# 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

## OPERATORS AND OPERATIONS

## BASIC STRING OPERATIONS

There are two type of basic string operators that can be used to perform operations between strings

- **Concatenation** operator

  used to perform concatenations between multiple strings

- **Repetition** operator

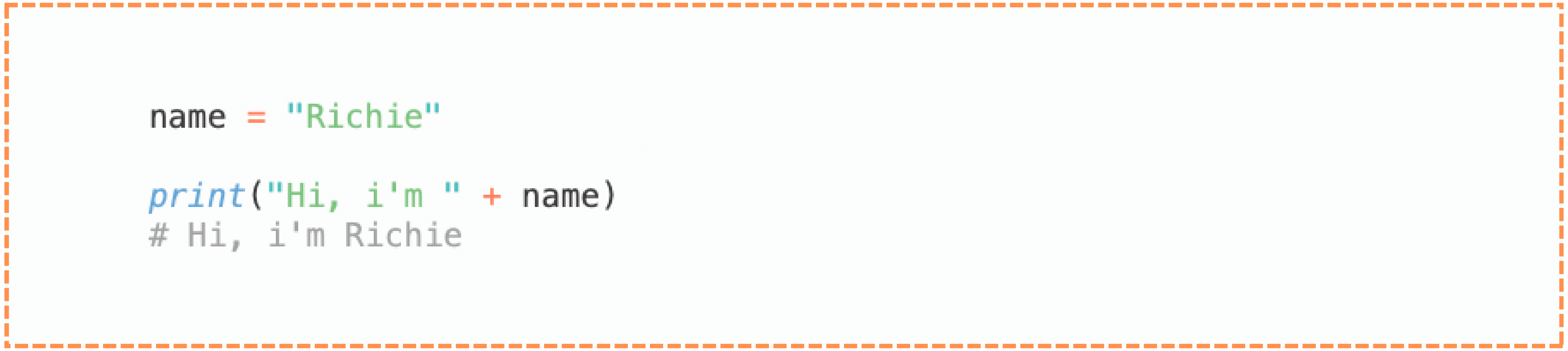
  used to concatenate n copies of the same string

**Any type of operation performed between strings will result in a string.**

PYTHON

## STRING OPERATIONS: CONCATENATE

- Strings concatenations consists in joining one or multiple string together
- Concatenation is performed using the + operator
- Result will be a single string containing all the characters of the concatenated strings



```python
name = "Richie"

print("Hi, i'm " + name)
# Hi, i'm Richie
```

## STRING OPERATIONS: CONCATENATE

Remember, as numerics, string also are immutable

```python
name = "Mark"

print(type(name))
#<class: 'str'>

name + " Trevor"

print(name)
# Mark

name = name + " Trevor"

print(name)
# Mark Trevor
```

## STRING OPERATIONS: REPEAT

- Strings repetitions consists in concatenating multiple copies of the same string
- Repetition is performed using the * operator as follows: string * times (int)
- Result will be a single string containing n times the characters of repeated string

```python
msg = "Hey "

repeated_msg = msg * 3 # repeat the string contained in msg variable 3 times

print(repeated_msg)
# Hey Hey Hey
```

## STRING OPERATIONS: PRIORITY

- Strings operation are performed from left to right
- Priorities are established using round parentheses

```
name = "Mark"
work = "Programmer"
msg = "Hey "

message = "I'm " + name + " and i'm a " + work + " " + msg * 3

print(message)

# I'm Mark and i'm a Programmer Hey Hey Hey

second_message = (name + " " + work) * 3

print(second_message)

# Mark ProgrammerMark ProgrammerMark Programmer
```

