Text' buttons. The main area contains two code cells. The first cell has a '[1]' prompt and contains the code: `x = 2`, `y = 3`, `z = 2 + 3`, and `x`. Below this code is an output cell showing the value '2'. The second cell has a play button icon and contains the code: `txt1 = 'Hello '`, `txt2 = 'World!'`, and `print(txt1+txt2)`. Below this code is an output cell showing the text 'Hello World!'. Blue arrows point from text labels to specific elements: 'Run button – Runtime-> run all' points to the 'Runtime' menu item; 'code' points to the first code cell; 'output' points to the output cell below the first code cell; 'code' points to the second code cell; and 'output' points to the output cell below the second code cell.

Run button – Runtime-> run all

code

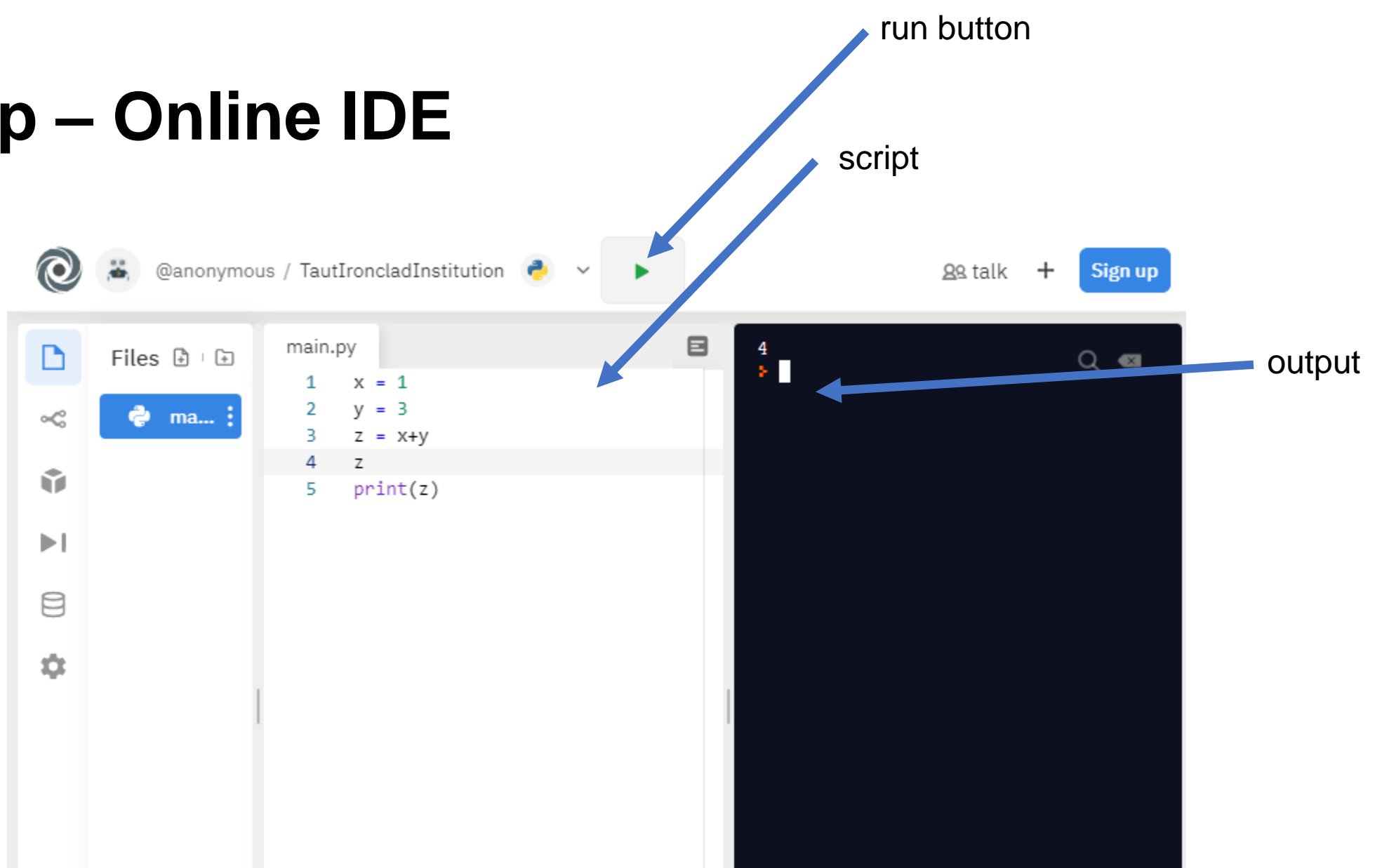
output

code

output

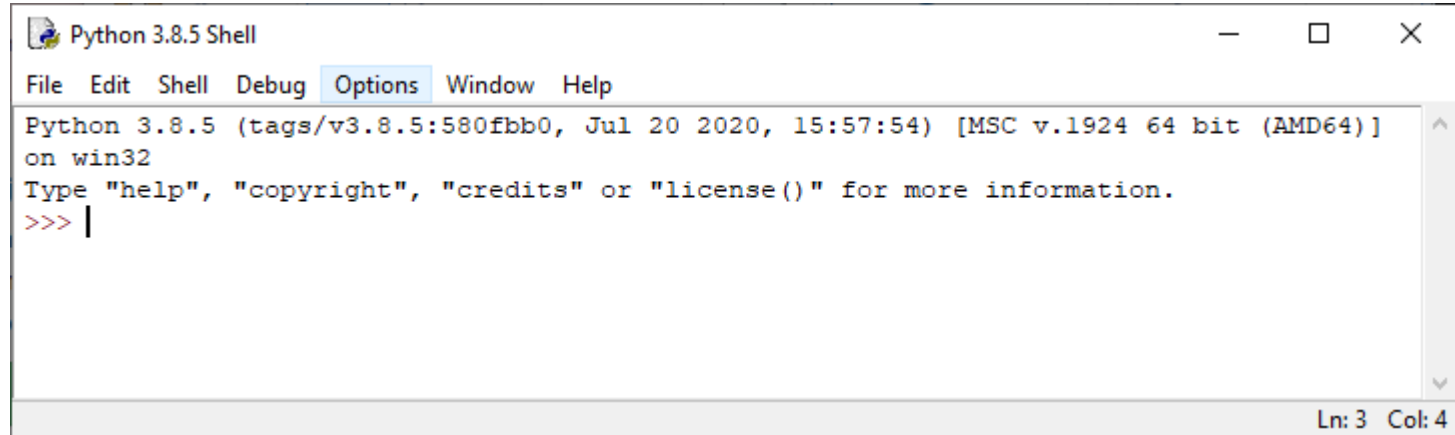
Recap – Online IDE

repl.it



Recap – IDLE: interpreter in a PythonShell

Window

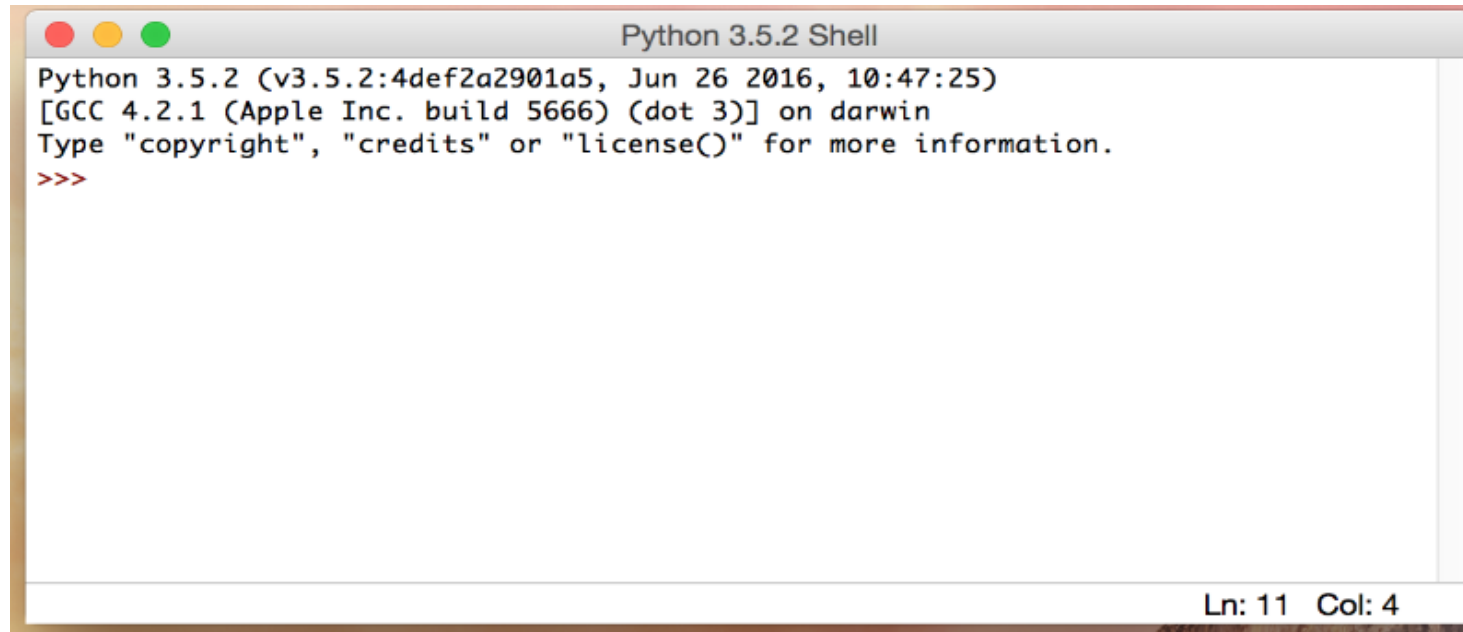


A screenshot of the Python 3.8.5 Shell window on a Windows operating system. The window has a standard Windows title bar with the text "Python 3.8.5 Shell" and standard minimize, maximize, and close buttons. Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options (which is currently selected and highlighted in blue), Window, and Help. The main text area contains the following text: "Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>> " followed by a vertical cursor bar. A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar shows "Ln: 3 Col: 4".

