KNOWLEDGE ORGANISER

KS3 COMPUTING: Year 8 Summer Term Unit 6 Introduction to Python

ST IVO ACADEMY
Astrea Academy Trust

Introduction: In this unit, you will learn how to write computer programs in Python that include a variety of common techniques and features. These will include: printing to the screen, user-input, variables, using different datatypes, doing some maths using BIDMAS, using Python if-elif-else statements and comparing values, and looping through code using Python while loops. You will also learn how to change data types using 'casting', and will become familiar with different types of errors and how to correct them.

Key Learning that will take place in this unit

- To know how to use Python to print text and numbers to the screen.
- To be able to ask the user for input using the keyboard.
- To know the different data types: string, integer, float and Boolean, and how these are used.
- To be able to correct basic syntax errors and program 'bugs'.
- To know how to use variables and to understand their purpose.
- To understand some of the rules (conventions) for naming variables.
- To understand the concept of a sequence of instructions (algorithm).
- To be able to change the data type of variables using 'casting'.
- To be able to do arithmetic in Python using operators and BIDMAS.
- To know how to use selection (Python if.. elif.. else) statements .
- To know what pseudocode is and why it is useful.
- To know three different types or error: syntax, logic and runtime.
- To know how to write selection (Python loop) statements.
- To understand how Python can be used to search for data.

Overview — KEY VOCABULARY: LOOK COVER, WRITE AND CHECK!

Algorithm	An algorithm is set of instructions or rules that need to be followed in order to perform calculations or to solve a problem.
Sequence	The set of instructions or rules that an algorithm uses have to be in the right order. We call instructions in the correct logical order a 'sequence'.
Assign	When we set a variable to a given value – like my_var = 3 – we say that we are "assigning the value of 3 to the variable my_var." We try not to say 'equals' !
Data type	A data type is used to identify data that has common characteristics and purpose. For example, text and numbers are different data types because they are used for different purposes. Python has four data types: string (text), integers (whole numbers), floats (decimal numbers) and Boolean (either a 'true' value or a 'false' value).
Variable	A variable is a name given to an item of data so that the data can be stored in memory while your Python program is running. Variables enable you to input data from the keyboard and to change the data however you need to.
Casting	When we want to change the data types of a value (or the value assigned to a variable), we use casting. Python provides us with the code to do this. So for example, this code changes 43 from a string data type to an integer: **int("43")**
Syntax Error	A syntax error is a mistake in your Python program that prevents it from running (executing). Syntax errors are like spelling and grammar errors. There are also other types of error besides a syntax error: logic error and runtime error.
Input and output	With Python, we can print text and numbers to the screen, and we can also ask the user to input text or numbers using the keyboard.
Pseudocode	Pseudocode is instructions that are written in English (or a language of individual choice). Pseudocode is used to plan-out the correct sequence of instructions and to clarify the key features you may also need to use to make your program work correctly – such as loops and selection statements.
Condition/ Selection	A condition or selection statement is the name given to Python's if-elif-else statement that is used to decide which path a program will take. If a condition is 'true' then Python will choose to run specific lines of code, but if false Python will choose to run different lines of code.
Loops	Python loops allow you to keep revisiting previous lines of code until a certain condition is false. We can do this to use Python to count from one number to another, and then stop. We can also use loops to keep asking the user for input from the keyboard until the user enters particular text (such as 'quit') or a number (such as zero).

operators for arithmetic

Operator	Meaning	Example
+	Addition	4+7 ⟶ 11
-	Subtraction	12-5 ⟶ 7
*	Multiplication	6*6 ⟶ 36
/	Division	30/5 ⟶ 6

Python data types

integer
A whole number

```
File  Edit  Format
print(3 + 2)
5
>>>
```

float
A decimal number

```
File  Edit  Format  Ru
print(3.95 * 2.34)
9.243
>>>
```

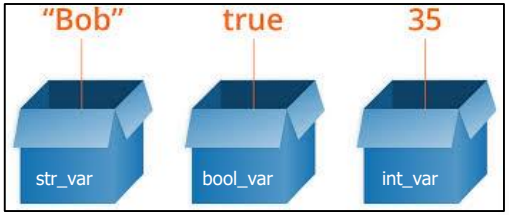
string
A character or text

```
File  Edit  Format  Run
print("hello world")
hello world
>>>
```