## EXERCISES

## SLICING AND RANGE-SLICING

## STRING OPERATIONS: SLICING

The operation of slicing consists in extracting a single part of an element

### PYTHON STRING INDEXING

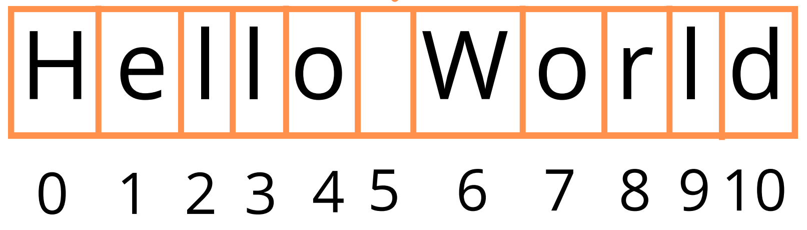
As we introduced, **strings** can be **considered** as some sort of **characters sequences**

- Every single character in a string has an **integer positional index**.
- In Python, positional indexing starts from 0, that means that the first character in a string, is considered in position 0
- You can extract characters from a string by using squared parentheses

## STRING OPERATIONS: SLICING

In this example, we have a my_string variable containing the string “Hello World”

```
my_string = "Hello World"
```



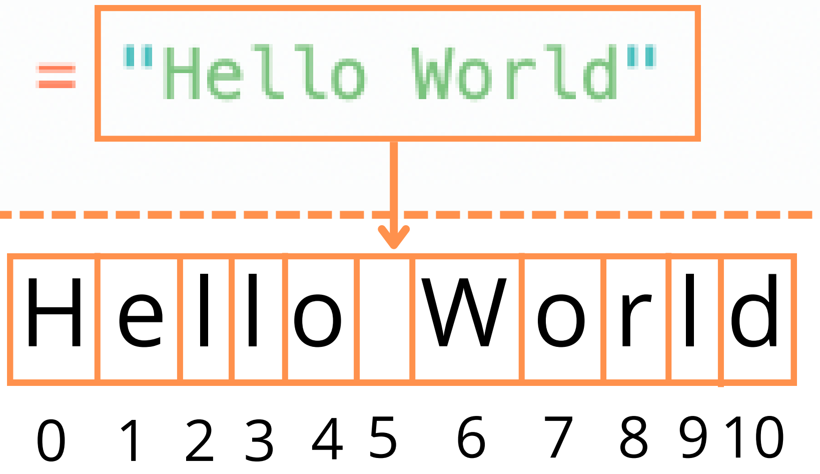
We can slice the string to get, for example, the first character from the string

```
first_char = my_string[0]

print(first_char)

# output will be H
```

## STRING OPERATIONS: SLICING

In this example, we have a my_string variable containing the string “Hello World”

```
my_string = "Hello World"
```



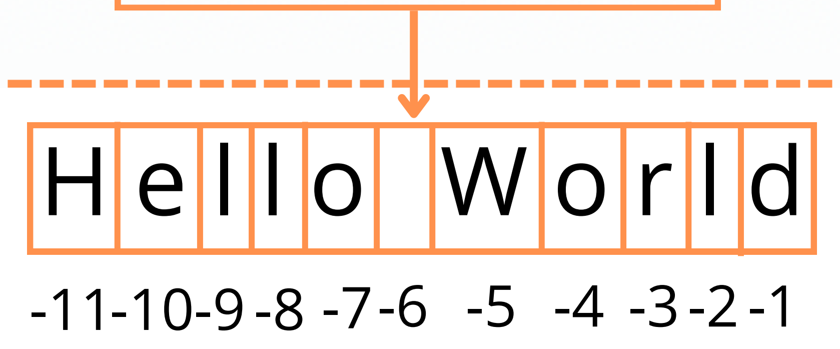
We can slice the string to get, for example, the first character from the string

```
first_char = my_string[0]

print(first_char)

# output will be H
```

## STRING OPERATIONS: SLICING

Python also consents inverse indexing, by using  negative position starting from -1 (last char)

```python
my_string = "Hello World"
```



We can slice the string using inverse indexing to get, for example, the last character