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

Mac/
Unix



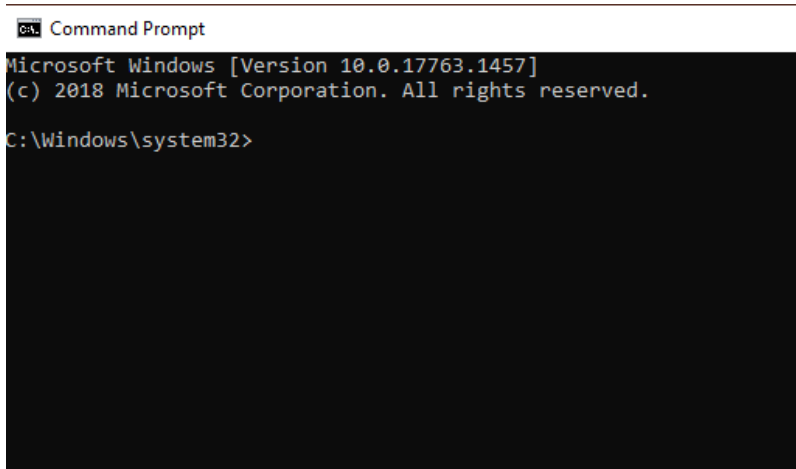
A screenshot of the Python 3.5.2 Shell window on a Mac/Unix operating system. The window has a standard Mac/Unix title bar with three colored window control buttons (red, yellow, green) on the left and the text "Python 3.5.2 Shell". The main text area contains the following text: "Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)", "[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin", "Type 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>> ". A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar shows "Ln: 11 Col: 4".

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Ln: 11 Col: 4

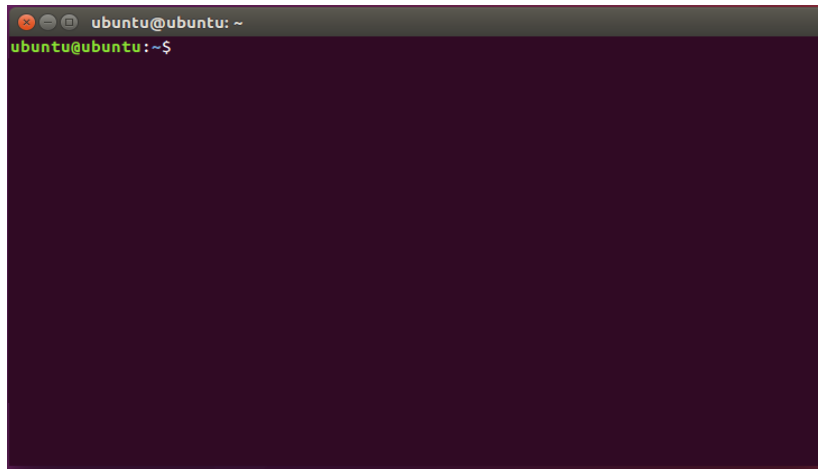
Consoles

- command prompt (Windows), command line, terminal (mac, unix)

