Week 1 quiz

Quiz

This quiz is worth 10 marks (0.5 marks per question). You can start whenever you like, and change your answers as often as you like, up until the due date and time. By then you need to have submitted your final answer to each question.

Question 1

How many syntax errors does the following code contain?

```
print ("Hi'   }
```

○ 0

○ 1

● 2

○ 3

Question 2

If you enter `6` when prompted by the following code it causes an error. Why?

```
num = input('Please enter a number: ')
print(num + 2)
```

○ The function `input` can only accept alphabetic characters

○ Since `num` is a variable it cannot be assigned a numerical value

● `num` gets assigned a string value and a string cannot be added to the number 2

○ Calculations cannot be performed inside the `print` function

Question 3

Consider the following piece of code:

PYTHON

```python
1 if (25 > 10):
2             print('25 is big')
3     else:
4                     print('25 is small')
```

Which lines contain an indentation syntax error?

● Lines 2, 3 and 4.

○ Just lines 2 and 4

○ Just lines 3 and 4.

○ Just line 3

Question 4

Suppose you want to comment-out multiple lines of code, such as the following three:

```
x = 10
x = x + 20
print(x)
```

In which one of the following ways could you do it?

○

```
##
x = 10
x = x + 20
print(x)
##
```

●

```
# x = 10
# x = x + 20
# print(x)
```

○

```
/#
x = 10
x = x + 20
print(x)
#/
```

○ You could do it in any of the above ways

Question 5

Which one of the following statements about objects and their attributes is true?

● Attributes can be functions but they need not be

○ Attributes can only be functions

○ Attributes cannot be functions

○ None of the above are true

Question 6

Suppose you want to apply the function `func` to the word "Hello". Which one of the following is the correct way to do so?

○

```
func(Hello)
```

●

```
func('Hello')
```

○

```
'Hello'(func)
```

○
```
func.Hello()
```

Question 7

Which one of the following is not a literal?

● x

○ "x"

○ 2

○ "2"

Question 8

Which one of the following claims about variables in Python is true?

● Variables must be declared before they can be assigned a value.

○ Once assigned a value, a variable cannot be assigned a different value.

○ Once assigned a value, a variable must always have some value.

○ None of the above are true.

Question 9

What should you infer from the following line of code:

```
FACTOR = 1.453
```

○ `FACTOR` is an important variable

● `FACTOR` is a constant and trying to change its value will cause an error

○ The value of `FACTOR` is intended to not change.

○ `FACTOR` can be used as a literal for the number 1.453

Question 10

Some expressions are simple and some are complex. Which one of the following expressions is a complex expression?

○ `23.5e-6`

○ `"Hello, world!"`

○ `my_long_variable_name`

● `1 + 2`

Question 11

Consider the following odd-looking line of code:

```
x(x)
```

Are there any values of the variable x for which this won't cause an error?

● No

○ Yes

Question 12

Consider the following two statements:

```
print(2 * '42')
print(2 + '42')
```

Why does the second statement cause an error whereas the first does not?

○ You can add integers in Python but you cannot multiply them.

○ You can add strings in Python but you cannot multiply them.

● You can multiply a string by an integer in Python but you cannot add an integer to a string.

○ You can add a string to an integer in Python but you cannot multiple a string by an integer.

Question 13

Why does the following code ouput "who" rather than "WHO"?

```
INITIALS = 'who'
INITIALS.upper()
print(INITIALS)
```

○ There is no `upper` method for strings.

○ `INITIALS` is a constant and cannot be modified.

○ The `print` function undoes the effect of the `upper` method.

● The `upper` method does not modify a string in place - it returns a new string.

Question 14

Why doesn't the following code generate an error, even though `cancelled` is not defined?

PYTHON

```python
done = True
if (done or cancelled):
    print('Over')
```

○ Python implicitly defines `cancelled` for us.

○ `cancelled` is a Python keyword and does not need to be defined.

● `or` is a short-circuiting operator, so `cancelled` is not evaluated.

○ `if` is a short-circuiting operator, so `cancelled` is not evaluated.

Question 15

Why does the following piece of code generate an error?

```
x = 5
if (x = 5):
    print('x is five')
```

○ There shouldn't be parentheses around `x = 5` in the second line.

○ `x = 5` should be `5 = x`.

● `x = 5` should be `x == 5`.

○ Since there is only one statement in the body of the `if` statement, it should be on the same line as `if`.

Question 16

Consider the following piece of code, containing a while statement:

PYTHON

```
1 x = 1
2 while (x <= 10):
3     if (x == 4): break
4     x = x + 1
```

How many times does line 4 get executed?

○ 1

● 3

○ 4

○ 10

Question 17

The following code generates an error if the user enters a letter:

PYTHON

```python
1 value = input('Please enter an integer: ')
2 value = int(value)
3 value = value**2
4 print(value)
```

You can prevent the error by using a `try ... except ...` statement. In which of the following ways could you use it?

○ Wrap it around line 1.

○ Wrap it around line 2.

● Wrap it around line 3.

○ Wrap it around line 4.

Question 18

Some statements are simple and some are compound. What is the defining difference?

○ Simple statements are easy to understand whereas compound statements are difficult.

○ Simple statements are fast to execute whereas compound statements are slow.

○ Simple statements contain other statements, whereas compound statements do not.

● Compound statements contain other statements, whereas simple statements do not.

Question 19

In which one of the following ways could you use the `ceil` function of the `math` module?

○

```
import math
x = math.ceil(3.5)
```

○

```
import math as m
x = m.ceil(3.5)
```

○

```
from math import ceil
x = ceil(3.5)
```

● You could use it in any of the above ways.

Question 20

Suppose you want to open a file called "results.txt" and write some results to it, but only if the file does not already exist. Which one of the following statements should you use?

○

```
f = open('results.txt', 'r')
```

○

```
f = open('results.txt', 'w')
```

●

```
f = open('results.txt', 'x')
```

○

```
f = open('results.txt', 'a')
```