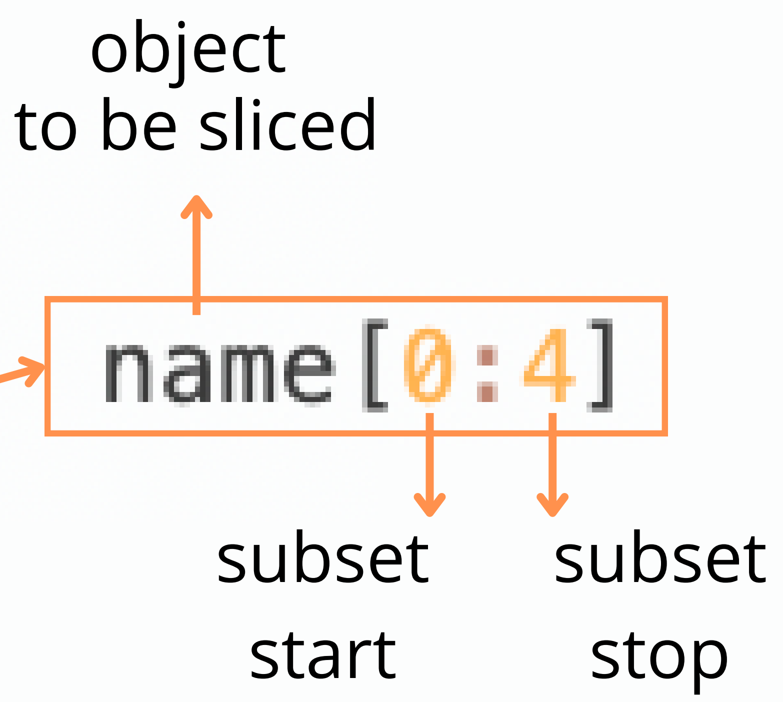
```python
last_char = my_string[-1]

print(last_char)

# output will be d
```

# STRING OPERATIONS: RANGE SLICING

Range slicing is a particular type of slicing, used to extract defined subsets from elements

- As for slicing, range slicing is performed using positional indexing and square brackets
- Range slicing as a proper sintax to be performed

```
name = "Andrea"

# Simple range slicing

subset = name[0:4]
```



## STRING OPERATIONS: RANGE SLICING

- **subset start:** positional index of the first element to be contained in the subset
- **subset stop:** positional index of the **first element to not be contained** in the subset

that means that **the element at position used as subset stop, will not be included in the subset**

```
print(subset)

# Output is Andr
# Remember, name[0] = A, name[1] = n, name[2] = d, name[3] = r, name[4] = e
```

## STRING OPERATIONS: RANGE SLICING

Range slice can also be performed using inverse positional indexing, without expliciting subset stop or start

```
name = "Andrea"
# Inverse positional indexing range slicing
subset = name[-2:]
print(subset)
# ea last two characters of the string
```