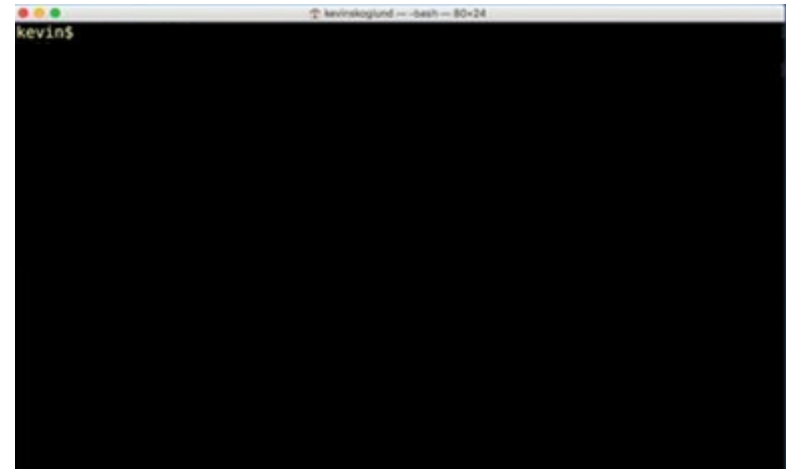


```
Command Prompt
Microsoft Windows [Version 10.0.17763.1457]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```



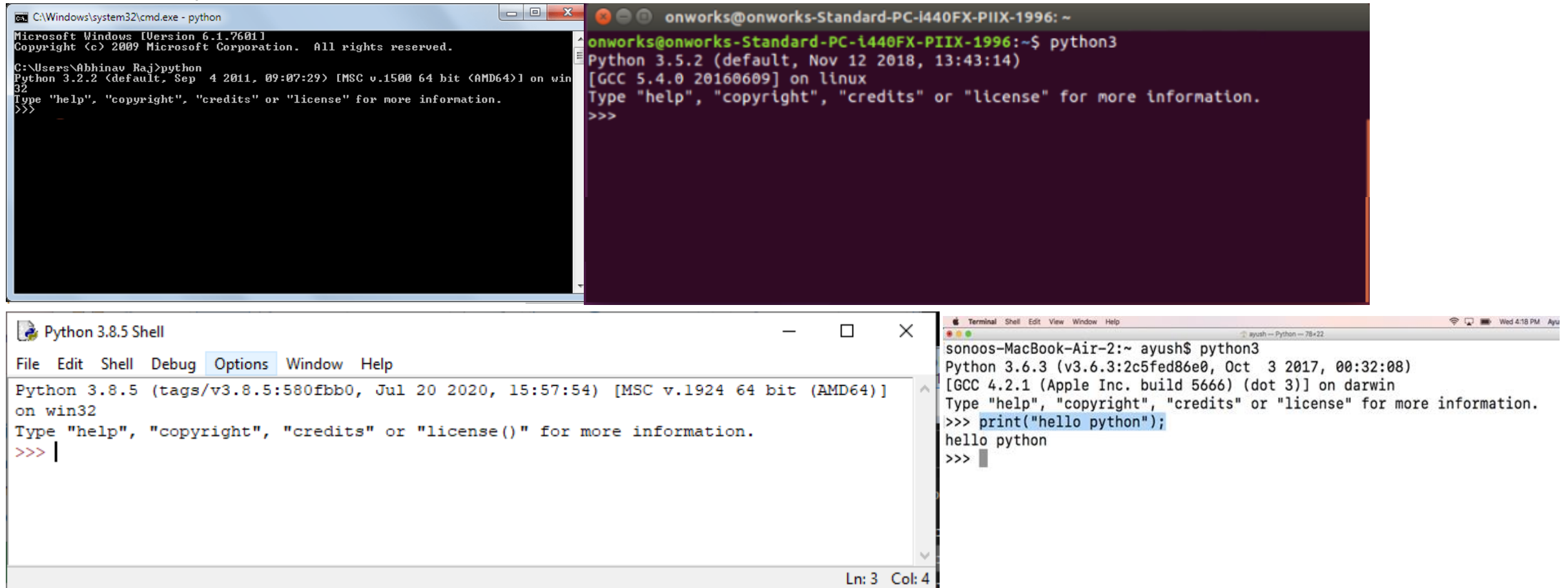
```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$
```



```
kevin$
```

Consoles

Run python interpreter within the console



The image displays three separate console windows, each showing the Python interpreter running on a different operating system.

Top Left (Windows): A Windows Command Prompt window titled "C:\Windows\system32\cmd.exe - python". It shows the output of running the "python" command, which displays the Python 3.2.2 version and build information for Windows.

Top Right (Linux): A Linux terminal window titled "onworks@onworks-Standard-PC-i440FX-PIIX-1996: ~". It shows the output of running the "python3" command, displaying the Python 3.5.2 version and build information for Linux.

Bottom Left (Windows): A Windows Python 3.8.5 Shell window titled "Python 3.8.5 Shell". It shows the output of running the "python" command, displaying the Python 3.8.5 version and build information for Windows.

Bottom Right (macOS): A macOS Terminal window titled "Terminal". It shows the output of running the "python3" command, displaying the Python 3.6.3 version and build information for macOS. The user has also entered the command `>>> print("hello python");` and the output "hello python" is displayed.

Recap – Variables and DataTypes

<code>variable_name1 = 'text value'</code>	<code><class 'str'></code>
<code>variable_name2 = 123</code>	<code><class 'int'></code>
<code>variable_name3 = 321.0</code>	<code><class 'float'></code>
<code>variable_name4 = 4 + 5j</code>	<code><class 'complex'></code>
<code>variable_name5 = True</code>	<code><class 'bool'></code>

ALWAYS use variables with **meaningful names**

`lower_case_with_underscores` for normal variables

`UPPER_CASE_WITH_UNDERSCORES` for constants

Recap - Terminologies

`return_values_or_output = function_always_comes_with_brackets()`

`output = function_name(input, also_known_as_arguments)`

Recap – Input Output

<class 'str'> = **input**(<class 'str'>)

<class 'str'> = **print**(<class 'str'>, end='\n')

Default of end='\n' (a newline)

To print the next output in the same line do this in the previous print statement

print(<class 'str'>, end=' ')

Recap – string format

1. `f'.....{variable_name}...{variable_name}'`
2. `'The three variables are {0}, {1}, {2}'.format(variable0,variable1, variable2)`
3. `'I can align the text like this {0:<15}, {1:^10}, {2:>12}'.format(txt0,txt1,txt2)`

<	left
>	right
^	centre

4. `'To set decimal places I can do {0:<15.1f}, {1:^10.2f}, {2:>12.0f}'.format(price0,price1,price2)`

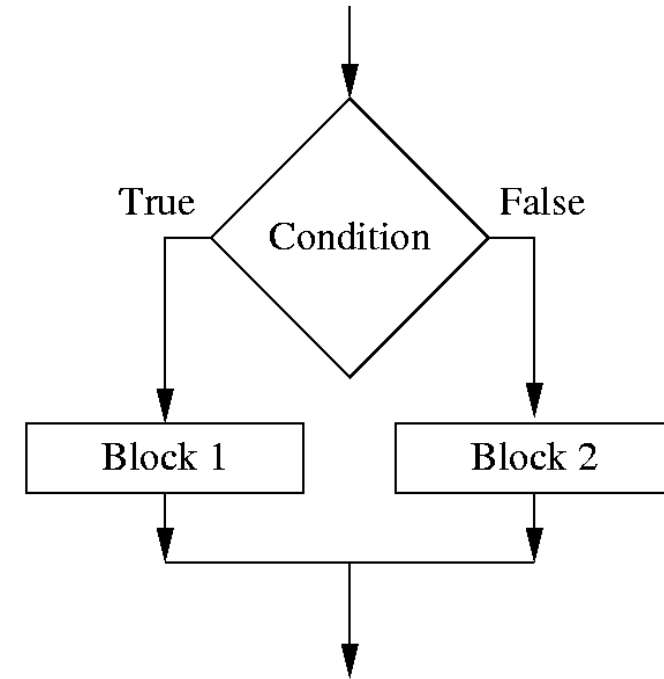
Recap – If-Else statement

