PYTHON COURSE

ENTRY LEVEL

Basics of programming in Python 3.10

This course will cover part of the arguments found in PCEP™ – Certified Entry-Level Python Programmer Certification

PYTHON'S LOGIC AND STRUCTURE

START BUILDING A PYTHON PROGRAM

As we said earlier, **a program is a sequence of instructions.**
So, in order to write more complex programs, we'll need some technical instruments.

You can use different tools, we'll start with a text editor called **Sublime Text. (Optional)**

In the next future, once you'll learned the basics of python**, you'll use an IDE (Integrated Development Environment) to program**

sublimetext.com/downloads

RUN PYTHON PROGRAMS

- Create a new empty file and write:

```
print("Hello World")
```

- Save the file as hello.py **(name is optional, but file must have .py extension)**

- open the PowerShell or Terminal, go to the hello.py file storage location and type:

```
python3 hello.py
```

- python3 (or just python) will call the interpreter, then your file is passed (in this case, hello.py)

BUILD PYTHON PROGRAMS: ELEMENTS

Identifiers (Names):

In a python program, identifiers (names) can be written with all **uppercase or lowercase characters between A and Z**, **underscores and digits (except for the first characters).**

KEYWORDS

special type of identifiers, used as *reserved words of python language*, those **can't be used as identifiers for other stuff in your program**.
Everyone of them has a specific use in python, and, **when used, must be spelled exactly as we'll see.**

BUILD PYTHON PROGRAMS: ELEMENTS

Table of Python keywords

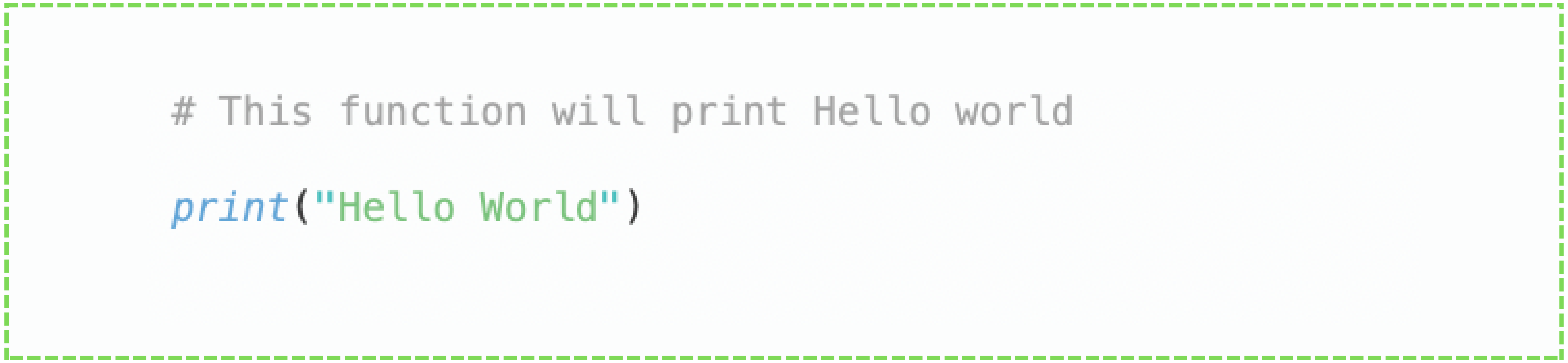
```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

During next lessons, we’ll see how to use the most of those keywords in python programs.

BUILD PYTHON PROGRAMS: ELEMENTS

COMMENTS

- Parts of the program **ignored by the interpreter** which can be **used to comment python code**



```python
# This function will print Hello world
print("Hello World")
```

- Use **# to start writing a comment**, all the following text in the line will be treated as a comment

BUILD PYTHON PROGRAMS: ELEMENTS

LITERALS

literals are a notation for representing a fixed value in source code.

```python
print("Hello World")
```

They are **used to provide variable values or to directly utilize them in expressions.**

LITERALS

data structure that can hold any value type, such as

Numeric

Numeric Literals

```
# Example of a Numeric literal
3

# Example of a Numeric literal
3.0
```

Character

Character literal

```
# Example of a character literal
'c'

# Example of a character literal
"c"
```

String

String literals

```
# Example of a string literal
'Hello'

# Example of a string literal
"Hello"
```

Boolean

Boolean literals

```
# Example of a boolean literal
True

# Example of a boolean literal
False
```

we'll see other types of literals in the next lessons, such as collections and special literals

NUMERICAL LITERALS

They are immutable and there are three types of numeric literal:

1. INTEGER
2. FLOAT
3. COMPLEX

PYTHON

identified as:

NUMERICAL LITERALS: INTEGER

int

Positive and negative numbers including 0, with no fractional part.

Python supports **different numeral systems for integer literals**

```
# Decimals

50

# Binary

0b10100 # 20 decimal
```

```
# Octal
0o320 # 208 decimal
# Hexadecimal
0x12b # 299 decimal
```

when **displayed**, all will be **converted into decimal.**

```
>>> print(0b10100)
20
```

PYTHON

NUMERICAL LITERALS: FLOAT

identified as:

float

Real numbers having both integer and fractional parts.