Boolean
A True or False value

```
File  Edit  Forma
print(True)
print(False)
True
False
>>>
```

Using Python variables




```
# variables are like values stored in boxes until they are needed.
# the name of the variable is the name of the box.
# these values can change and can be put back in the same boxes as well.

str_var = "Bob"
bool_var = True
int_var = 35

int_var = int_var + 100
print(int_var)

print(str_var + "bidy " + "Bob")
```

```
================
135
Bobbidy Bob
>>>
```

Casting to different data types

We often need to change a data type using casting. For example, if text contains numbers and we want to use it to do maths, we need to change the data type from a string to an integer or a float. Data input from the keyboard is an example of this because the data input is always a string data type and never numbers until we use casting to convert it to an integer or a float.

Sleep calculator

- Extend the problem to find the total number of hours spent sleeping in a month
- Assume an average of 4.35 weeks per month

```
hourspernight = input("How many hours per night do you sleep? ")
hoursperweek = float(hourspernight) * 7
print ("You sleep",hoursperweek,"hours per week")
hourspermonth = hoursperweek * 4.35
print ("You sleep",hourspermonth,"hours per month")
```

Casting from string to a float – so we can do maths!

KNOWLEDGE ORGANISER

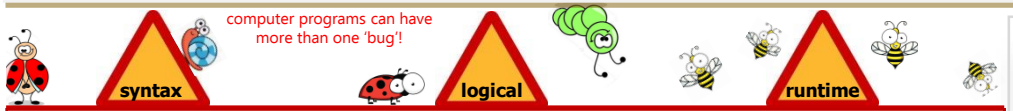
KS3 COMPUTING: Year 8 Summer Term Unit 6 Introduction to Python

ST IVO ACADEMY
Astrea Academy Trust

TEST YOURSELF

1. What is a data type?
2. What can we do with data that is an integer or float data type that we cannot do with data of a string data type?
3. Explain why we use variables in our computer programs. What would we *not be able to do* if variables had never been invented?
4. Write a line of Python code to show how you would use casting to change a float data type to a string data type.
5. Give an example of where you have made a syntax error in your code. What was the error and how did you correct it?
6. Why do you think we don't use the symbol 'x' to multiply numbers together in Python?

Selection and Comparison, Pseudocode and Errors: In this unit you will learn how to make selections using Python's *if..elif..else* statements. You will also learn how to compare numbers using special symbols – as you do in maths when you ask the question is 3 > 4 or is 2<=1. You will learn how two write and to use pseudocode to plan algorithms, and you will learn about the three types of programming errors: syntax errors, logical errors and runtime errors.



syntax	logical	runtime
This error is like a grammar or spelling mistake, and your program won't run until you find and correct the error.	*Your program runs, but it is incorrect. For example, to add 3 and 4 you put 3 – 4 instead of 3 + 4 in your code by mistake.*	*An unexpected error that happens when a program is running. For example, your program can't find a file that it needs.*

colon space...colon space...colon space...colon space.. **INDENTATION** *colon space...colon space...colon space...colon space..*

not indented
indented
not indented
indented
not indented

When you use *selection* in Python, you a colon goes at the end of the *if, elif or else line* and indent the next lines by 2 or more spaces. 2 is ok, 4 is common.

```
if my_age < 18:
    print("too young to vote")
else:
    print("time to vote!")
```

```
If my_age < 18: colon
4 print("too your
else: colon
```

Comparison operators

Operator	Meaning	Example	Evaluates to
==	equal to	7==7	True
!=	not equal to	6!=7	True
>	Greater than	7>6	True
<	Less than	5<8	True
>=	Greater than or equal to	6>=8	False
<=	Less than or equal to	7<=7	True

```
print( 7==7 )  # does 7 equal 7 ? True!

my_age = 34
if my_age <= 50:
    print("True!")

if "hello" != "world":
    print("Nope.. 2 diffrent words!")
```

```
True
True!
Nope.. 2 diffrent words!
>>>
```