```
if (some condition):
```

```
# block statements if condition  
# is True
```

```
else:
```

```
# block statements if condition  
# is False
```



Recap – if - elif - elif - ... - else

```
if (condition1):
```

```
# condition1 is true  
statement  
statement  
...
```

```
elif (condition2):
```

```
# condition1 is false and condition2 is true  
statement  
statement  
...
```

```
elif (condition3):
```

```
# condition1 is false, condition2 is false, and condition3 is true  
statement  
statement  
...
```

```
else:
```

```
# all the conditions are false  
statement  
statement  
...
```

Recap – Common Mistakes

Don't forget the colon :

```
for (x == 2)
^
SyntaxError: invalid syntax
```

if (some condition) :

- this is
- a block
- of codes
- that is indented
- by the same amount
- of spaces

else:

- usually
- we use 2, 3 or 4 spaces for
- indentation

Wrong indentation,
mix-up between spaces and tabs
mix-up number of spaces

What happens when there is no indent:

```
^
IndentationError: expected an indented block
> |
```

Make your choice of indentation and use it consistently!

How does it look like?

```
while (<condition that returns True | False>):
```

```
    # block statements when condition is True
```



The first while-loop example

```
for i in range(0,10):  
    print(i)
```

i = 0, print(i)	→	0
i = 1, print(i)	→	1
i = 2, print(i)	→	2
i = 3, print(i)	→	3
i = 4, print(i)	→	4
i = 5, print(i)	→	5
i = 6, print(i)	→	6
i = 7, print(i)	→	7
i = 8, print(i)	→	8
i = 9, print(i)	→	9

initialization statement

→ i = 0

while (i < 10):
 print(i)

← conditional statement

post-loop statement

→ i = i + 1

Going backwards

initialization statement

`i = 9`

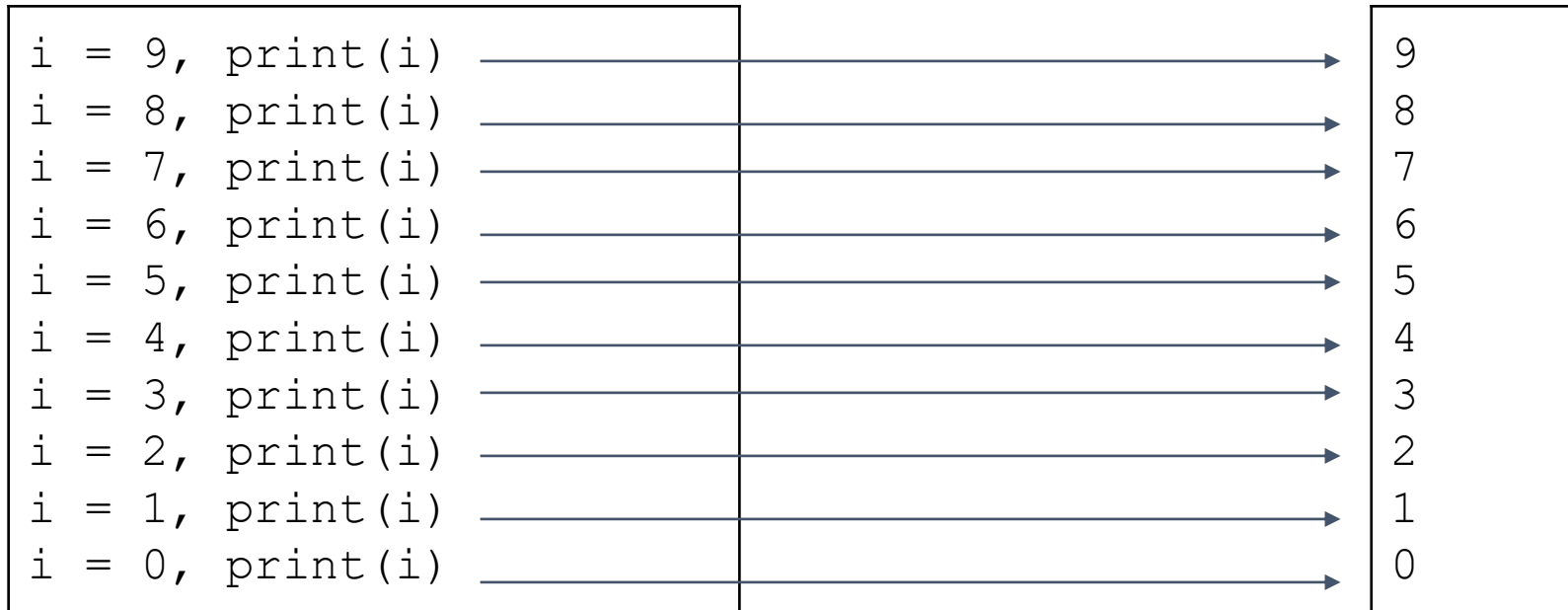
`while (i >= 0):`

`print(i)`

post-loop statement

`i = i - 1`

conditional statement



Times table example

