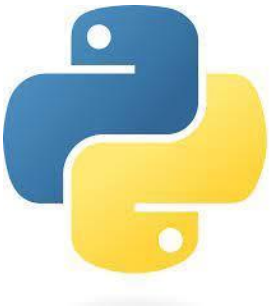
Code Clock

Tutorial 1: Basic Input/Output

Time.to.code



Programming with Python

@ QUB

Introduction to Python

Welcome to the exciting world of Python programming! 🐍 Python is a fun and powerful programming language that's perfect for beginners, especially kids like you. Whether you want to create your own games, solve puzzles, or build cool projects, Python makes it easy to turn your ideas into reality.

What is Python?

Python is a type of language that computers understand. Just like we use English to talk to each other, we use Python to tell computers what to do. It's one of the most popular programming languages in the world because it's simple to learn and very powerful.

Why Learn Python

1. **Easy to Read**: Python uses simple words and symbols, making it easy to understand.

2. **Versatile**: You can use Python to make games, control robots, create websites, and much more!

3. **Fun and Interactive**: With Python, you can quickly see the results of your work, making it really fun to learn and experiment.

What Can You Do with Python?

- **Create Games**: Imagine making your own video game from scratch!

- **Solve Problems**: Python can help you solve maths problems, puzzles, and even real-world challenges.

- **Build Projects**: From simple calculators to complex robots, Python lets you build amazing things.

Getting Started

You will be using **Idle** in QUB to learn Python. Idle is a free-to-download Python Code Editor.

Your First Program

Let's write your very first Python program together. It's called "Hello, World!" and it's super simple. This program will make the computer say "Hello, World!" on the screen. Ready? Let's go!

Open up Idle and create a new Python file called “HelloWorld.py”. Save it on your desktop.

Enter the following code:

```
print("Hello, World!")
```

What Happens Here?

- `print`: This is a command that tells the computer to display something on the screen.

- `"Hello, World!"`: This is the message we want to show.

When you run this program, you'll see the words "Hello, World!" appear on your screen. Congratulations, you just wrote your first Python program!

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code as seen below

```
counter=1

while(counter<=5):
    print(2*counter)
    #increments counter by 1 each iteration of the loop
    counter=counter+1
    print("The value of the counter is " + counter)

print("The loop has now terminated")
```

Python will give you an error if you skip the indentation.

Python Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

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

Enter the following code:

```
#print(“Hello world”)
print(“Cheers matey”);
```

To add a multiline comment you could insert a # for each line:

Enter the following code:

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

Variables

Variables are containers for storing data values.

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Enter the following code:

```
x = 5
y = "John"
print(x)
print(y)
```

Now enter the following code:

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot 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)
- A variable name cannot be any of the Python keywords.

Enter the following code:

```
myvar = "Code Clock"

my_var = "Code Clock"

_my_var = "Code Clock"

myVar = "Code Clock"

MYVAR = "Code Clock"

myvar2 = "Code Clock"
```

Now try entering the following illegal variable names:

```
2myvar = "John"

my-var = "John"

my var = "John"
```

Casting

If you want to specify the data type of a variable, this can be done with casting.

Now enter the following code:

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

You can get the data type of a variable with the type() function.

Now enter the following code:

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Case-Sensitive

Variable names are case-sensitive.

Now enter the following code:

```
a = 4
A = "Sally"
print(a)
print(A)
#A will not overwrite a
```

Output Variables

The Python print() function is often used to output variables.

Enter the following code:

```
x = "Code Clock is awesome"
print(x)
```

In the print() function, you output multiple variables, separated by a comma:
Now enter the following code:

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Boolean Type:	bool

Python Numbers

There are two numeric types in Python:

- int
- float

Variables of numeric types are created when you assign a value to them.

Enter the following code:

```
x = 1    # int
y = 2.8  # float
```

Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers.

The first line should go at the top of your python file. This basically imports the python code into the memory of the program you are creating. In this case this imports random into your program so you can generate a random number

Enter the following code:

```
import random

print(random.randrange(1, 10))
```

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

Enter the following code: (notice the indentation)

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Elif

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

Now enter the following code:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Else

The else keyword catches anything which isn't caught by the preceding conditions.

Now enter the following code:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

Enter the following code:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

And

The and keyword is a logical operator, and is used to combine conditional statements:

Test if a is greater than b, AND if c is greater than a:

Enter the following code:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or

The or keyword is a logical operator, and is used to combine conditional statements:

Test if a is greater than b, OR if a is greater than c:

Enter the following code:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Not

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

Test if a is NOT greater than b:

Enter the following code:

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested If

You can have if statements inside if statements, this is called *nested* if statements.

Enter the following code:

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Challenge: Using basic input/output commands, write the code which does the following:

1. Prints today's date
2. Asks the user to enter their name, gender, and age and prints it out.
3. Using the user's age, checks if they are under 12 in which case, it should say "You are too young to be here!"
4. Checks the user's age against the following age boundaries for each year group and prints their year out. i.e. Year 8 (12), Year 9 (13), Year 10 (14), Year 11 (15) etc