TYPE-B:PYTHON FUNDAMENTALS:CH-6

1) From the following, find out which assignment statement will produce an error. State reason(s) too.

(a) x = 55
(b) y = 037
(c) z = 0o98
(d) 56thnumber = 3300
(e) length = 450.17
(f) !Taylor = 'Instant'
(g) this variable = 87.E02
(h) float = .17E - 03
(i) FLOAT = 0.17E - 03

sol:

1. y = 037 (option b) will give an error as decimal integer literal cannot start with a 0.
2. z = 0o98 (option c) will give an error as 0o98 is an octal integer literal due to the 0o prefix and 8 & 9 are invalid digits in an octal number.
3. 56thnumber = 3300 (option d) will give an error as 56thnumber is an invalid identifier because it starts with a digit.
4. !Taylor = 'Instant' (option f) will give an error as !Taylor is an invalid identifier because it contains the special character !.
5. this variable = 87.E02 (option g) will give an error due to the space present between this and variable. Identifiers cannot contain any space.
6. float = .17E - 03 (option h) will give an error due to the spaces present in exponent part (E - 03). A very important point to note here is that float is NOT a KEYWORD in Python. The statement float = .17E-03 will execute successfully without any errors in Python.
7. FLOAT = 0.17E - 03 (option i) will give an error due to the spaces present exponent part (E - 03).

2) How will Python evaluate the following expression ?

sol:

(i) 20 + 30 * 40	(ii) 20 - 30 + 40	(iii) (20 + 30) * 40	(iv) 15.0 / 4 + (8 + 3.0)
= 20 + 30 * 40	= 20 - 30 + 40	=(20 + 30) * 40	=15.0 / 4 + (8 + 3.0)
20 + 1200	-10+40	50 * 40	15.0 / 4 + 11.0
1220	30	2000	3.75 + 11.0=14.75

3) Find out the error(s) in following code fragments:

sol:

(i)
```
temperature = 90
print temperature
```
= the bracket in the print statement is missing.. the correct statement will be ,print(temperature)

(ii)

```
a = 30
b=a+b
print (a And b)
```

=the error is in the print statement . in there we should not use 'And' instead we should uses as: print(a,b)

(iii)

```
a, b, c = 2, 8, 9
print (a, b, c)
c, b, a = a, b, c
print (a ; b ; c)
```

= the mistake is again in the second print statement, we should not use ';' instead we should use print(a,b,c)

(iv)

```
X = 24
4 = X
```

= the error is in the identifier of the statement 2. integer alone cannot be and identifier.

(V)

```
print('x=',X)
```

= the error is ,the variable 'X' is not defined here

4) What will be the output produced by following code fragment (s) ?

(i)

```
X = 10
X = X + 10
X = X - 5
print (X)
X, Y = X - 2, 22
print (X, Y)
```

output:

```
15
13 22
```

(ii)

```
first = 2
second = 3
third = first * second
print (first, second, third)
first = first + second + third
third = second * first
print (first, second, third)
```

output:

```
2 3 6
11 3 33
```

(iii)

```
side = int(input('side') )  #side given as 7
area = side * side
print (side, area)
```

output:

```
side5
5 25
```

5) What is the problem with the following code fragments ?

(i)

```
  a = 3
    print(a)
  b = 4
    print(b)
  s = a + b
    print(s)
```

output:

```
  ERROR!
  Traceback (most recent call last):
   File "<main.py>", line 2
    print(a)
  IndentationError: unexpected indent
```

(ii)

```
  name = "Prejith"
  age = 26
  print ("Your name & age are ", name + age)
```

output:

```
  ERROR!
  Traceback (most recent call last):
   File "<main.py>", line 3, in <module>
  TypeError: can only concatenate str (not "int") to str
```

(iii)

```
  a = 3
  s = a + 10
  a = "New"
  q = a / 10
```

output:

```
  ERROR!
  Traceback (most recent call last):
   File "<main.py>", line 4, in <module>
  TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

6) Predict the output:

a)

```
  x = 40
  y = x + 1
  x = 20, y + x
  print (x, y)
```

output:

```
  (20, 81) 41
```

b)

```
x, y = 20, 60
y, x, y = x, y - 10, x + 10
print (x, y)
```

output:

```
50 30
```

c)

```
a, b = 12, 13
c, b = a*2, a/2
print (a, b, c)
```

output:

```
12 6.0 24
```

d)

```
a, b = 12, 13
print (print(a + b))
```

output:

```
25
None
```

7) Predict the output

```
a, b, c = 10, 20, 30
p, q, r = c - 5, a + 3, b - 4
print ('a, b, c :', a, b, c, end = '')
print ('p, q, r :', p, q, r)
```

output:

```
a, b, c : 10 20 30p, q, r : 25 13 16
```

8) Find the errors in following code fragment

a) `y = x + 5` - here x is not defined that must rise error ;
`print (x, Y)`

b) `print (x = y = 5)` - Python doesn't allow assignment of variables while they are getting printed

c) `a = input("value")` - as the 'input' statement take the input as string and we know that
`b = a/2` string does not undergo division operator thus line 2 rise error.
`print (a, b)`

9) Find the errors in following code fragment : (The input entered is XI)

```
c = int (input ( "Enter your class") )
print ("Your class is", c)
```

sol:

as the c is give ‘int (input ("Enter your class"))’ it only allows integer as an input but when we try to give the roman numbers it rises error

10) Consider the following code :