```
for i in range(1,10):  
    print("{0} x {1} = {2}".format(i, 5, 5*i))
```

i = 1, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	1 x 5 = 5
i = 2, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	2 x 5 = 10
i = 3, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	3 x 5 = 15
i = 4, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	4 x 5 = 20
i = 5, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	5 x 5 = 25
i = 6, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	6 x 5 = 30
i = 7, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	7 x 5 = 35
i = 8, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	8 x 5 = 40
i = 9, print("{0} x {1} = {2}".format(i, 5, 5*i))	→	9 x 5 = 45

```
i = 0  
while (i < 10):  
    print("{0} x {1} = {2}".format(i, 5, 5*i))  
    i = i + 1
```

Friend of 10 table

```
for i in range(0,11):  
    print("{0:>2} + {1:>2} = {2:>2}".format(i, 10 - i, 10))
```

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

```
i = 0  
while (i <= 10):  
    print("{0:>2} + {1:>2} = {2:>2}".format(i, 10 - i, 10))  
    i = i + 1
```

Questions



What is the output of the following codes?

A

```
i = 0
while (i < 10):
    print(i)
    i = i + 2
```

B

```
i = 0
while (i < 10):
    i = i + 2
    print(i)
```


Questions



What is the output of the following codes?

C

```
i = 10
while (i < 10):
    print(i)
    i = i + 1
```

D

```
i = 5
while (i < 10):
    print(i)
    i = i + 1
```

E

```
i = 5
while (i < 10):
    i = i + 1
    print(i)
```

Questions



What is the output of the following codes?

F

```
i = 0
i = i + 1
while (i < 10):
    print(i)
    i = i + 1
```

G

```
i = 0
while (i < 10):
    print(i)
```

H

```
while (cat < 10):
    print(cat)
    cat = cat + 1
```

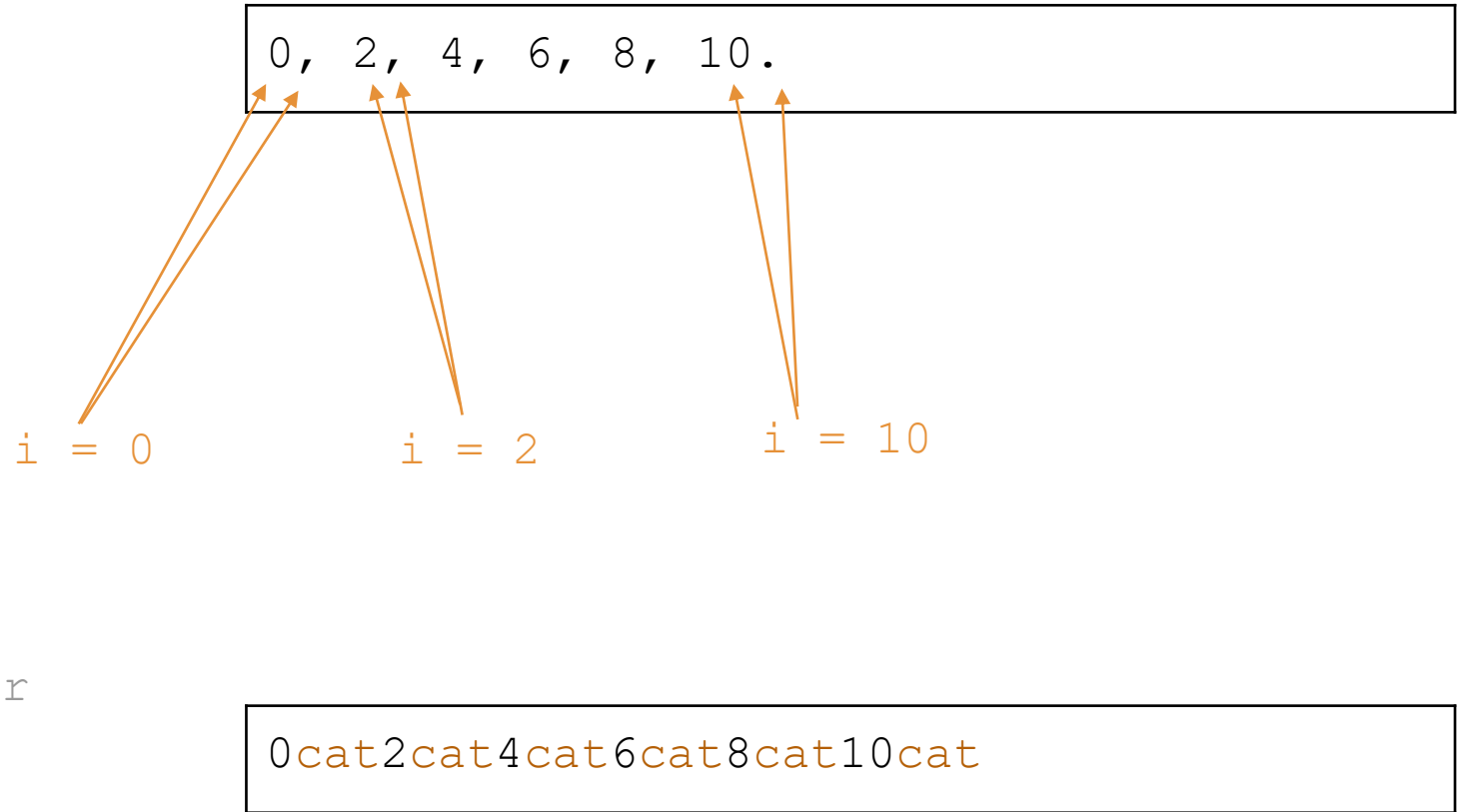
Even numbers

