Assignment1

November 15, 2025

Q1. Basic Variable Assignment

Assign values to variables (age, name, is_student, gpa, courses) and print their types using type().

[5]:
```
age = 26
name = 'John'
is_student = True
gpa = 8.7
courses = ('Psychology', 'Mathematics', 'Biology', 'Economics')
print(type(age))
print(type(name))
print(type(is_student))
print(type(gpa))
print(type(courses))
```

```
<class 'int'>
<class 'str'>
<class 'bool'>
<class 'float'>
<class 'tuple'>
```

Q2. Datatypes Identification

100 -> int 3.14 -> float ‘c’ -> str ‘Hello’ -> str True -> bool None -> NoneType (1,2,3) -> tuple {‘a’:1} -> dict

Q3. Keywords check

Import the keyword module and write a program that checks if a word is a Python keyword.

[19]:
```
import keyword

word = input('Enter a word : ')
if keyword.iskeyword(word):
    print('Yes ', word, ' is a keyword')
else:
    print('Not a keyword')
```

```
Enter a word :  if
```

```
Yes  if  is a keyword
```

Q4. Literals Classification

Given a literal as string input, classify it as Integer, Float, String, Boolean, None, or Other.

[28]:
```
num = '123'
print('Integer: ', int(num))
print('Float: ', float(num))
print('String: ', str(num))
```

```
Integer:  123
Float:  123.0
String:  123
```

[29]:
```
print('Boolean:', bool(num)
```

```
  Cell In[29], line 1
    print('Boolean:', bool(num)
                                ^
_IncompleteInputError: incomplete input
```

[30]:
```
print('None:', None(num))
```

```
<>:1: SyntaxWarning: 'NoneType' object is not callable; perhaps you missed a
comma?
C:\Users\Ayoush Paul\AppData\Local\Temp\ipykernel_8788\494198750.py:1:
SyntaxWarning: 'NoneType' object is not callable; perhaps you missed a comma?
  print('None:', None(num))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 print('None:', None(num))

TypeError: 'NoneType' object is not callable
```

Q5. Type Casting Basics

Demonstrate explicit casting between int, float, str, and bool with valid and invalid cases.

[48]:
```
a = 3.7
print(int(a))

b = "42"
print(int(c))

c = True
print(int(c))
```

```
d = "hello"
print(int(d))
```

```
3
0
1
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[48], line 11
      8 print(int(c))
     10 d = "hello"
---> 11 print(int(d))

ValueError: invalid literal for int() with base 10: 'hello'
```

[46]:
```
a = 10
print(float(a))

b = "3.14"
print(float(b))

c = False
print(float(c))

d = "abc"
print(float(d))
```

```
10.0
3.14
0.0
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[46], line 11
      8 print(float(c))
     10 d = "abc"
---> 11 print(float(d))

ValueError: could not convert string to float: 'abc'
```

[49]:
```
a = 123
print(str(a))

b = 3.14
```

```
print(str(b))

c = True
print(str(c))
```

```
123
3.14
True
```

[50]:
```
a = 0
print(bool(a))

b = 42
print(bool(b))

c = "Hello"
print(bool(c))
```

```
False
True
True
```

Q6 — Mixed Types Arithmetic

Perform addition based on input type: numeric addition or string concatenation.

[55]:
```
a = 12
b = 2
c = '2'
d = '4'
print(a + b)
print(c + d)
print(type(c+d))
```

```
14
24
<class 'str'>
```

Q7. Temperature Converter

Convert between Celsius and Fahrenheit using user input and type conversion.

[60]:
```
cel = int(input('Enter Celcius:'))
fah = cel * (9/5) + 32
print(float(fah))
```

```
Enter Celcius: 24

75.2
```

Q8. Variables & Mutability

Explain mutable vs immutable with code showing list (mutable) and tuple (immutable) behavior.

[63]:
```
# Mutable in python means we can change the value.
l = [1,'hello',2.25,True]
l[2] = 3
print(l)
```

```
[1, 'hello', 3, True]
```

[65]:
```
# Immutable in python means we cannot change the value.
t = (1,'hello',2.25,True)
t[2] = 4
print(t)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[65], line 3
      1 # Immutable in python means we cannot change.
      2 t = (1,'hello',2.25,True)
----> 3 t[2] = 4
      4 print(t)

TypeError: 'tuple' object does not support item assignment
```

Q9. Input Validation (type casting + condition)

Take age as input, validate numeric, and classify into Child, Teenager, Adult, or Senior.

[68]:
```
age = int(input('Enter Age: '))
if age <= 12:
          print('Child')
elif age > 12 and age < 18:
          print('Teenager')
elif age >= 18 and age < 60:
        print('Adult')
else:
    print('Senior')
```

```
Enter Age:  13
```

```
Teenager
```

Q10. Keywords & Identifiers

Explain valid Python identifiers and write code to check invalid ones or keywords. Identifier is a name given to a variable to identify names in the program myVar - camel case -> Here first word starts with small letter and in all words first letter is capital. MyVar - Pascal case -> Here every first letter in a word is capital. my_var - snake case -> here we put underscore after each word.

[69]:
```
1myVar = 12
my-var = 3
my@var = 2
```

```
if = 2
my var = 12
```

```
  Cell In[69], line 1
    1myVar = 12
    ^
SyntaxError: invalid decimal literal
```

Q11. Advanced: Expression Parser

Accept simple expressions like '12 * 3.5' and evaluate using type casting and exception handling.

[72]:
```
a = '12 * 3.5'
print(float(a))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[72], line 2
      1 a = '12 * 3.5'
----> 2 print(float(a))

ValueError: could not convert string to float: '12 * 3.5'
```