## STRING OPERATIONS: RANGE SLICING

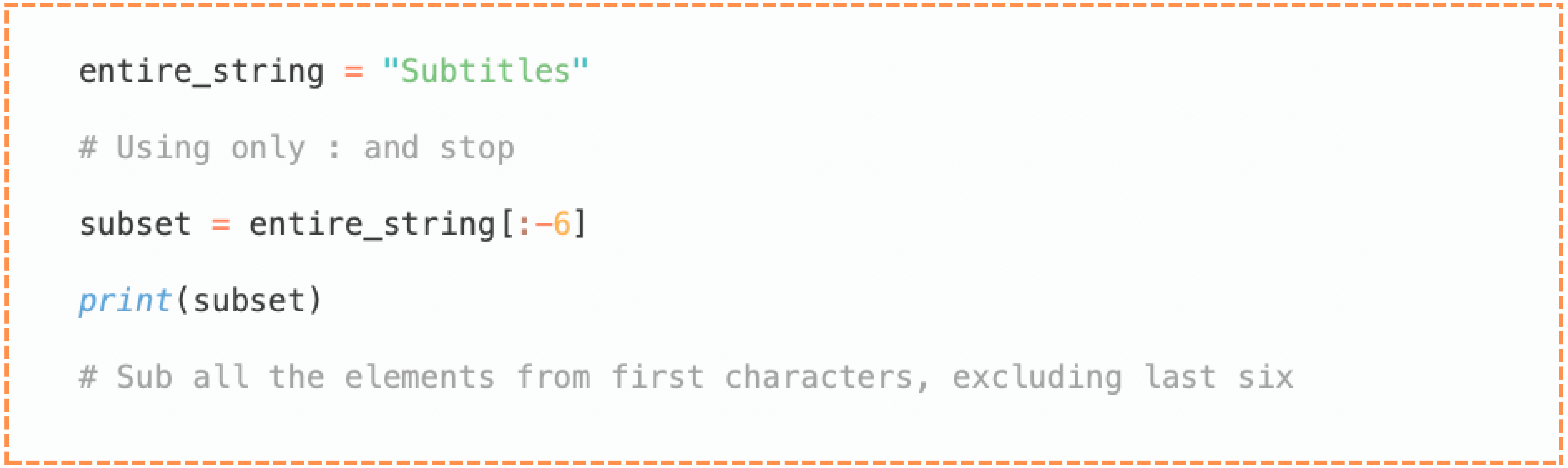
Range slicing can be used without declaring subset_start or subset_stop, to get all the values from or to a certain positional index

- **Not declaring stop**, will get all the elements from start (included) to the end.

```
entire_string = "Subtitles"

# Using only start and :

subset = entire_string[3:]

print(subset)

# titles (all the elements from the index 3 position to the end)
```

## STRING OPERATIONS: RANGE SLICING

- **Not declaring start**, will get all the elements from the first character to the stop positional index (excluded)



```
entire_string = "Subtitles"

# Using only : and stop

subset = entire_string[:-6]

print(subset)

# Sub all the elements from first characters, excluding last six
```

## STRING OPERATIONS: RANGE SLICING

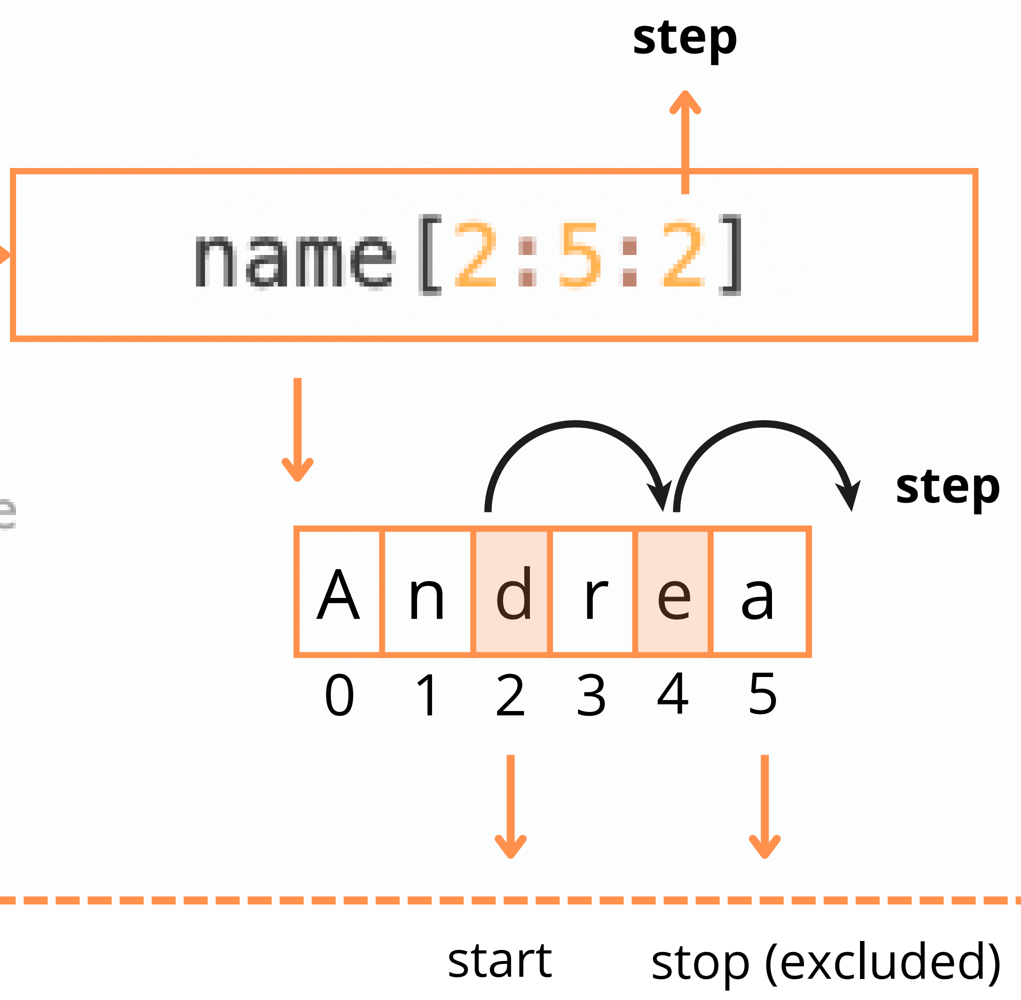
- **Not declaring both start and stop**, will get all the elements

```
entire_string = "Subtitles"

# Using only :

subset = entire_string[:]

print(subset)

# Subtitles
```

