Python Basics — Assignment Solutions

THEORY QUESTIONS

1. What is Python, and why is it popular?
Python is a high-level, interpreted programming language known for readable syntax, strong standard library, cross-platform support, and a large ecosystem (web, data, automation, ML). It's popular because it's easy to learn, productive, and has broad community and library support.

2. What is an interpreter in Python?
An interpreter executes Python code line-by-line at runtime, translating Python bytecode to machine instructions (or executing via a virtual machine). The standard CPython interpreter compiles source to bytecode then executes it on a VM.

3. What are pre-defined keywords in Python?
Keywords are reserved words with special meaning in Python (e.g., def, return, if, else, for, while, import, try, except, class, lambda, True, False, None). They cannot be used as identifiers.

4. Can keywords be used as variable names?
No — using a keyword as a variable name is a SyntaxError.

5. What is mutability in Python?
Mutability refers to whether an object's state can be changed after creation. Mutable objects (like list, dict, set) can be changed in-place; immutable objects (like int, float, str, tuple) cannot.

6. Why are lists mutable, but tuples are immutable?
Lists are implemented with a dynamic array structure that supports item assignment and resizing, so operations change the same object. Tuples are fixed-length and have no operations that change items — constructing a new tuple is required to change values.

7. What is the difference between '==' and 'is' operators in Python?
'==' checks value equality (do objects have equal values?). 'is' checks identity (do both names reference the same object in memory?). Example: a == b may be True while a is b may be False for distinct but equal objects.

8. What are logical operators in Python?
Logical operators combine boolean expressions: and, or, not.

9. What is type casting in Python?
Type casting (conversion) changes an object from one type to another, e.g., int('3') -> 3.

10. What is the difference between implicit and explicit type casting?
Implicit casting is automatic conversion by Python (e.g., int to float in expressions). Explicit casting is done by the programmer using constructors like int(), float(), str().

11. What is the purpose of conditional statements in Python?
Conditional statements (if, elif, else) allow branching: executing code only when a condition is True.

12. How does the elif statement work?
elif provides an additional condition after an if; it is checked only if previous if/elif were False. The first True branch executes.

13. What is the difference between for and while loops?
for loops iterate over a sequence (list, range, string). while loops run while a condition remains True and are typically used when iteration count is not known.

14. Describe a scenario where a while loop is more suitable than a for loop.
When repeatedly asking a user for correct input until they provide it, a while loop is more suitable because the number of attempts is not pre-known.

PRACTICAL QUESTIONS — Solutions (Python code examples)

For each practical problem below is sample code you can run in Python:

1. Print "Hello, World!"

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

2. Display your name and age

```
name = "Alice"
age = 21
print(f"My name is {name} and I am {age} years old.")
```

3. Print all pre-defined keywords using the keyword library

```
import keyword
print(keyword.kwlist)
```

4. Check if a given word is a Python keyword

```
import keyword
w = input("Enter a word: ").strip()
print(f"{w} is a keyword? ->", keyword.iskeyword(w))
```

5. Create a list and tuple and demonstrate changing an element

```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
# Lists are mutable
my_list[0] = 100
# Tuples are immutable: the following would raise an error if uncommented
# my_tuple[0] = 100
print('list after change ->', my_list)
print('tuple remains ->', my_tuple)
```

6. Function to demonstrate mutable vs immutable arguments

```
def modify(x_list, x_num):
    x_list.append(99)
    x_num += 1

lst = [1,2]
num = 10
modify(lst, num)
print('lst after modify:', lst)   # changed
print('num after modify:', num)   # unchanged
```

7. Basic arithmetic operations on two user-input numbers

```
a = float(input('Enter first number: '))
b = float(input('Enter second number: '))
print('Sum =', a+b)
print('Difference =', a-b)
print('Product =', a*b)
if b != 0:
    print('Quotient =', a/b)
else:
    print('Cannot divide by zero')
```

8. Demonstrate logical operators

```
a = 10
print(a > 5 and a < 20)
print(a < 5 or a % 2 == 0)
print(not (a == 10))
```

9. Convert user input from string to int, float, and boolean

```
s = input('Enter something (e.g. 5 or 3.14 or true): ').strip()
try:
    i = int(s)
except:
    i = None
```

```
try:
    f = float(s)
except:
    f = None
b = s.lower() in ('true','1','yes','y')
print('int:', i, 'float:', f, 'bool:', b)
```

10. Type casting with list elements
```
str_list = ['1','2','3']
int_list = [int(x) for x in str_list]
print(int_list)
```

11. Check if a number is positive, negative, or zero
```
n = float(input('Enter a number: '))
if n > 0:
    print('Positive')
elif n < 0:
    print('Negative')
else:
    print('Zero')
```

12. For loop to print numbers 1 to 10
```
for i in range(1,11):
    print(i)
```

13. Sum of all even numbers between 1 and 50
```
total = sum(i for i in range(1,51) if i%2==0)
print('Sum of even numbers 1..50 =', total)
```

14. Reverse a string using a while loop
```
s = input('Enter a string: ')
rev = ''
i = len(s) - 1
while i >= 0:
    rev += s[i]
    i -= 1
print('Reversed:', rev)
```

15. Factorial using a while loop
```
n = int(input('Enter non-negative integer: '))
if n < 0:
    print('Factorial not defined for negative numbers')
else:
    fact = 1
    i = n
    while i > 1:
        fact *= i
        i -= 1
    print(f'Factorial of {n} =', fact)
```