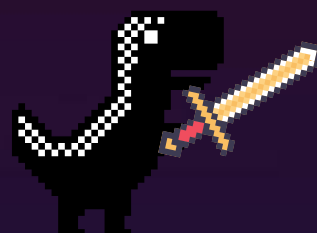




# 2 LEARNING BASICS


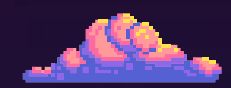


EXIT





# OBJECTIVES

- Creating and manipulating **variables** such as integers, strings and lists.
  - Using **Boolean** expressions in conditional statements such as if ... else.
  - Use iterative structures like for loops and while loops.
  - Deep dive into functions which are **built-in** or user defined.
  - Exploring the most common **errors** and **bugs** when learning.
- 
- 



# 2.1

## CREATING VARIABLES

2

numbers

S

strings



lists



more





# 2.1 CREATING VARIABLES

We label and store data elements in memory by creating

**variables**

We shall pick unique

**names**

that describes the data as good labels.

To assign it a value we simply use the equals (=) sign.

**condition**



# WHAT DOES THIS DO?

```
coins = 0  
print(coins)
```

Value Stored

Variable Name

For a coin collector game  
a good label to store the  
number of coins  
collected would be  
"coins".

A. input

B. output value

C. assign value

★ Multiple Choice



# NUMBER VARIABLES

Number variables are useful in games to perform mathematical **calculations**

We can label and store whole numbers in a variable of type **integer**

We can label and store **real** numbers in a variable of type float.

**float**



# What is happening?

```
coins = 100  
bonus = 30  
score = (coins * 2) + 30  
print("Your score: ", score)
```

Arithmetic



Short Answer



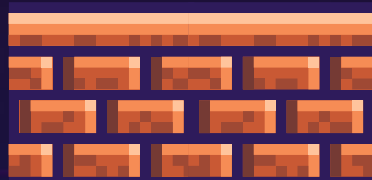
# INTERPRET CODE LIKE A PRO

As the user is playing we use an integer variable to store the number of coins collected and bonus coins that are awarded at the end. The score the player sees on the screen is based on a calculation that has coins and bonus.



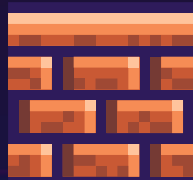
# INTEGER OR FLOAT?

16



age

-3.1



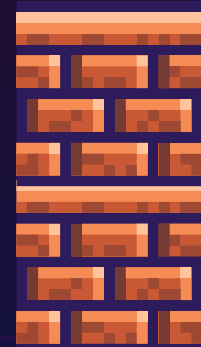
temperature

24.6



distance

3



lives



# ARITHMETIC OPERATIONS

```
a = 10
b = 2
c = a + b # addition
print(c)
c = a - b # subtraction
print(c)
c = a * b # multiplication
print(c)
c = a / b # division
print(c)
```

12

8

20

5

# GUESS THE OUTPUT

```
balance = 350  
price_of_cat = 250  
balance = balance - price_of_cat  
print(balance)
```

A. 1.4

B. 600

C. 100

★ Multiple Choice

# MORE ARITHMETIC

```
a = 5.0
b = 2.0
c = a % b # remainder after division
print(c)
c = a // b # floor division
print(c)
c = a ** b # power
print(c)
```

```
1.0
2.0
25.0
```



# LESSON CHALLENGE

- Time to put the theory into practice.
- There is a warm-up exercise and a more difficult exercise.
- You need to answer questions that are marked by a todo.
- You must use all you learned so far.





# STRING VARIABLES



String variables are used to store data that is **alphanumeric** in nature.

We can label and store a sequence of **characters**, "Hi there, Tom!".

Any keyboard **input** the user should be stored in a string variable.