```
i = 0
while (i <= 10):
    trailing = "cat"

    # display the number
    print(i, end="")

    # display the trailing
    print(trailing, end="")

    # update the even number
    i = i + 2
```

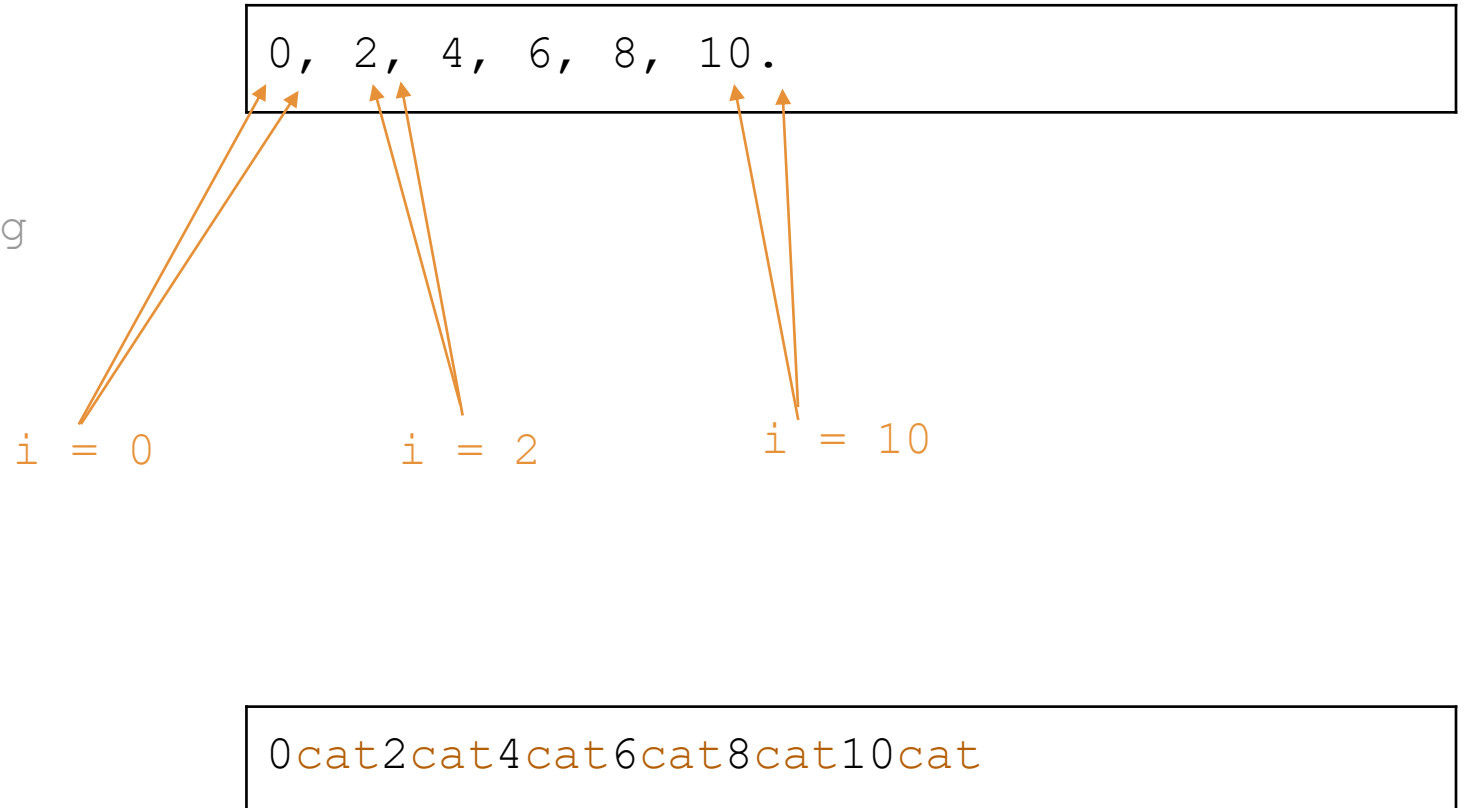


Even numbers

```
i = 0
while (i <= 10):
    # determine the trailing
    if (i < 10):
        trailing = ", "
    else:
        trailing = "."