```
# example of float literal

37.8
```

```
>>> print(37.8)
37.8
```

PYTHON

STRING LITERAL

identified as:

str

Text that can be sourrounded by ‘ or “, can be multi-line if sourrounded by ‘”, or “””
(a number in a string will be treated as text)

```
# Example of string literals

"Hello world"

'Hello world'
```

MULTI-LINE STRINGS

You can write multi-line string by using “”” or ‘’’ to surround text:

```
# Example of multi line string


"""
Here's a multi-line
string
hello World!
"""
```

if not used, a multi-line string will be ignored by the compiler (treated as a comment and so not executed)

multi_line.py

```python
print("""
Here's a multi-line
string
hello World!
""")
```

```
[> python3 multi_line.py

Here's a multi-line
string
hello World!
```

identified as:

str

CHARACTER LITERAL

not an express type, (python don't provide a specific data type for characters) is a single component string (a string literal is composed by characters)

```
# Example of character

"a"
```

a **special type of characters** are **"escape characters"**

- represented by **a backslash followed by the character to insert**, escape characters are used to perform some actions or handle tricky characters when used inside strings

```
# this escape character is used as next line

"\n"
```

CHARACTER LITERAL

example of using escape characters in strings

escape_chars.py

```python
# print the string in two lines

print("Hello\nWorld!")

# using apices as escape characters to be correctly printed when inside a string

print("Hello to the so called \"World\"")
```

```
[> python3 escape_chars.py
 Hello
 World!
 Hello to the so called "World"
```

PYTHON

BOOLEAN LITERAL

identified as:

bool

Represents a boolean value, and can only be True or False (representing 1 and 0)

```
# example of boolean literals

True

False
```

python supports all of this data-types and others with built-in data types to perform different operations on it in your programs

BUILT-IN DATA TYPES

python supports all of this value types and others with **built-in data types intended to manage data and perform different operations on it in your programs.**

Every different data-type, in python, can be **used to perform different operations** and **not every operation can be performed same way on each of those data types.**

So, from now on, pay attention to **chose the correct data type that suits the operations you need to execute in your program.**

RECAP: DATA TYPES TABLE

Data Type	Python Notation	Example Python Syntax
String	str	“Hello World!”
Boolean	bool	True
Integer	int	178
Float	float	34.76
Complex	complex	6j

RECAP: ARGUMENTS

- How to write and run in console python programs as .py files
- Basic elements in a program: Keywords, Comments
- Basic elements in a program: Literal values
- Integers and Float numbers
- Different integer numeral systems in python
- Strings
- Characters, escape characters
- Booleans

QUESTION TIME

QUESTION 1:

Select all the correct statements:

1. command python3 main.py in console is used to execute a python program named main.
2. keywords are a special type of identifier which can be used as identifier for everything in a program
3. comments are written by using #, all the following text in same line will be ignored from the compiler.
4. “True” is an example of boolean literal.

PYTHON

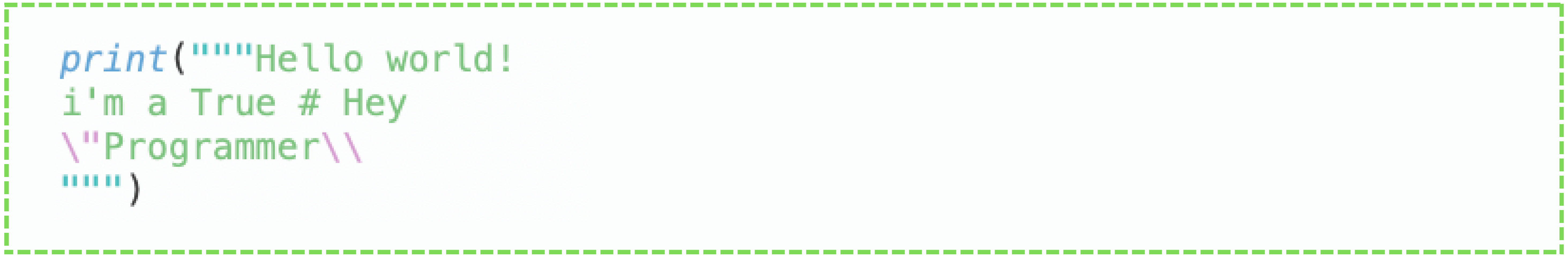
QUESTION 2:

Select all the correct statements:

1. binary numerical system based literals are an example of Integer literals.
2. Python has a specific built-in data type for characters
3. “\n” escape character is used to split strings in multiple lines.
4. Float literals represents floating point numbers like 45.7

QUESTION 3:

What is the output of the following python program:



```
print("""Hello world!
i'm a True # Hey
\"Programmer\\
""")
```

1: Hello world!

i'm a True # Hey

“Programmer\

2: Hello world!

i'm a True

“Programmer\

3: Hello world! i'm a True “Programmer