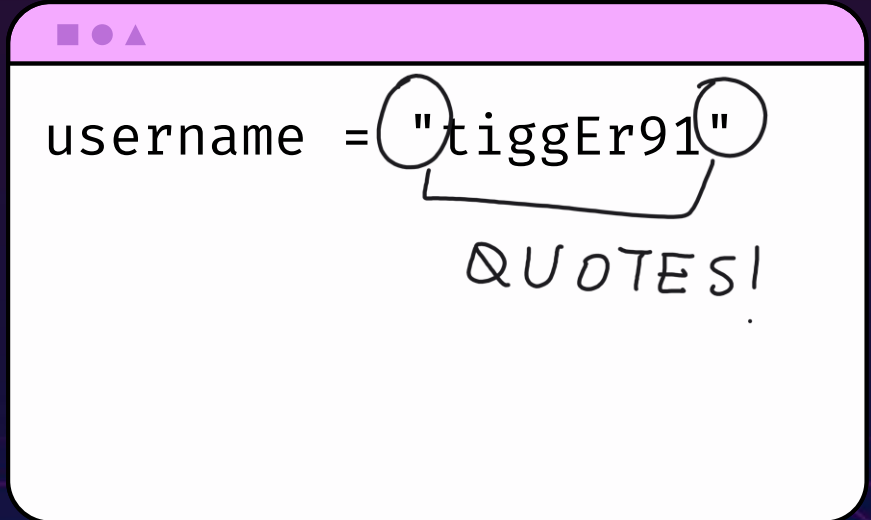
**real**





# STRING VARIABLES

- Strings can be assigned to variables.
- In this example we are assigning the string tiggEr91 to the username variable.
- You need to use quotation marks to indicate the start and end of a string.
- String variables in Python are flexible and you can join multiple string variables to create a new one.



```
username = "tiggEr91"
```

QUOTES!



# WHAT DOES THIS DO?

## Quote Marks

Quotation marks indicate that the value of a variable is of type string.

```
username = "tiggEr91"  
greeting = "Hi there, " + username
```

## Plus Operator

A. input

B. join strings

C. loop

★ Multiple Choice



# DID YOU UNDERSTAND?

## COMPLETE THE PROGRAM

Write a program that stores a suffix at in a variable and create other string variables for possible words by joining a prefix with the suffix.

end part of the word  
↓

`suffix = _____`

`_____ = "h" + suffix`

`bat = "b" + _____`

`print("Words ending with ", suffix)`

`print(hat, bat)`

hat

suffix

"at"



Fill in the Blanks





# STRING VARIABLES

- Python enables us to perform more complex operations on string variables by giving us a number of built-in functions.
- For a word search game it might be useful to count the number of characters in a string using the function len() to award the player points.
- A gentle reminder that a built-in function is a bundle of code that we can instantly use when we install Python.



# GUESS THE OUTPUT



```
user_word = "amaze"  
count_z = user_word.count("z")  
word_length = len(user_word)  
bonus_points = count_z * 2  
points = word_length + bonus_points  
print(points)
```

A. 1

B. amaze

C. 7

★ Multiple Choice



# INTERPRET CODE LIKE A PRO

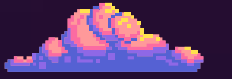
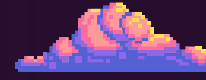
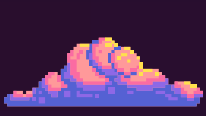
As the user is playing we use a string variable to store a word found . Points are awarded based on the length of the word. Additional bonus points are awarded for the occurrence of special letters like "z". The program displays the total points awarded for the given word.



# LESSON CHALLENGE

- Time to put the theory into practice.
- You will build a small component of a word search game.
- You must use all you learned so far.
- Find your task!





# LIST VARIABLES

A list is a data structure that can store a collection of data items.

We can store multiple data items under one unique label.