    print(i, end="")
    print(trailing, end="")

    i = i + 2
```



Display equations

Enter start number: **4**

Enter end number: **7**

Equations:

4 + 4 = 8

5 + 5 = 10

6 + 6 = 12

7 + 7 = 14

```
# ask user for start number
```

```
# ask user for end number
```

```
# display equations between the two input numbers
```

Display equations

```
# ask user for start number and end number
user_input = input("Enter start number: ")
number_start = int(user_input)

user_input = input("Enter end number: ")
number_end = int(user_input)

# display equations between the two input numbers
print("Equations:")

# initialise number to the start number
number = number_start

# repeat as long as number is <= number_end
while(number <= number_end):
    print("{0} + {1} = {2}".format(number, number, number*2))
    # increase the number by 1
    number = number + 1
```

4	+	4	=	8
5	+	5	=	10
6	+	6	=	12
7	+	7	=	14

Example 1: While loops that runs forever!

```
while True:
    user_input = input("Enter something: ")
    print("You have entered: " + user_input)
```

This program will run forever!

```
Enter something: Clocks on fox tick
You have entered: Clocks on fox tick
Enter something: Clocks on Knox tock
You have entered: Clocks on Knox tock
Enter something: Six sick bricks tick
You have entered: Six sick bricks tick
Enter something: Six sick chicks tock
You have entered: Six sick chicks tock
.....
```

Example 2: This while loop will stop if user enters q

```
while True:
    user_input = input("Enter something (or q to quit): ")
    if (user_input == "q"):
        print("Goodbye!")
        break ← use break to stop the loop
    print("You have entered: " + user_input) print()
```

```
Enter something (or q to quit): Clocks on fox tick
You have entered: Clocks on fox tick
```

```
Enter something (or q to quit): Clocks on Knox tock
You have entered: Clocks on Knox tock
```

```
Enter something (or q to quit): Six sick bricks tick
You have entered: Six sick bricks tick
```

```
Enter something (or q to quit): q
Goodbye!
```


Example 3: Keep asking until user enters a positive number

```
Enter a positive integer: -2
```

```
Enter a positive integer: 0
```

```
Enter a positive integer: -5
```

```
Enter a positive integer: 20
```

```
You have entered: 20
```

```
Enter a positive integer: 6
```

```
You have entered: 6
```

Example 3: Keep asking until user enters a positive number

```
while True:
    user_input = input("Enter a positive integer: ")
    number = int(user_input)
    if (number > 0):
        break
print()
print("You have entered: {}".format(number))
```

← User has entered a positive number. Hurray!!!

```
Enter a positive integer: -2
Enter a positive integer: 0
Enter a positive integer: -5
Enter a positive integer: 20

You have entered: 20
```

Example 4: Counting even and odd numbers

```
Enter an integer (or q to quit): 5  
Enter an integer (or q to quit): 7  
Enter an integer (or q to quit): 0  
Enter an integer (or q to quit): 13  
Enter an integer (or q to quit): 8  
Enter an integer (or q to quit): 15  
Enter an integer (or q to quit): q  
You have entered 2 even numbers  
You have entered 4 odd numbers
```

Example 4: Counting even and odd numbers

```
even_count = 0
odd_count = 0
while True:
    user_input = input("Enter an integer (or q to quit): ")

    if (user_input == "q"):
        break

    number = int(user_input)

    if (number%2 == 0):
        even_count += 1
    else:
        odd_count += 1

print("You have entered {0} even numbers".format(even_count))
print("You have entered {0} odd numbers".format(odd_count))
```

Example 5: Green egg and ham?

```
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Oh well, you don't know what you're  
missing!
```

```
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : Y  
That's a smart choice!
```

Example 5: Green egg and ham?

```
# how many time we ask the question
ask_count = 0

# keep asking green egg question
while True:
    answer = input("Would you like green eggs and ham? (Y/N): ")
    ask_count = ask_count + 1

    if (answer == "Y"):
        print("That's a smart choice!")
        break ←———— use break to stop the loop
    if (ask_count == 10):
        # after 10 times, user still says NO, ok enough!
        print("Oh well, you don't know what you're missing!")
        break ←———— use break to stop the loop
```

Extra:

A little more about print()

You can have multiple arguments

To define separators between the input, use sep='<class 'str'>'

Default of sep=' ' (a space)

```
# Useful for formatting a date
```

```
>>>print('09','12','2016', sep='-')
```

```
09-12-2016
```

Extra:

Learning to read Python docs

Any questions?