# STRING OPERATIONS: RANGE SLICING

Range slicing, can also handle a third parameter known as “step”

```python
name = "Andrea"

subset = name[2:5:2]

print(subset)
# Output is de
# You're taking all the characters in the string name
# from position 2 (third element, d)
# to position 5 (sixth element, a exluded)
# with step 2 (a character every two)
```



## STRING OPERATIONS: RANGE SLICING

Negative step  will result in the extracted subset but reversed (at least one between start and stop has to not be declared)

```
name = "Andrea"

subset = name[::-2]

print(subset)
# Output is arn
# first the string is reversed
#then everityhing (no start, no stop) is picked with step 2
```

## QUESTION TIME

PYTHON

## QUESTION 1:

What’s the output of the following program:

```
variable = "it's a string"

variable[0] = 'a'

variable = (variable + variable) * 2

print(variable)
```

1. erroneous code, TypeError at line 5
2. at’s a stringat’s a stringat’s a stringat’s a string
3. it’s a string
4. it’s a stringit’s a string

PYTHON

## QUESTION 2:

What’s the output of the following program:

```
variable = "Python"

print(variable[::-1])
```

1. TypeError at line 3
2. nohtyP
3. SintaxError at line 3
4. Pytho

## QUESTION 3:

What's the output of the following program:

```
variable = "Python"

x = variable[::2][:][-1]

print(x)
```

1. TypeError at line 3

2. P

3. o

4. None of the above

PYTHON

## QUESTION 4:

What is the output of the following python program:

```
variable = "Python"

x = variable[::2][:][0]

print(x)
```

1. P
2. Python
3. SyntaxError
4. None of the above

PYTHON

## QUESTION 5:

What is the output of the following python program:

```
variable = "Hey Joe"

x = 3 * (variable[0] + variable[-1])

print(x)
```

1.Joe

2. TypeError

3. HeHeHe

4. Hey Joe

PYTHON

## QUESTION 6:

What is the output of the following python program:

```
variable = "Hey Joe"

x = (3 * (variable[0] + variable[-1]))[2]

print(x)
```

1. H

2. SyntaxError at line 3

3. e

4. J

## EXERCISES