```
name = input ("What is your name?")
print ('Hi', name, ',')
print ("How are you doing?")
```

was intended to print output as

Hi <name>, How are you doing ?
But it is printing the output as :

Hi <name>,
How are you doing?
What could be the problem ? Can you suggest the solution for the same ?

sol:

The print() function appends a newline character at the end of the line unless we give our own end argument. Due to this behaviour of print() function, the statement print ('Hi', name, ',1) is printing a newline at the end. Hence "How are you doing?" is getting printed on the next line.To fix this we can add the end argument to the first print() function like this:

11) Find the errors in following code fragment :

```
c = input( "Enter your class" )
print ("Last year you were in class") c - 1
```

sol:

the first error is with a string this statement use ‘-‘ operator to minus 1 , but it is not possible. and the the ‘c-1’ statement given out of the print() statement. thus these are the 2 errors found in this code.

12) What will be returned by Python as result of following statements?

(a) >>> type(0)

Answer

<class 'int'>

(b) >>> type(int(0))

Answer

<class 'int'>

(c) >>>.type(int('0')

Answer

SyntaxError: invalid syntax

(d) >>> type('0')

Answer

<class 'str'>

(e) >>> type(1.0)

Answer

<class 'float'>

(f) >>> type(int(1.0))

Answer

<class 'int'>

(g) >>>type(float(0))

Answer

<class 'float'>

(h) >>> type(float(1.0))

Answer

<class 'float'>

(i) >>> type(3/2)

Answer

<class 'float'>

13) What will be the output produced by following code ?

(a) >>> str(print())+"One"
output: 'NoneOne'

(b) >>> str(print("hello"))+"One"
output: hello
'NoneOne'

(c) >>> print(print("Hola"))
output: Hola
None

(d) >>> print (print ("Hola", end = " "))
output: Hola None

14) Carefully look at the following code and its execution on Python shell. Why is the last assignment giving error ?

```
>>> a = 0o12
>>> print(a)
10
>>> b = 0o13
>>> c = 0o78
  File "<python-input-41-27fbe2fd265f>", line 1
  c = 0o78
      ^
SyntaxError : invalid syntax
```

sol:
Due to the prefix 0o, the number is treated as an octal number by Python but digit 8 is invalid in Octal number system hence we are getting this error.

15) Predict the output

```
a, b, c = 2, 3, 4
a, b, c = a*a, a*b, a*c
print(a, b, c)
```

output: 4 6 8

16) The id( ) can be used to get the memory address of a variable. Consider the adjacent code and tell if the id( ) functions will return the same value or not(as the value to be printed via print()) ? Why ?
[There are four print() function statements that are printing id of variable num in the code shown on the right.

sol:
In Python, variables are not "boxes" that hold values; rather, they are labels or references attached to objects in memory. The id() function returns a unique identifier for an object, which is its memory address in the CPython implementation.

Immutability and Object Reassignment: Integers are immutable objects. When you perform an operation like num = num + 3, you are not changing the value of the original integer object. Instead, a new integer object with the value 16 is created, and the num label is reassigned to reference this new object. The id() changes because num now points to a different memory location.

Integer Caching/Interning: Python optimizes memory usage by caching commonly used immutable objects, especially small integers (typically -5 to 256). For values within this range, if an object with that value already exists, Python reuses the existing object instead of creating a new one.

In the example, num is initially 13 (within the cached range).

After num = num + 3, num becomes 16, a new object is referenced, so the id() is different.

After num = num - 3, num becomes 13 again. Because 13 is cached and still in memory, Python reuses the original object for the value 13, so the id() returns to its initial value.

Type Change: When num is assigned the string "Hello", it refers to a completely different type of object in a different memory location, so the id() will be different again.

17) Consider below given two sets of codes, which are nearly identical, along with their execution in Python shell. Notice that first code-fragment after taking input gives error, while second code-fragment does not produce error. Can you tell why ?

(a)

```
>>> print(num = float(input("value1:")) )
value1:67

TypeError: 'num' is an invalid keyword argument for this function
```

sol:
You can't use num= inside print() because it treats it as a keyword argument; assign the value first, then print it.

(b)

```
>>> print(float(input("valuel:")) )
value1:67

67.0
```

sol:
This works because input() gets the value, float() converts it to a number, and print() displays the result.

18) Predict the output of the following code :

```
days = int (input ("Input days : ")) * 3600 * 24
hours = int(input("Input hours: ")) * 3600
minutes = int(input("Input minutes: ")) * 60
seconds = int(input("Input seconds: "))
time = days + hours + minutes + seconds
print("Total number of seconds", time)
```

If the input given is in this order : 1, 2, 3, 4

output:

```
Input days : 1
Input hours: 2
Input minutes: 3
Input seconds: 4
Total number of seconds 93784
```

19) What will the following code result into ?

```
n1, n2 = 5, 7
n3 = n1 + n2
n4 = n4 + 2
print(n1, n2, n3, n4)
```

output:

```
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 3, in <module>
```

NameError: name 'n4' is not defined

20) Correct the following program so that it displays 33 when 30 is input.

```
val = input("Enter a value")
nval = val + 30
print(nval)
```

sol:

```
val = int(input("Enter a value"))
nval = val + 30
print(nval)
```