Python Essentials (2)

June 4, 2024

1 Python Assignment: Essentials

1.1 1. Write a script that prints the square root of the integer 17. What class is the resulting value? Why?

[ ]:
```
import math

print(math.sqrt(17))
print(type(math.sqrt(17)))
```

```
4.123105625617661
<class 'float'>
```

The math.sqrt() function calculates the square root of a number. The resulting value is a float, which is why its type is <class 'float'>.

1.2 2. What error do you get when you run print(81 + 'Forty-two')? Can you explain the name of the error?

[ ]:
```
try:
    print(81 + 'Forty-two')
except TypeError as e:
    print("Error:", e)
```

```
Error: unsupported operand type(s) for +: 'int' and 'str'
```

This code attempts to add an integer (81) to a string ('Forty-two'). The TypeError is raised because Python doesn't allow addition between different types. (integer. string)

1.3 3. What is the result of the calculation 2 ** 3 / 2 + 8? What is the type? Why?

[ ]:
```
print(2 ** 3 / 2 + 8)
print(type(2 ** 3 / 2 + 8))
```

```
12.0
<class 'float'>
```

The expression follows the standard order of operations. The result is a float because the result of a division returns always a type float. When adding a float and an integer it returns a float.

1.4 4. Write a third line that uses a format string and the variables x and y to print '81 plus 23 equals 104'

[ ]:
```
x = 81
y = 23
print(f"{x} plus {y} equals {x + y}")
```

```
81 plus 23 equals 104
```

The f-string format allows easy insertion of variables into strings. It substitutes the values of x and y into the string for printing.

1.5 5. Write a fourth line that uses a format string and the variables x and y to print '81 divided by 23 equals 3 with a remainder of 12'

[ ]:
```
print(f"{x} divided by {y} equals {x // y} with a remainder of {x % y}")
```

```
81 divided by 23 equals 3 with a remainder of 12
```

The format string incorporates placeholders for variables x and y, performs integer division (x // y) to calculate the quotient, and uses the modulus operator (x % y) to determine the remainder.

1.6 6. Are dictionaries mutable or not?

Dictionaries in Python are mutable, which means that their contents can be modified after they are created. This allows for the addition, removal, and modification of key-value pairs within the dictionary.

1.7 7. Given my_dictionary as given above, write code that takes the second description, and prints the individual words of the description in reverse order.

[ ]:
```
my_dictionary = {
    'aardvark': 'a nocturnal badger-sized burrowing mammal of Africa, with long␣
 ↪ears, a tubular snout, and a long extensible tongue, feeding on ants and␣
 ↪termites.',
    'aardwolf': 'a black-striped nocturnal African mammal that feeds mainly on␣
 ↪termites.',
    'abaca': 'a large herbaceous Asian plant of the banana family, yielding␣
 ↪Manila hemp.',
}

description = my_dictionary['aardwolf']
words = description.rstrip('.').split()  # Remove dot at the end of the␣
 ↪sentence and split sentence in words
print(words)
print(" ".join(reversed(words))+ ".")
```

```
['a', 'black-striped', 'nocturnal', 'African', 'mammal', 'that', 'feeds',
'mainly', 'on', 'termites']
```

termites on mainly feeds that mammal African nocturnal black-striped a.

1.8 8. Write code that computes the sum of an arbitrary list of numbers, for example the list l = [1, 2, 3.0]

[ ]:
```
l = [1, 2, 3.0]
print(sum(l))   # The sum() function calculates the total of all elements in␣
↪the list.
```

```
6.0
```

1.9 9. Compute the factorial of 15 using a for loop

[ ]:
```
factorial = 1
for i in range(1, 16):  # Iterates through numbers from 1 to 15 and multiplies␣
↪them together to find the factorial.
    factorial *= i
print(factorial)
```

```
1307674368000
```

1.10 10. Print numbers divisible by 3, 5, or 7 in the range 0 to 100

[ ]:
```
for num in range(101):
    if num % 3 == 0 or num % 5 == 0 or num % 7 == 0:
        print(num)
```

```
0
3
5
6
7
9
10
12
14
15
18
20
21
24
25
27
28
30
33
35
36
39
```

1.11 11. Check if 12829 is a prime number

[ ]:
```python
def is_prime(n):    # function that takes some input
    if n <= 1:
        return False    # false since prime numbers are greater than 0
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(12829))
```

```
# Explanation: The function checks if the number is divisible by any number␣
 ↪other than 1 and itself.
```

```
True
```

1.12 12. Compute the inner product of two lists

[ ]:
```
def inner_product(list1, list2):
    return sum(a * b for a, b in zip(list1, list2)) # zip function pairs␣
 ↪corresponding elements which are then multiplied together and then the sum␣
 ↪of the new list is returned

# Test the function
list1 = [1, 2, 3]
list2 = [2, 3, 4]
print(inner_product(list1, list2))
```

```
20
```

1.13 13. Return a list containing unique values from the input list

[ ]:
```
def unique_values(lst):
    count_dict = {}
    for item in lst:
        count_dict[item] = count_dict.get(item, 0) + 1

    unique_values_list = [key for key, value in count_dict.items() if value ==␣
 ↪1]
    return unique_values_list

def make_unique(lst):
    return list(set(lst))

# Test the functions
input_list = [1, 2, 3, 2, 4, 5, 1]

# Test the first function
unique_values_result = unique_values(input_list)
print("Unique values:", unique_values_result)

# Test the second function
make_unique_result = make_unique(input_list)
print("List with unique values:", make_unique_result)
```

```
Unique values: [3, 4, 5]
List with unique values: [1, 2, 3, 4, 5]
```

1.14 Exercise 14: Compute the surface area and circumference of a circle

[ ]:
```python
def circle_info(radius):
    surface_area = math.pi * radius ** 2
    circumference = 2 * math.pi * radius
    return surface_area, circumference

# Test
radius = 1
print(circle_info(radius))
```

```
(3.141592653589793, 6.283185307179586)
```

1.15 Exercise 15: Check if any letters from the first string occur in the second string

[ ]:
```python
def check_letters(string1, string2):
    for char in string1:    # iterate over each char
        if char in string2:
            return True
    return False

# Test
print(check_letters("hello", "world"))
print(check_letters("abc", "def"))
```

```
True
False
```

1.16 Exercise 16: Check if a number is prime

[ ]:
```python
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

print(is_prime(7))
print(is_prime(10))
```

```
True
False
```

1.17 Exercise 17: Return a list of all primes smaller than N

[ ]:
```python
def primes_smaller_than_N(N):
    prime_list = []
    for num in range(2, N):
        if is_prime(num):
            prime_list.append(num)
    return prime_list

# Test
print(primes_smaller_than_N(20))
```

```
[2, 3, 5, 7, 11, 13, 17, 19]
```

1.18 Exercise 18: Define a Point class with x and y attributes

[ ]:
```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

#Test
point1 = Point(1, 2)
print(point1.x, point1.y)
```

```
1 2
```

1.19 Exercise 19: Define a Translation class with a transform method

[ ]:
```python
class Translation:
    def __init__(self, dx, dy):
        self.dx = dx
        self.dy = dy

    def transform(self, point):
        point.x += self.dx
        point.y += self.dy
```

1.20 Exercise 20: Test the Point and Translation classes

[ ]:
```python
point = Point(3, 4) # create point
translation = Translation(4, 2) # create translation
translation.transform(point)    # apply transformation on point
print(f"Translated Point: ({point.x}, {point.y})")
```

```
Translated Point: (7, 6)
```