Python selection: 'if..elif..else'

If statements

```
my_house = 95

if my_house >=100:
    print("You've walked too far..")

if my_house < 100:
    print("You're in the right place..")

# this isn't the best code. We like to combine
# selection into single blocks if we can .....
```

using if .. else..

```
my_house = 95

if my_house >=100:
    print("You've walked too far..")
else:
    print("You're in the right place..")

# this is better. There is less code and it
# is more logical to express the condition
```

using if.. elif.. else..

```
my_house = 95

if my_house >=110:
    print("You've walked too far..")
elif my_house < 110 and my_house >= 70:
    print("You're in the right place..")
else:
    print("I think you may be lost!")

# if.. elif.. else can cater for lots of
# possibilities!
```

this is how Booleans can be used..

```
bool_var = ( 3 < 4 )   # this is true..
print(bool_var)        # 3 is less than 4 !
```

```
True
>>>
```

Pseudocode is used to plan an algorithm with the correct sequence of instructions before writing the code in Python. There is no syntax. You can use ordinary English to plan your algorithm if you want.

```
if unlock button pressed then
    display PIN unlock screen
    input PIN
    if correct PIN entered then
        unlock phone
    else display "Try again"
    endif
else display lock screen
endif
```

Pseudocode: saying 'becomes'

Although pseudocode doesn't have a specific syntax, and can be written in English, you may see the arrow symbol <- when a variable is being assigned a value. In programming, it's quite useful to use the word 'becomes' rather than 'equals' because in programming we are not really expressing equality like we do in maths.

In maths..	..we would say:
5 = 3 + 4	*five 'equals' three plus four.*
In pseudocode..	**..we would say:**
my_var <- 3 + 4	*my_var **'becomes'** three plus four.*
lives <- lives - 1	*lives **'becomes'** the old value of lives minus 1.*

```
#GhostHunter
if ghost catches man then
   lives ← lives - 1
endif
if lives = 0 then
   display "Game Over!"
else
   display "Arghhh!"
endif
```

AT HOME. (Parents/carers may be able to help with this.)

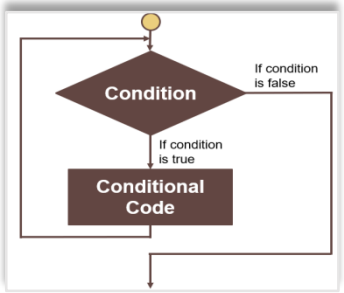
Write a computer program in Python that asks the user to input two numbers. Your program should determine which number is the smallest and print a message like "You input 23 and 45. 23 is the smallest number". Once you have this working, ***challenge yourself*** to extend your program to ask the user for *three numbers.* Can you write a Python program to find the smallest of these three numbers, and print out a similar message?

KNOWLEDGE ORGANISER

KS3 COMPUTING: Year 8 Summer Term Unit 6 Introduction to Python

ST IVO ACADEMY
Astrea Academy Trust

Loops and Searches: This unit extends your knowledge about conditions. You will learn how to express a Python loop as a flowchart, write loops in Python - for example to count until a particular number is reached. We will use pseudocode to plan an algorithm and you will learn how to place an if-else-statements inside loops as well. We then compare two very common search algorithms: the linear search and the binary search.



A Python loop allows us to revisit previous lines of code and repeat them until some condition we have set becomes false. Without loops, we would have to write the same lines of code over and over again!

For a condition to change from true to false, something must change while the loop is running. Maybe we are counting from 1 to 10 and we reach 10. The condition would be: *while counter != 10* (is True).
Once *counter* becomes the value of 10, the condition is no longer true; it is false and so the loop stops. The change in the loop happens when we specify *counter <- counter + 1*. Without this line of code, the loop would continue forever!

