Day - 2

**Data Types &**

**String manipulation**

Data Types, Numbers, Operators, Type Conversion, f-strings

**2.1 Data Types**

1. String
2. Integer
3. Float
4. Boolean

* Index of string - SUBSCRIPTING: **"Hello"[3]** gives 4th 'l'.
* Declaring Integers: Just write the number, no keyword needed.
* Other format: Other thing is **123456** is can be written as **123\_456** it is slimier to *123,456*.
* Declaring floats: Just write the number, no keyword needed.
* Declaring Boolean: Just write ***True*** or ***False***, no keyword needed. Notice Uppercase is used.

padded\_questions\_in\_batch = **np.array**(**apply\_padding**(questions\_in\_batch, questnWrd2Int))

padded\_answers\_in\_batch = **np.array**(**apply\_padding**(answers\_in\_batch, ansWrd2Int))

**yield** padded\_questions\_in\_batch, padded\_answers\_in\_batch

padded\_questions\_in\_batch = **np.array**(**apply\_padding**(questions\_in\_batch, questnWrd2Int))

padded\_answers\_in\_batch = **np.array**(**apply\_padding**(answers\_in\_batch, ansWrd2Int))

**yield** padded\_questions\_in\_batch, padded\_answers\_in\_batch

**2.2 Type casting & checking with type()**

**type**(data)

* Type casting: Casting in python is done using constructor functions:

1. **int()** - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
2. **float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
3. **str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals

* Example:

|  |  |  |
| --- | --- | --- |
| #*Integers:*  x = **int**(1)   #*x will be 1*  y = **int**(2.8) #*y will be 2*  z = **int**("3") #*z will be 3* | #*Floats:*  x = **float**(1)     #*x will be 1.0*  y = **float**(2.8)   #*y will be 2.8*  z = **float**("3")   #*z will be 3.0*  w = **float**("4.2") #*w will be 4.2* | #*Strings:*  x = **str**("s1") #*x will be 's1'*  y = **str**(2)    #*y will be '2'*  z = **str**(3.0)  #*z will be '3.0'* |

* Exercise 2.1: sum of two digit number

two\_digit\_number = **input**("Type a two digit number :")

**print**("Type of this input is : " + **type**(two\_digit\_number))

type casting **is** happening below

sum = **int**(two\_digit\_number[1]) + **int**(two\_digit\_number[0])

**print**(sum)

**2.3 Mathematical operations**

Similar to other programming languages.

* Dividing ***int*** returns ***float*** output.
* Exponent sign: In python " **\*\*** " is used for power. c/c++/Java/C# has no this power operator.
* Order of precedence matters: remember "PEMDASLR"

|  |  |
| --- | --- |
| * Exercise 2.2: Change following to return 3.0, instead of 7.0. | #*POMDASLR rule applied*  **print**(3\*3 + 3 / 3 -3 )  **print**(((3\*(3 + 3)) / 3) -3 ) |

* Exercise 2.3: BMI calculator (Body-Mass-Index)

weight = **input**("Enter weigt : ")

heihgt = **input**("Enter height : ")

bmi = **int**(weight)/(**int**(heihgt)\*\*2)

**print**("The result is : " + **str**(**int**(bmi)))

**2.4 Rounding numbers in python**

* round(): Use **round**(float, precision)
* floor division " // " : **print(8 // 3)** gives floored number directly. As an int.
* Shorthand operators: **x = x/2** is similar to **x /= 2**. In form "variable **operator=** variable"

**2.5 f-string formatted output**

* In this case we don’t need any type conversion.
* Also don’t need "+" for concatenation. It's more like C's formatted output.

#*f-string*

score = 0

height = 1.8

isWinning = **True**

**print**(f"your score is {score}, your height is {height}. Are you winning: {isWinning}")

* As you see data is goes inside {}. And format is

**print**(**f**"{data1}string{data2}string{data3}… .. .. ")

* Exercise 2.3: Life in weeks.

age = **input**("What is oue current age? :")

remaining\_year = 90 - **int**(age)

remaining\_month = remaining\_year\*12

remaining\_week = remaining\_year\*52

remaining\_day = remaining\_year\*365

message = f"If you live upto 90 years then you have remaining {remaining\_year} years, or {remaining\_week} weeks, or {remaining\_month} months, or {remaining\_day} days"

**print**(message)

* Exercise 2.4: Final project, Tip calculator:

total = **input**("What was the total bill? : $")

percentage = **input**("What parcentage tip ypu like to give? 0, 10, 12, or 15 ? :")

split = **input**("How many people you want to split the bill? :")

payTotal = **float**(total) + (**float**(percentage)/100.0)\***float**(total)

finalPayment = **round**(payTotal/**int**(split), 2)

**print**(f"Each person should pay: ${finalPayment}")

**2.6 More formatted output:**

What happens if the result shows 14.0 instead of 14.00?

**"{:.2f}".format(finalPayment)**

Above formatting displays any number in 2 digit precision float number.

total = **input**("What was the total bill? : $")

percentage = **input**("What parcentage tip ypu like to give? 0, 10, 12, or 15 ? :")

split = **input**("How many people you want to split the bill? :")

payTotal = **float**(total) + (**float**(percentage)/100.0)\***float**(total)

finalPayment = **round**(payTotal/**int**(split), 2)

**print**(f"Each person should pay: ${finalPayment}")

**#*to inspect formatting works : use $150  with 12% splitted to 5***

**formatted = "{:.2f}".format(finalPayment)**

**print(f"Each person should pay (Formatted): ${formatted}")**