Every data item in a list has a particular position which is a number.

output





# LIST VARIABLES

**One label**

```
fruit = ["apple", "banana", "cherry", "apple"]
```

**Collection**

Multiple values  
separated by a comma  
inside square brackets

```
numbers = [6, 7, 13, 2, 45]
```



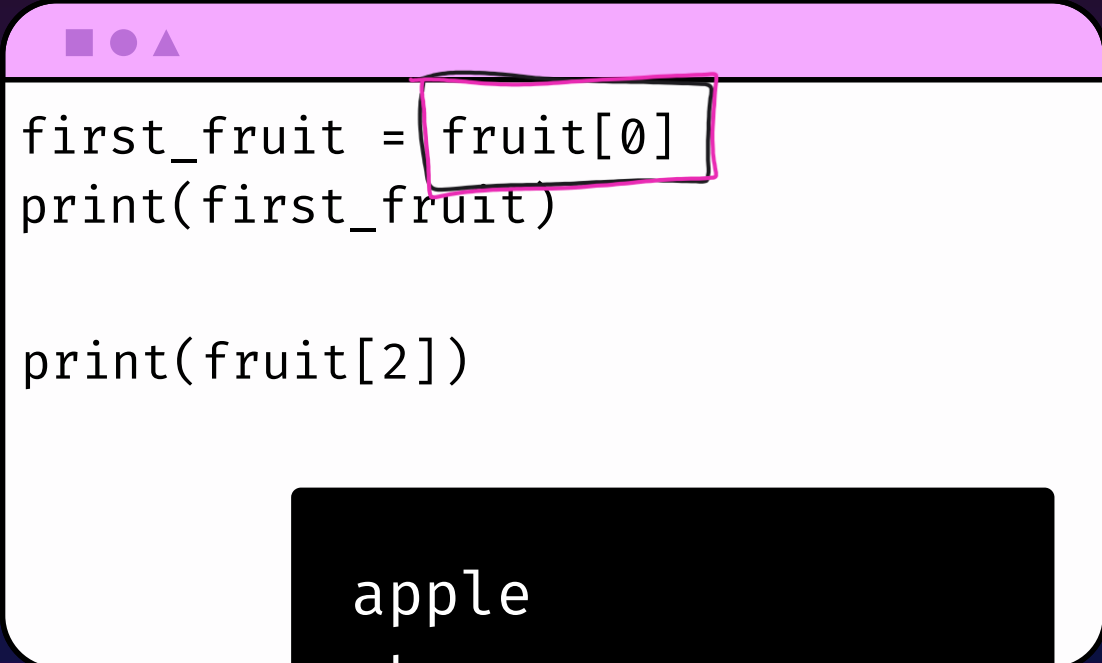
**Position**

The first position or index  
in a list is 0



# LIST VARIABLES

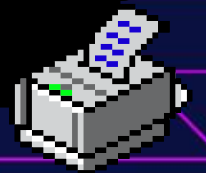
- Working with lists in Python is easy.
- An item from a list can be accessed by typing the name of the list and the position within square brackets.
- Each data item can be treated like a regular variable.
- Just remember that the first position is 0.



```
first_fruit = fruit[0]  
print(first_fruit)  
  
print(fruit[2])
```



apple  
cherry







# LIST VARIABLES

- Python enables us to perform more complex operations on list variables by giving us a number of built-in functions.
- For example count the number of occurrences of a data item in a list of work using the function `count()`.
- A gentle reminder that a built-in function is a bundle of code that we can instantly use when we install Python.
- We will cover functions in more detail very soon.



# LIST OPERATIONS

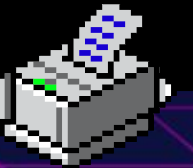
```
1.fruit = ["apple", "banana", "cherry", "apple"]  
2.apple_count = fruit.count("apple")  
3.print(apple_count)  
4.fruit.reverse()  
5.print(fruit)
```

## Output List

You can easily display  
a list variable using the  
print function

2

['apple', 'cherry', 'banana', 'apple']



# GUESS THE OUTPUT



```
my_numbers = [7, 2, 1, 4]  
print(my_numbers[3])
```

A. 7

B. 1

C. 4

★ Multiple Choice



# LESSON CHALLENGE

- Time to put the theory into practice.
- You will continue to build a small component of a word search game.
- You have to use all you learned so far.
- Find your task!