'While loop' pseudocode

'Nagging' Example

```
print "Mum, can I have an ice cream?"
input answer
while answer is "No"
    print "Please can I have an ice cream?"
    input answer
end while loop
print "Thank you!"
```

This block of code is the while loop

INDENTATION

colon space...colon space...colon space...colon space..

not indented ***indented***
not indented ***indented***
not indented ***indented***

Just like using selection, we write the first line of a *while loop* with a colon at the end. The next lines are indented. 2 spaces are ok and 4 is ok because this is the same as using the TAB key to indent.

```
counter = 0
while counter != 11:
    print(counter)
    counter = counter + 1
```

colon space...colon space...colon space...colon space..

```
while counter != 11: colon
    print(counter)
    counter = counter + 1
```

4

Examples of Python loops

```
counter = 0
while counter != 11:
    print(counter)
    counter = counter + 1
```

```
# a nagging example!
answer = input("Mum, can I have some ice cream please? ")

while answer == "no":
    answer = input("Mum, can I have some ice cream please? ")

print("Thanks Mum!")
```

Using an if.. else.. condition inside a Python loop

```
sys_password = "pe$1256"
password = input("Please enter your password >> ")
attempts = 1
logged_in = False
while attempts != 3:
    if password == sys_password:
        attempts = 3 # stops the loop
        print("password correct..")
        logged_in = True
    else:
        attempts = attempts + 1
        password = input("Please enter your password >> ")

if logged_in == False:
    print("you have been locked out")
```

Random numbers

Python let's you generate random numbers using its random number function. Here is an example to get a random number between 1 and 20 inclusive:

```
import random

randnumber = random.randint(1, 20)
print(randnumber)
```

```
>>>
9
>>>
4
>>>
14
>>>
6
```

Searching for data using Python loops: the linear search and the binary search

Imagine searching for the number 8 in the list below. It's the last item in the list in this example, and that's significant.

23 5 19 1 9 45 10 8

A **linear search** would start with 23 and goes through each item, one-by-one, until it finds the number 8. If number 8 were nearer the front of the list, it would be found faster.

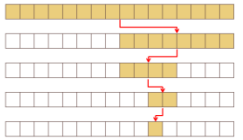
What does this tell you about how efficient a linear search is? What would make this search take even longer? What could we do to make it more efficient?

Linear search pseudocode

```
read the first item in list
while not end of list and item not found
    read next item in list
    if item found then
        display item
    end if
end while
```

What tactic could you use to guess a number between 1 and 100?

If you guessed incorrectly, then you could ask if you were too high or too low, and then keep guessing higher or lower. Each guess, you get closer and closer to the correct answer until eventually you have one number left. This is how a binary search works!



Looking for the number **40** in a list 1–100

number	max	min	mid	number is
40	100	1	50	lower
40	49	1	25	higher
40	49	26	37	higher
40	49	38	43	lower
40	42	38	40	found!
40	found in 5 reads!			

```
max ← highest number
min ← 1
mid ← (max + min) // 2
while number <> mid
    if number < mid then
        max ← mid - 1
    else
        min ← mid + 1
    end if
    mid ← (max + min) /2
end while loop
print 'number found'
```

A binary search *will only work on a list that is sorted in order*, but it's still more efficient than a linear search because it always rejects the half of the list that the item will never be found in!

Binary searches are popular for searching large amounts of sorted data.

TEST YOURSELF

1. Explain how a while loop needs a condition to change from true to false for the loop to work.
2. Explain why a linear search is not very efficient.
3. What needs to be in place before we can use a binary search on a list of data?
4. Explain the principles of a binary search.
5. Why is a binary search more efficient than a linear search?
6. If we wanted to use a while loop to count from 1 to 10 inclusive, is this condition correct or incorrect? why?

   while counter !=11:

7. Give three examples of games, that could be written as computer programs, where you might put a random number or random numbers to good use.
8. What code syntax must you have in place for a while loop to work?

Useful links:

https://www.bbc.co.uk/bitesize/topics/zhy39j6

https://www.w3schools.com/python/

http://introtopython.org/

https://www.tynker.com/

https://trinket.io

https://www.codecademy.com/catalog/language/python/

http://www.practicepython.org/

check IO

AT HOME. (Parents/carers may be able to help with this)

Write a Python program that counts from -30 to 100. This will represent the temperature in degrees centigrade. You will need to Google how to convert centigrade into Fahrenheit and then print out the temperature in both centigrade and Fahrenheit. Include some code to print "zero reached" when the loop reaches -18 as it is counting up to 100.